AKBC 2026

The 11th Workshop on Automated Knowledge Base Construction

Proceedings of the Workshop

October 28, 2026

The AKBC organizers gratefully acknowledge the support from the following sponsors.

Diamond Level





©2026 Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
317 Sidney Baker St. S
Suite 400 - 134
Kerrville, TX 78028
USA
Tel: +1-855-225-1962
acl@aclweb.org

ISBN 979-8-89176-464-4

Preface

These proceedings present the contributions to the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), co-located with EMNLP 2026 and held on October 28, 2026 in Budapest, Hungary. AKBC brings together researchers and practitioners working on the construction, integration, and use of structured knowledge in the era of large language models (LLMs). While LLMs have revolutionized natural language processing, they remain prone to hallucinations, reasoning "mode-collapse" in open-ended generation, and a lack of factual provenance. Knowledge Bases (KBs) address a key missing piece of the generative era: They serve as ground truth for fact verification, the semantic backbone for constrained decoding in generation, and a resource behind Retrieval-Augmented Generation (RAG). AKBC sits at the intersection of natural language processing, knowledge representation, databases, and machine learning, with a particular focus on how symbolic structure can ground, constrain, and enhance generative models, both at training time and at inference time.

AKBC 2026 is centered around a distinguished lineup of invited speakers who provide the intellectual core of the workshop: Denny Vrandečić (Wikimedia Foundation), founder of Wikidata and co-creator of Semantic MediaWiki, who has shaped the landscape of collaborative knowledge bases and the Semantic Web; Mausam (IIT Delhi and University of Washington), an AAAI Fellow whose pioneering work spans information extraction, neuro-symbolic AI, and LLM agents for knowledge and reasoning; Heng Ji (University of Illinois Urbana-Champaign), a Grainger College of Engineering professor and ACL Fellow known for her work on multimedia multilingual information extraction, knowledge-enhanced LLMs, and her leadership of the NIST TAC Knowledge Base Population task; Alon Halevy (Google), an ACM Fellow recognized for his contributions to data integration and knowledge representation, whose early work on web data management and schema matching laid foundations for modern knowledge extraction; Dan Roth (University of Pennsylvania/Oracle), an ACL Fellow whose research integrates machine learning and reasoning for natural language understanding; Amir Globerson (Tel Aviv University/Google), whose research spans structured prediction, deep learning, and language and visual understanding; and Parisa Kordjamshidi (Michigan State University), whose pioneering research on spatial language grounding and multimodal LLMs bridges language understanding with visual and spatial reasoning. The workshop is designed to facilitate exchange between these senior researchers and the next generation of researchers presenting poster sessions, creating an intimate and collegial setting for dialogue across career stages.

The workshop follows a successful series of previous editions: as an independent conference in 2022, 2021, 2020, 2019; as a workshop at NeurIPS in 2017, NAACL in 2016, and NeurIPS in 2014; at CIKM in 2013 and NAACL in 2012; and as a stand-alone event in Grenoble, France in 2010.

We received 58 research and vision submissions and accepted 27 papers (23 direct submissions and 4 ARR commitments), an acceptance rate of 46.6%. The program combines invited talks, featured paper presentations, a poster session, a shared task session, and community discussion.

The shared task asked participants to predict the complete set of correct object strings for given subject – relation pairs, going beyond standard factual question answering to construct complete and precise knowledge base entries. Seventeen teams describe their systems in these proceedings. The winning teams were: "Coverage Is Not Knowledge: Why Closed-Book Knowledge Base Construction Looks Selection Bound" by Maksim Silchenko (winner); "MAVE: Multi-Angle Voting for Emission in Closed-Book Knowledge Base Construction" by Jonas Kunze and Andreas Both (second place); and "The Instruction Tax on Factual Recall: Base Models Can Beat Their Instruction-Tuned Counterparts" by Christophe Khalil and Aya Mourad (third place). We congratulate all three teams and thank everyone who took part!

We also gratefully acknowledge the support of our sponsors, SLB (formerly Schlumberger) and Bloomberg, whose generous contribution made this workshop possible.

Finally, we thank all authors for their contributions, the members of the program committee for their careful reviews and constructive feedback, and all participants for advancing research on knowledge base construction and knowledge-grounded generation.

The AKBC Organizing Committee
Jan-Christoph Kalo, University of Amsterdam, Netherlands
Ndapa Nakashole, University of California, San Diego, USA
Simon Razniewski, ScaDS.AI Dresden/Leipzig & TU Dresden, Germany
Fabian M. Suchanek, Télécom Paris, Institut Polytechnique de Paris, France
Andreas Vlachos, University of Cambridge, United Kingdom
Andrew McCallum, University of Massachusetts Amherst, USA

Organizing Committee

Workshop Chairs

Jan-Christoph Kalo, University of Amsterdam, Netherlands
Ndapa Nakashole, University of California, San Diego, USA
Simon Razniewski, ScaDS.AI Dresden/Leipzig & TU Dresden, Germany
Fabian M. Suchanek, Télécom Paris, Institut Polytechnique de Paris, France
Andreas Vlachos, University of Cambridge, United Kingdom
Andrew McCallum, University of Massachusetts Amherst, USA

Shared Task Organizers

Jan-Christoph Kalo, University of Amsterdam
Tuan-Phong Nguyen, Vietnam National University
Simon Razniewski, ScaDS.AI & TU Dresden
Bohui Zhang, King's College London

Program Committee

Program Committee

Sameer Singh, University of California, Irvine
Gerhard Weikum, Max Planck Institute for Informatics
Sunita Sarawagi, IIT Bombay
Annalisa Gentile, IBM
Nitisha Jain, Microsoft Research Cambridge
Sören Auer, University of Hannover
Danqi Chen, Princeton University
Doug Downey, Allen Institute for AI / Northwestern University
Zifeng Ding, Cambridge University
Moy Yuan, Amazon
Paolo Papotti, EURECOM
Gerard de Melo, Hasso Plattner Institute
Steffen Staab, University of Stuttgart
Rainer Gemulla, University of Mannheim
Luis Galárraga, INRIA Rennes
Lihu Chen, Imperial College London
Advitya Gemawat, Microsoft Research
Estevam Hruschka, Megagon Labs
Jingbo Shang, University of California, San Diego
Roi Cohen, HPI
Hiba Arnaout, Darmstadt University
Shrestha Ghosh, Tübingen University
Paul Groth, University of Amsterdam
Filip Ilievski, Vrije Universiteit Amsterdam
Russa Biswas, Aalborg University
Derry Wijaya, Monash University Indonesia
Pierre-Henri Paris, Paris-Saclay University
Daniel Daza, Vrije Universiteit Amsterdam
Jieying Chen, Vrije Universiteit Amsterdam
Selene Báez Santamaría, University of Zurich
Mohamed Gad-Elrab, Bosch Research
Benno Kruit, Amsterdam UMC
Andrea Schimmenti, Università di Bologna
Roman Klinger, University of Bamberg
Jay Pujara, USC Information Sciences Institute
Zachary Ives, University of Pennsylvania
Samy Haffoudhi, Institut Polytechnique de Paris
François Crespin, Institut Polytechnique de Paris
Christian Bizer, University of Mannheim
Aparna Varde, University of Southern Denmark
Wassila Ouerdane, CentraleSupélec, Paris-Saclay University
Raphaël Troncy, EURECOM
Jonathan Berant, Tel-Aviv University/Google DeepMind
David Campos, University of Amsterdam
Robin Jia, University of Southern California
Sai Kolapudi, Amazon Web Services

Table of Contents

Epistemic Twins: Enabling a Symbolic Science of Language Model Knowledge
Simon Razniewski, Shrestha Ghosh, Luca Giordano, Yujia Hu, Josua Kowalzik and Tuan-Phong Nguyen 1

Auditing a KB Elicitation of Frontier LLM Knowledge: A Multi-dimensional Analysis of GPTKB v1.5
Shrestha Ghosh, Luca Giordano, Yujia Hu, Tuan-Phong Nguyen and Simon Razniewski 8

Edge Knowledge Bases for Sustainable and Explainable AI
Aparna S. Varde, Emil Njor, Francesco Daghero and Mahyar Tourchi Moghaddam 19

A Temporal Knowledge Graph for Music Festival Lineup Forecasting
Julia Gastinger, Thilo Dieing, Christian Meilicke and Heiner Stuckenschmidt 26

The Canonical Order Problem: When Large Language Models Are Unreliable Knowledge Bases for Multi-Valued Relations
Timo Pierre Schrader, Annemarie Friedrich, Simon Razniewski and Lukas Lange 45

FALCON: A Model and Dataset Agnostic Framework for Synthetic Data Generation for NL2SQL Pairs
Darian Lee, Shannon Rumsey, Jack St. Clair, Xinyi Tang, Aditya Bansal and Yuanming Shi . . 56

When Relevance Is Not Evidence: A Controlled Diagnostic of Claim-Conditioned Support Ranking
Ke Lu 88

Fine-Grained Table Retrieval Through the Lens of Complex Queries
Wojciech Kosiuk, Xingyu Ji, Yeounoh Chung, Fatma Özcan and Madelon Hulsebos 94

EventSocraticKG: Event-Centric Knowledge Graph Construction for Narrative Retrieval-Augmented Generation
William Rosengren 108

HTTP for Meaning: Towards Linguistically Complete Knowledge Representations with Knowledge State Protocol
Ian Yuen 128

When String Matching Stands In for a Knowledge Base: A Role-Aware Audit of WikiBigEdit
Edison Yang, Neel Marripalapu and Akshaj Agadi 139

MUSART: A Scalable Multi-Domain Benchmark for Measuring Factual Recall in Large Language Models
Rajaa El Hamdani, Thomas Bonald and Fragkiskos Malliaros 151

Scaling E-Commerce Attribute Extraction with Parallel Decoding
Nikhita Vedula, Dushyanta Dhyani, Bryan Wang and Shervin Malmasi 169

Evidence-Carrying Claims: Fail-Closed Verification Gates for Language-Model-Assisted Research as Knowledge Base Construction
Robert Sneiderman 178

Evidence-Guided Schema Normalization for Temporal Tabular Reasoning
Ashish Thanga, Vibhu Dixit, Abhilash Shankarampeta and Vivek Gupta 185

GRACE: Graph-Grounded Reflective Agent Copilot Engine for Expert-in-the-Loop Knowledge Expansion
John Seon Keun Yi, Joshua R. Minot and Dokyun Lee 224

Can We Do Interpretable NLI with Graphs Based on Atomic Propositions?
Younes Boufouss, Luc Pommeret, Thomas Gerald, Patrick Paroubek and Sophie Rosset 234

A Rhetorical Citation Graph Model for Characterizing Scientific Findings: Resource Construction and Application to Plant Health
Anne-Sophie Foussat, Robert Bossy, Vincent Guigue, Nicolas Sauvion and Claire Nédellec . 247

Schema-Guided Extraction of Interaction Knowledge Bases from Narrative Text: A Folktale Case Study
Sergey Kovalchuk, Anastasia Ugleva and Oleg Kubryak 261

Operationalization Is Part of the Claim: Operationalization-Aware Knowledge in the Era of Generated Answers
Jan-Christoph Kalo 284

GRACE-Mem: Graph Retrieval with Agentic Curation of Evidence for Long-Term Conversational Memory
Heng Wang, Xuan Ni Chen, Pei-Yun Chang, Chih-Hsi Chen, Hsin-Yu Huang, Chienyi Lin and I-Ling Chung 293

Explaining Textual Entailment with Lexical Entailments: Using LLMs to Supply Lexical Relations for Formal Proofs
Jorryt de Jong, Stefan Morača, Ettore Cesari and Lasha Abzianidze 316

LMKG: a flexible method for ontology-aware Knowledge Graph Extraction from text
Gianmarco Pappacoda and Paolo Torroni 334

Zero-Shot Relation Classification with Large Language Models: Binary Prompting and Relevance Elicitation
Sheila Alemany, Danielle Sullivan-Pao and Olga Simek 343

Distill-RE: Few-Shot Relation Extraction via Teacher LLM Reasoning Distillation
Danielle Sullivan-Pao, Sheila Alemany and Olga Simek 359

From Imitation to Reasoning: Sample-Efficient LLM Agents for Knowledge Graph Completion
Chenjun Zheng, Zhigang Wang, Wenjia Zhang, Zhe Wang and Xiaolin Yang 379

Revisiting RAG Retrievers: An Information Theoretic Benchmark
Wenqing Zheng, Dmitri Kalaev, Noah Fatsi, Daniel Barcklow, Owen Reinert, Igor Melnyk, Senthil Kumar and C. Bayan Bruss 393

Summary of the AKBC 2026 Shared Task: Language Models for Knowledge Base Construction (LM-KBC, 5th Edition)
Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang and Tuan-Phong Nguyen 409

Closed-Book Knowledge Base Construction with Calibrated Abstention and Numeric Self-Consistency
Tanvir Ahmed 417

A Forced Answer Channel for Closed-Book Knowledge Base Construction at LM-KBC 2026
Yiqun Sun, Pengfei Wei and Lawrence B. Hsieh 427

REAP: Relation-Aware Elicitation and Parsing for Closed-Book Knowledge Base Construction from LLMs
Thanh-Dan Bui, Thanh-Trung Do and Tuan-Phong Nguyen 434

SetProbe: Description-First Relation-Wise Self-Consistency for Closed-Book Knowledge Base Construction
Jannic Alexander Cutura 441

Coverage Is Not Knowledge: Why Closed-Book Knowledge Base Construction Looks Selection Bound
Maksim Silchenko 447

Knowledge Base Construction from Language Models: Mechanisms and Calibrations for Multi-Object Relational Extraction under Parameter Budgets
Chengcheng Wei, Ze Chen, Jiewen Zheng, Jiarong He and Max Gao 456

Pre-training Aware Directed Reasoning for Closed-Book Knowledge Base Construction
Mikołaj Tym 462

Exemplars Primarily Shift Retrieval, Verification Instructions Can Shift Rejection: An Anatomy of Inference-Time Interventions for Closed-Book Knowledge Base Construction
Takashi Sasaki 473

COVER-KBC: Evidence-Centric Adaptive Inference for Closed-Book Knowledge Base Construction
Quoc-Linh Vo 480

MAVE: Multi-Angle Voting for Emission in Closed-Book Knowledge Base Construction
Jonas Kunze and Andreas Both 488

MuseList: Confidence-Gated Multi-Surface Recall for Closed-Book Knowledge Base Construction
Adrian Zimmermann 496

The Instruction Tax on Factual Recall: Base Models Can Beat Their Instruction-Tuned Counterparts
Christophe Khalil and Aya Mourad 518

One Workflow Does Not Fit All Relations: A System Description for the AKBC 2026 LM-KBC Shared Task
Andreea-Elena Cochintele, Anca Dobrovăț and Sergiu Nisioi 527

CoRe: Construct-versus-Recall Routing for Knowledge Base Construction Under a Parameter Budget
Fatih Naufal Habibillah and Filbert Aurelian Tjiaranata 535

Ensembling Elicitation Strategies for Knowledge Base Construction
Vlad-Mădălin Ungureanu, Matei Neacșu and Ianis Belu 543

DataMind at AKBC Shared Task 2026: Relation-Specific Multiple Prompting for Closed-Book Knowledge Base Completion
Tanaree Patcharatrakul and Peerapon Vateekul 550

Relation-Specific Elicitation of a Base Model under a 32B Closed-Book Budget for LM-KBC 2026
Ruggero Marino Lazzaroni, Tim Booker, Pau Soler Valadés, Mathias Angermaier and Kirill Solovev 560

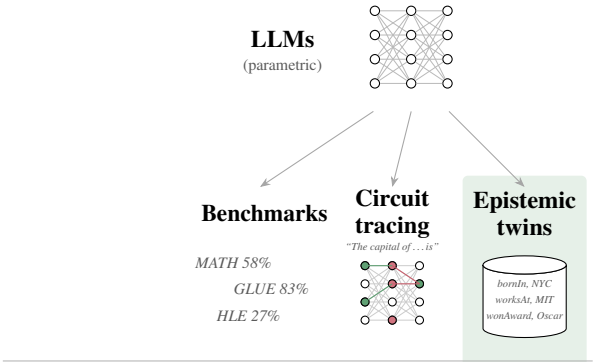
Epistemic Twins: Enabling a Symbolic Science of Language Model Knowledge [Vision Paper]

Simon Razniewski[1] **Shrestha Ghosh**[2] **Luca Giordano**[1]
Yujia Hu[1] **Josua Kowalzik**[1] **Tuan-Phong Nguyen**[3]

[1] ScaDS.AI & TU Dresden
[2] University of Tübingen
[3] VNU University of Engineering and Technology

Abstract

Large Language Models (LLMs) are impactful yet opaque artifacts. Subsymbolic by design, they are defined by billions of numeric weights that interact in a largely inscrutable manner. Current analysis paradigms are either black-box benchmarks that test model performance on pre-defined tasks, or mechanistic interpretability approaches that trace back outputs to specific weights. Both analysis methods are limited by the experimenter's hypothesis space - one must know what to look for to find it. In this article, we argue for a third, radically different analysis paradigm: **Epistemic Twins**. We propose constructing large-scale symbolic approximations of LLMs in human-readable formats. This enables the comprehensive inspection of factual knowledge (or beliefs) inherent in the model without predefining hypotheses, facilitating large-scale auditing towards better understanding and explainability.



	Benchmarks	Circuit tracing	Epistemic twins
Scaling	(✓)	✗	✓
Factual tracing	✗	✓	✓
Closed-model applicable	✓	✗	✓
Hypothesis-free	✗	✗	(✓)

Figure 1: Three paradigms for analyzing LLMs. Benchmarks operate at reasonable scale, but do not enable factual tracing nor are hypothesis-free. Circuit tracing enables tracing, but is hard to scale, and requires narrow hypotheses. We propose Epistemic Twins as novel analysis paradigm.

1 Introduction

LLMs have transformed machine intelligence, and serve as the cornerstone of the current wave of artificial intelligence. Despite their impact, our understanding of their functioning and capabilities remains vague. The public discourse is often driven by hyped marketing language, with only a weak grounding in rigorous statistical or theoretical analyses of LLM knowledge and functioning. Consequently, there is a significant barrier to LLM deployment in safety-critical applications.

Currently, two paradigms dominate the analysis of LLMs (see Fig. 1):

1. *Black-box benchmarks:* This approach ignores the internal functioning of LLMs and focuses on its input-output-behavior in test cases, often prompting it with thousands of them to assess specific capabilities (Ribeiro et al., 2020; Chang et al., 2024). While most common, this paradigm is inherently limited by the coverage of the benchmarks ("availability bias" (Tversky and Kahneman, 1973)); it cannot reveal behaviors or knowledge not anticipated by the benchmark designers (Raji et al., 2021; Bean and others, 2025).

2. *Circuit tracing (mechanistic interpretability):* This approach attempts to open the black box, tracing responses back to individual circuits and neurons to establish causal links (Rai et al., 2024). While this enables real explainability, it is computationally expensive and fragile to distribution shifts, with possibly parallel explanations for a single task, and offers little insight into the global knowledge state of the model (Méloux et al., 2025; Sharkey et al., 2025; Somvanshi et al., 2026).

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 1–7
October 28, 2026 ©2026 Association for Computational Linguistics

In light of these limitations, we advocate for a third, complementary and novel paradigm: **Epistemic Twins**. *An epistemic twin is a symbolic, comprehensive and comprehensible model of the factual knowledge stored within an LLM*, stored in a classical relational or RDF database/knowledge base.

If this epistemic twin is sufficiently faithful to its parametric LLM, i.e., approximates its knowledge well, then LLM providers and users can then answer questions about the epistemic state of the LLM by established database technology over the twin. Epistemic twins offer four opportunities:

1. **Scaling**: They can offer insights about a sample of unprecedented size, or even the entirety of an LLM's knowledge, containing millions of entities and hundreds of million of assertions, where even large benchmarks only operate at the scale of a few tens of thousands of instances.

2. **Factual tracing**: They enable to trace LLM responses to factual prompts to specific statements within (or not represented in) the epistemic twin. This way, LLM behavior can be explained by its knowledge state.

3. **No weight access needed**: Unlike mechanistic interpretability, which requires access to an LLM's weights, epistemic twins can be constructed even for API-only models (see Sec. 3).

4. **Hypothesis-free**: They are hypothesis-free human-readable representations of the LLMs knowledge, i.e., they do not underlie the cognitive availability bias (Tversky and Kahneman, 1973) that benchmarks and circuit tracing face, where the experiment designers predisposition crucially limits the possible experiment results.[1]

In this article, we present both the idea of epistemic twins, as well as a call to action. We argue that for auditability, it should become a standard that akin to other documentation in model cards (Mitchell et al., 2019), model providers release epistemic twins along with newly released LLMs. We proceed by introducing the concept of epistemic twins (Section 2), their construction (Section 3), formal approaches to estimate approximation quality (Section 4), and exemplar analyses enabled by symbolic twins (Section 5). We conclude with a perspective on cost and a call for action (Section 7).

[1] Of course, any individual analysis conducted on top of an epistemic twin still require a hypothesis, therefore, we used the symbol (✓) in Fig. 1.

2 What is an Epistemic Twin?

The central premise of our case relies on two observations.

1. **Translating subsymbolic knowledge into symbolic form:** LLM knowledge is by default subsymbolic and numeric: When one downloads a model, all one gets are billions of real-valued weights, with no apparent meaning or interpretation. To be truly auditable, this subsymbolic form must be translated into a human-readable, symbolic form.

2. **Comprehensive knowledge mapping:** Current assessment by benchmarks is sample-based, and hence strongly subject to experimenter bias. An epistemic twin aims to extract the *entirety* of the LLM's factual knowledge, thereby enabling serendipitous discoveries and statistical analyses over the whole underlying knowledge base.

A epistemic twin manifests as a database or knowledge base (KB) (Weikum et al., 2021), which provides both a human-readable data format, and a format readily accessible to statistical analysis via SQL or SPARQL queries.

The reader may wonder how epistemic twins differs from simply releasing an LLM's training corpus, as available, e.g., for the OLMo series (Groeneveld et al., 2024; Ettinger et al., 2025). We emphasize that training corpora serve the purpose of pipeline reproducibility, not explainability. Knowing that a fact existed in the training data does not guarantee the LLM learned it (Kandpal et al., 2023), nor does it reveal the hallucinations, false inferences, or conflations the model generated during training. An epistemic twin represents the model's actual final *post-training state*, not its pre-training input.

3 How to Construct Epistemic Twins

Locating and isolating knowledge in LLMs from other capabilities has been advocated by various works (Geva et al., 2021; Meng et al., 2022; Mahowald et al., 2024). Large-scale extraction

and transformation of LLM knowledge into symbolic form is an emerging field, yet promising approaches already exist. Cohen et al. (2023) introduced LMCrawl, a recursive prompt pipeline for extracting the internalized knowledge graph by a breadth-first search (BFS). This method was scaled by Hu et al. (2025) as the GPTKB methodology, to enable the concurrent extraction of millions of entities and statements. The process can be formalized as a graph traversal algorithm over the latent knowledge graph inside the LLM:

1. **Seed Initialization:** The process begins with a set of seed entities (e.g., a list of popular head entities).

2. **Recursive Expansion:** The system prompts the LLM to generate statements associated with the current entities.

3. **Traversal:** Named entities within the generated statements become new subject nodes for the next iteration.

4. **Termination:** The recursion continues until a stopping criterion is met, such as graph depth, a compute budget, or a confidence threshold (e.g., when the average confidence of newly discovered facts drops below 90%).

The exact choice of formalism for statements is important. Knowledge bases typically rely on subject-predicate-object triples, though these have limitations in terms of expressivity, which in turn affects the faithfulness of the twinning. The formalism choice requires a balance between structure and expressivity, and this is an open challenge of the epistemic twin concept.

Epistemic twins can be constructed open-domain, or restricted to a specific topic. In the open-domain, the GPTKB series presents the most mature prototypes, with *GPTKB 1* (Hu et al., 2025), *GPTKB 1.5* (Hu et al., 2026a) and *GPTKB 2.0* (Hu et al., 2026b) constructing epistemic twins for GPT-4o-mini, GPT-4.1, and GPT-5.1, respectively. These KBs reach scales of 2-6M entities and 40-100M statements each. For topic-specific extractions, Giordano and Razniewski (2026) showed how the GPTKB methodology can be restricted in scope, and constructed *babylonGPTKB*, *tBBTGP-TKB* and *dax40GPTKB*, comprising 13k, 1k and 2k statements, respectively, extracted from GPT-4.1-mini.

Similar methodologies exist, for instance, non-recursive list-based extraction schemes (Hao et al., 2023; Parović et al., 2025), taxonomic knowledge (Funk et al., 2023; Zeng et al., 2024), or a text-oriented adaptation that has already extracted 1M Wikipedia-style articles (Saeed and Razniewski, 2026). The underlying codes are publicly released, and can be deployed against public as well as private LLM endpoints, hence, we expect a proliferation of similar knowledge extractions in the near future.

4 How Can we Guarantee Twinning Faithfulness?

For epistemic twins to be useful, one must be able to assess their *twinning faithfulness*, i.e., the degree to which they truly reflect the parametric knowledge inside the modeled LLM. We propose three key dimensions of twinning faithfulness:

1. Twinning accuracy (precision): The fraction of assertions in the epistemic twin that are actually beliefs held by the twinned LLM.[2]

2. Twinning coverage (recall): The fraction of (factual) beliefs of the LLM that are contained in the epistemic twin.

3. Twin determinism (reproducibility): The degree to which twin construction is a reproducible process.

The perfect twin would be 100% accurate, cover all of the twinned LLM's knowledge, and be unique. However, as LLMs are probabilistic constructs, perfect twinning is not possible. We propose the following approaches to measure twinning faithfulness:

1. For twinning accuracy, for a given *s-p-o* triple in the twin, one can simply prompt the LLM with *s* and *p* (and some paraphrases), and observe whether *o* is indeed returned by the LLM. For example, for GPTKB 1.5, this way, we observe a twinning accuracy of 74%.

2. For twinning coverage, one can use abundance estimators from ecology on random walks

[2]Note that *Twinning accuracy* should not be mixed up with the Twin's *factual accuracy*. For example, GPTKB 1.0 is reported to have a factual accuracy of 31%, i.e., 31% of its statements are true w.r.t. the real-world, but its twinning accuracy relative to GPT-4o-mini is far higher, at around 60%. The converse, i.e., a high factual accuracy, but low Twinning accuracy, is also possible.

Analysis	Method	Example from GPT-4.1 twin *(GPTKB 1.5)*
Topical coverage	Count relevant entities	GPT-4.1 knows more than **1M persons** and more than **100k companies**.
Gender bias	Count gender identity facts	GPT-4.1 has a positive bias towards females (**1.16:1**); however, this strongly flips outside English-language nationalities.
Sensitive content	Search for sensitive attributes	GPT-4.1 stores virtually no private phone numbers and addresses, but **thousands of statements** about ethnic, religious, and sexual identities.
Logical consistency	Check satisfaction of logical rules	GPT-4.1's knowledge is highly inconsistent: only **15%** of border-relation facts are present in both directions.
Knowledge cutoff	Observe latest dates	GPT-4.1's knowledge cutoff is **mid-2024**.

Table 1: Examples of analyses enabled by epistemic twins. Taken from (Ghosh et al., 2025).

over the model's knowledge graph (Luggen et al., 2019), or BFS property extrapolation (Kurant et al., 2011), to predict the fraction of all factual knowledge covered within the epistemic twin. For example, for GPTKB 1.5, by BFS run extrapolation, we estimate a coverage of 42% of all named entities within GPT-4.1.

3. For twin determinism, one can use repeated runs of the twin construction process with different seed nodes or random seeds, and observe the overlap in the resulting twins. For example, based on repeated runs of the construction process for babylonGPTKB, it was observed that repeated runs overlap in roughly two thirds of their named entities, even from different starting entities (Giordano and Razniewski, 2026). This indicates that despite some variability, beliefs remains largely consistent across extraction runs.

Of course, the faithfulness dimensions and metrics proposed above are only a start. Future research will extend and refine the methods for defining and measuring twinning faithfulness.

5 What Analyses Are Enabled by Epistemic Twins?

The existence of multiple and faithful epistemic twins, as imagined in Figure 2 can transform LLM research and certification. It enables:

1. **Quantification of Knowledge:** We can complement estimates via parameter counts (Allen-Zhu and Li, 2024) with estimates based on counting actual "knowledge units" (facts). This allows us to compare models based on information density. For example, based on BFS extrapolation, the GPT-4.1 epistemic twin GPTKB 1.5 can be used to predict that GPT-4.1 knows a total of around 15 M named entities.

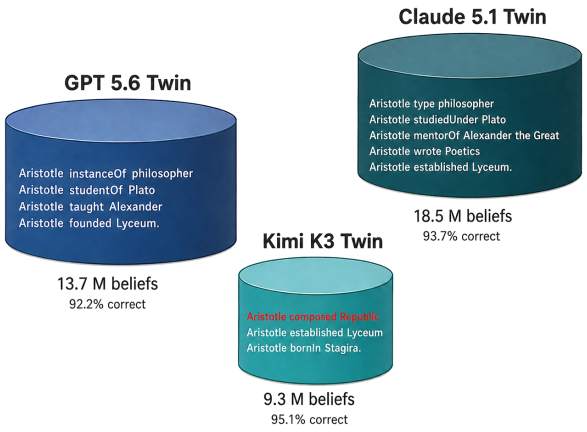


Figure 2: The envisioned twins enable the quantitative and qualitative comparisons of LLM's factual knowledge.

2. **Global Accuracy Estimation:** By evaluating the epistemic twin against reference sources, we can calculate global accuracy scores that are not overfitted to specific benchmark test sets. For example, (Hu et al., 2026a) observe a global accuracy of 76% of the GPT-4.1 twin GPTKB 1.5, well below the numbers reported in typical factuality benchmarks.

3. **Bias and Topic Auditing:** An epistemic twin allows for topic modeling over the model's beliefs. We can visualize the distribution of knowledge (e.g., Western vs. Non-Western history) and detect systemic biases (e.g., gender associations in professional roles) by querying the database rather than probing the model stochastically. More specific examples are given in Table 1.

Such analyses can benefit three types of stakeholders. For **users**, this aids in model selection. For

providers, it offers a transparency mechanism for accountability. For **regulators**, it provides a tangible artifact to evaluate compliance without needing access to weights or training data.

6 Challenges and Research Opportunities

Realizing epistemic twins as a practical auditing instrument raises several open challenges:

1. **What are we twinning?** Whether LLMs can be described as having beliefs remains contested (Bender et al., 2021; Herrmann and Levinstein, 2025). Recent work argues that belief-like representations can be studied under explicit adequacy criteria (Herrmann and Levinstein, 2025), and mechanistic evidence supports the existence of belief-like internal states (Mendozza and Søgaard, 2026). Still, the question remains unsettled. Pragmatic evidence suggests that factual knowledge may be encoded in task-specific ways (Elhelo et al., 2026), or that models can internally encode information that is not faithfully reflected in their surface outputs (Orgad et al., 2025; Gekhman et al., 2025). The concept of twinned knowledge therefore needs further refinement.

2. **Representation and faithfulness.** LLMs possess many capabilities beyond factual knowledge, e.g., translation, planning, or creative generation. Our focus is factual knowledge, but even this narrower target inherits the classical difficulties of knowledge representation: all models are wrong (lossy), and a careful balancing between expressivity and structure needs to guide the actual representation.

3. **Cost and scale.** Constructing epistemic twins is resource-intensive. For example, GPTKB 1.5, a 100-million-triple prototype derived from GPT-4.1, cost approximately $14,000 in API calls (Hu et al., 2026a). Higher-fidelity twins may plausibly cost hundreds of thousands of dollars, especially if they require repeated runs or robustness analyses.

7 Call for Action

The challenges are substantial, and the costs are nontrivial. Yet they still pale relative to the costs spent for training frontier models. The Stanford AI Index estimates the compute cost of training GPT-4 at $78 million and Gemini Ultra at $191 million (AI Index Steering Committee, 2024); a $100,000 epistemic twin would therefore correspond to just 0.1% of the training costs. Furthermore, the investment can start small. Topic-specific twins can be constructed at single-digit-dollar budgets (Giordano and Razniewski, 2026).

Epistemic twins offer a complementary path toward model accountability. Benchmarks measure performance on predefined tasks, and mechanistic interpretability studies internal computations, but neither by itself provides a broad, queryable, model-specific map of factual knowledge. Epistemic twins could fill this gap by making model knowledge inspectable, comparable, and auditable at scale. In this sense, they extend the logic of model cards (Mitchell et al., 2019): not only reporting how a model performs, but exposing what it appears to know under a transparent extraction protocol.

We therefore call for a community effort around epistemic twins: more and better construction methods, more validation of faithfulness, and public twin artifacts for important model families. If we want to understand model knowledge at scale and with less experimenter bias than fixed-question benchmarks allow, epistemic twins are a timely and concrete next step.

Limitations

Factual knowledge focus Our concept of epistemic twins currently focuses strictly on declarative, factual knowledge. LLMs also possess other knowledge, e.g., procedural knowledge on how to code or multiply, or linguistic knowledge on how to rhyme or argue. These non-factual capabilities are not currently captured by the fact-oriented knowledge graph structure proposed here.

Probabilistic-to-boolean LLMs are inherently probabilistic, while knowledge bases by default represent knowledge in Boolean form (true/false), and the epistemic twins constructed so far follow this representation. A translation from probabilities to a Boolean format necessarily leads to some information loss. It is in principle possible to expand representation to a probabilistic format, though probabilistic databases are much less established, and the translation of LLM-internal token likelihoods into truth probabilities is a nontrivial task as well (Jiang et al., 2020).

References

AI Index Steering Committee. 2024. AI index report 2024. Technical report, Stanford Institute for Human-Centered Artificial Intelligence (HAI).

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv*.

Andrew M. Bean and others. 2025. Measuring what matters: Construct validity in large language model benchmarks. In *neurIPS*.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *FAT*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and others. 2024. A survey on evaluation of large language models. *Transactions on Intelligent Systems and Technology*.

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling the internal knowledge-base of language models. In *Findings of EACL*.

Amit Elhelo, Amir Globerson, and Mor Geva. 2026. LMs as task-specific knowledge bases: An interpretability analysis.

Ettinger et al. 2025. Olmo 3. *arXiv*.

Maurice Funk, Sebastian Hosemann, Jean Christoph Jung, and Carsten Lutz. 2023. Towards ontology construction with language models. In *KBC-LM*. Knowledge Base Construction from Pre-trained Language Models (KBC-LM).

Zorik Gekhman, Eyal Ben David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. 2025. Inside-out: Hidden factual knowledge in LLMs.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *EMNLP*.

Shrestha Ghosh, Luca Giordano, Yujia Hu, Tuan-Phong Nguyen, and Simon Razniewski. 2025. Mining the mind: What 100m beliefs reveal about frontier LLM knowledge. *AKBC@EMNLP*.

Luca Giordano and Simon Razniewski. 2026. Foundations of LLM knowledge materialization: Termination, reproducibility, robustness. In *EACL Findings*.

Groeneveld et al. 2024. OLMo: Accelerating the science of language models. In *ACL*.

Shibo Hao, Bowen Tan, Kaiwen Tang, Bin Ni, Xiyan Shao, Hengzhe Zhang, Eric P. Xing, and Zhiting Hu. 2023. BertNet: Harvesting knowledge graphs with arbitrary relations from pretrained language models. In *Findings of ACL*.

Daniel A. Herrmann and Benjamin A. Levinstein. 2025. Standards for belief representations in llms. *Minds and Machines*.

Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, Moritz Müller, and Simon Razniewski. 2026a. GPTKB v1.5: A massive knowledge base for exploring factual LLM knowledge. In *AAAI*.

Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, and Simon Razniewski. 2025. Enabling LLM knowledge analysis via extensive materialization. In *ACL*, pages 16189–16202.

Yujia Hu, Tuan-Phong Nguyen, and Simon Razniewski. 2026b. GPTKB 2.0: Browsing, querying, and auditing a disambiguated LLM-derived knowledge base. In *EMNLP*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *TACL*, 8.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *ICML*.

Maciej Kurant, Athina Markopoulou, and Patrick Thiran. 2011. Towards unbiased BFS sampling. *IEEE Journal on Selected Areas in Communications*, 29(9).

Michael Luggen, Djellel Difallah, Cristina Sarasua, Gianluca Demartini, and Philippe Cudré-Mauroux. 2019. Non-parametric class completeness estimators for collaborative knowledge graphs—the case of Wikidata. In *ISWC*.

Kyle Mahowald, Anna A. Ivanova, Idan A. Blank, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. 2024. Dissociating language and thought in large language models. *Trends in Cognitive Sciences*.

Maxime Méloux, Silviu Maniu, François Portet, and Maxime Peyrard. 2025. Everything, everywhere, all at once: Is mechanistic interpretability identifiable? In *ICLR*.

Alessandro Mendozza and Anders Søgaard. 2026. LLM beliefs are in their heads. In *ACL*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *neurIPS*.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *FAT*.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2025. LLMs know more than they show: On the intrinsic representation of LLM hallucinations.

Marinela Parović, Ze Li, and Jinhua Du. 2025. Generating domain-specific knowledge graphs from large language models. In *Findings of ACL*.

Dinesh Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Zijun Yao. 2024. A practical review of mechanistic interpretability for transformer-based language models. ArXiv.

Inioluwa Deborah Raji, Emily Denton, Emily M. Bender, Alex Hanna, and Amandalynne Paullada. 2021. AI and the everything in the whole wide world benchmark. In *neurIPS*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *ACL*, pages 4902–4912.

Muhammed Saeed and Simon Razniewski. 2026. LLM-pedia: A transparent framework to materialize an LLM's encyclopedic knowledge at scale. *EMNLP*.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeffrey Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Isaac Bloom, Stella Biderman, Adrià Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Mary Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, William Saunders, and 10 others. 2025. Open problems in mechanistic interpretability. *Transactions on Machine Learning Research*.

Shriyank Somvanshi, Md Monzurul Islam, Amir Rafe, Anannya Ghosh Tusti, Arka Chakraborty, Anika Baitullah, Tausif Islam Chowdhury, Nawaf Alnawmasi, Anandi Dutta, and Subasish Das. 2026. Bridging the black box: A survey on mechanistic interpretability in ai. *ACM Comput. Surv.*, 58(8).

Amos Tversky and Daniel Kahneman. 1973. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5(2).

Gerhard Weikum, Xin Luna Dong, Simon Razniewski, and Fabian Suchanek. 2021. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends in Databases*, 10.

Qingkai Zeng, Yuyang Bai, Zhaoxuan Tan, Shangbin Feng, Zhenwen Liang, Zhihan Zhang, and Meng Jiang. 2024. Chain-of-layer: Iteratively prompting large language models for taxonomy induction from limited examples. In *CIKM*.

Auditing a KB Elicitation of Frontier LLM Knowledge: A Multi-dimensional Analysis of GPTKB v1.5

Shrestha Ghosh[1], **Luca Giordano**[2], **Yujia Hu**[2], **Tuan-Phong Nguyen**[3], **Simon Razniewski**[2]

[1] University of Tübingen [2] ScaDS.AI & TU Dresden
[3] VNU University of Engineering and Technology

shrestha.ghosh@uni-tuebingen.de tuanphong@vnu.edu.vn

{simon.razniewski,yujia.hu,luca.giordano}@tu-dresden.de

Abstract

LLMs are remarkable artifacts that have revolutionized a range of knowledge-intensive tasks. A significant contributor is their factual knowledge, which, to date, remains poorly understood, and is usually analyzed from biased samples. In this paper, we provide a framework and the results of a multi-dimensional analysis of GPTKB v1.5 (Hu et al., 2025a), a recursively elicited Knowledge Base (KB) of 100 million facts (or beliefs) of a frontier LLM, namely, GPT-4.1. Given the scale of the elicited facts, we provide a multi-dimensional approach to qualitatively and quantitatively analyze these facts as opposed to the mainstream fact completion benchmarks, which are prone to availability bias. We find that the models' factual knowledge differs quite significantly from established knowledge bases, and that its accuracy is significantly lower than indicated by previous benchmarks. We also find that inconsistency, ambiguity and hallucinations are major issues, shedding light on future research opportunities in neuro-symbolic AI concerning extraction, consolidation and verification of factual LLM knowledge.

Direct LLM querying	**LLM-mined knowledge analysis**
✗ Lacks aggregate access to internalized knowledge ✗ Gives uninformative and wrong answers	✓ Accesses millions of extracted records ✓ Provides quantitative insights
ChatGPT interface	**Mined knowledge (GPTKB v1.5)**
How many individual persons do you know?	
I don't "know" any individual persons [..]. [..] millions to billions, depending on [..]. →uninformative	1,200,692
Do you know more males or more females?	
I don't have a built-in count of individuals by gender. However, in the publicly available sources I draw from [..]. So [..] I would "know" more males than females. →false	1 : 1.16
Do you know more about Chinese, or Canadian entities?	
I likely "know" more about Chinese entities than Canadian ones, simply because China [..] more [..] documented. →false	1 : 3.29

Table 1: Direct LLM querying, vs. large-mined knowledge analysis. ChatGPT conversation archived at https://chatgpt.com/share/68b6cd03-85e4-800d-81c8-afebc74ccc9d.

1 Introduction

Motivation Large Language Models (LLMs) are remarkable artifacts that have demonstrated impressive performance across a range of tasks such as coding, math (Claude 3.5 Sonnet 92% in HumanEval and 91.6% in MGSM (Anthropic, 2024)), or general language understanding (GPT-4.1 90.2% on MMLU (OpenAI, 2025)). A major driver of their success is their internalized knowledge, however, little is understood about what knowledge they have actually internalized, what the accuracy of their knowledge is, and what biases this exhibits.

It may be tempting to prompt the models directly, however, as shown in Table 1 (bottom left), such questions are usually answered in an unhelpful or incorrect way. Recently, the construction of epistemic twins from LLMs has been proposed as a way to comprehend LLM knowledge (Nguyen and Razniewski, 2022; Cohen et al., 2023; Hu et al., 2025b; Razniewski et al., 2026). Unlike sample-based studies (Petroni et al., 2019; Veseli et al., 2023; Sun et al., 2024; Amirizaniani et al., 2024), epistemic twins provide a view beyond reconstructing existing data via persistent knowledge base (KB) resource for repeated investigation (Hu et al., 2025b).

We answer four research questions to determine whether such a massive-scaled LLM-mined KBs serve as reliable and under what conditions. **RQ1.** How does the scale and coverage of LLM-mined knowledge compare to Wikimedia-related KBs? **RQ2.** What systematic biases does LLM-mined knowledge exhibit relative to real-world distributions? **RQ3.** How does factual accuracy change with long-tailed entities? **RQ4.** How consistent and temporally coherent is the mined KB?

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 8–18
October 28, 2026 ©2026 Association for Computational Linguistics

Contribution We provide a framework and the results of a multi-dimensional analysis of the knowledge of a frontier LLM, GPT-4.1, as surfaced via recursive elicitation and consolidation into GPTKB v1.5 (Hu et al., 2025a). We find:

1. Frontier LLMs can contain a vast amount of factual knowledge, with quite different focus and biases than existing KBs.

2. At overall 75% accuracy, the accuracy of GPT's knowledge is much higher than previous text-extraction-based KBs, but also significantly lower than human-curated resources and results on popular LLM benchmarks.

3. Inconsistency, ambiguity, and hallucinations are major issues, shedding light on the challenges and future research opportunities of extraction, consolidation and verification of factual LLM knowledge.

Our analysis specifically uses GPTKB v1.5, an epistemic twin constructed for over $14,000 from GPT-4.1, containing over 100M factual assertions for 6M entities, and there is no comparable project based on other LLMs available at the moment. While parts of our results are therefore specific to this LLM version, GPT-4.1 is an archetype of commercial frontier LLMs, and our analysis methodology generalizes beyond this model. Moreover, our way of analyzing closed-source model is especially curious as it provides an exemplary case study on how knowledge materialization can be used to reverse-engineer internals of closed-source models to reveal the knowledge/beliefs held by a frontier LLM (Table 1 right). We use the dataset and code provided on the project page.[1] The code for our multi-dimensional analysis is available at `https://github.com/Knowledge-aware-AI/gptkb-analysis`.

2 Background

Knowledge Probing of LLMs Petroni et al. (2019) had the first intuition to consider LMs as knowledge bases and introduced the LAMA benchmark, which consists of cloze-style statements derived from existing *(s, p, o)* triples and question–answer datasets. This was followed by multiple similar efforts to test LLM factuality on multilingual data (Kassner et al., 2021), head-to-tail popularity (Sun et al., 2024), unbiased Wikidata samples (Veseli et al., 2023) and unseen events with known entities (Zheng et al., 2024). However, these works on benchmark-based "knowledge probing", despite attempts to mitigate the issue with automatic prompt optimization (e.g. Jiang et al. (2020); Shin et al. (2020); Wu et al. (2025)) as pointed out by Hu et al. (2025b), suffer from availability bias (Tversky and Kahneman, 1973), i.e. they cannot reveal knowledge outside the benchmark's scope.

LLM Knowledge Extraction and Epistemic Twins Cohen et al. (2023) propose LMCRAWL that extracts a knowledge graph of facts from a language model. Given a seed entity, first its paraphrases are generated, then all possible relations and their respective paraphrases, and finally objects for all relations. Parović et al. (2025) propose the CoVe method to extract domain-specific KGs from LLMs with an LLM-generated schema and verification prompts. Both these methods are limited in their scope of knowledge extracted from LLMs. LMCRAWL generates less than 100 triples per seed entity in two hops and the CoVe KGs on books and landmarks contain tens of thousands of entities and hundreds of thousands of triples per domain. The idea has been lifted into the epistemic twin paradigm: The construction of comprehensive approximations of an LLM's knowledge in symbolic form (Razniewski et al., 2026).

GPTKB v1.5 The GPTKB methodology (Hu et al., 2025b) combines a recursive knowledge elicitation process with a post-hoc knowledge consolidation phase. Starting from a seed subject, the LLM is prompted to return knowledge about it in the form of triples. New named entities in these triple objects are identified via LLM-based named-entity recognition (NER) for further elicitation in a recursive BFS-based graph exploration process. To address the redundancy and variance introduced during knowledge elicitation, a greedy clustering algorithm is used to iteratively merge relations and classes into more frequent ones, given a sufficiently high label embedding similarity. This approach at "knowledge crawling" mitigates the availability bias typical of probing-based benchmarks and enables knowledge analysis more flexible than memorization quantification approaches, which depend on access to training data and the LLM's ability to generate exact substrings of the training data (Carlini et al., 2023a,b).

[1] `https://gptkb.org`

Resource	#entities	#assertions
Wikimedia-related		
Wikidata (Vrandecic and Krötzsch, 2014)	113M	1.62B
Yago 4.5 (Suchanek et al., 2024)	50M	140M
DBpedia (Auer et al., 2007)	3.8M	75M
Text-extracted		
NELL (Carlson et al., 2010)	1.9M	12M
ReVerb (Fader et al., 2011)	2.3M	15M
Generative		
LMCRAWL (Cohen et al., 2023)	?	69.8 ± 52.9
CoVe (Parović et al., 2025)	22K-65K	53K-153K
GPTKB v1.5 (Hu et al., 2025a)	6.1M	100M
LLM Factuality Benchmarks		
Head-to-Tail (Sun et al., 2024)	?	18K
LAMA Probe (Petroni et al., 2019)	?	50K
WD-KNOWN (Veseli et al., 2023)	2.9M	3.9M

Table 2: Size comparison of knowledge resources and LLM factuality benchmarks.

Class	Frequency	Relation	Frequency
person	1,061,852	instanceOf	6,588,295
human	138,840	locatedIn	2,121,234
film	123,021	occupation	1,962,461
company	122,847	country	1,927,856
book	112,188	notableWork	1,500,584
song	103,824	genre	1,268,067
fictional_character	93,779	nationality	1,236,491
album	81,500	language	1,224,199
city	80,861	location	1,072,795
village	78,753	notableFor	977,561

Table 3: Most frequent GPTKB classes and relations.

Domain	Fraction	Domain	Fraction
Science	0.2%	Business	7.0%
Technology	4.0%	Health	0.0%
Engineering	1.3%	Environment	1.9%
Mathematics	0.0%	event	0.8%
Arts	23.8%	unknown	0.0%
Humanities	60.0%	food	1.1%
Social Sciences	0.2%	right	0.0%

Table 4: Entity proportion of first-order subbranches of the taxonomy, based on a random sample of 1000 entities.

3 Content Scale and Coverage

In this section we address *RQ1. "How does the scale and coverage of LLM-mined knowledge compare to Wikimedia-related KBs"*?

Size The very first observation is that there is a lot of knowledge in GPT, orders of magnitude more than tested for in any previous benchmark. Table 2 relates the size of GPTKB v1.5 to other prominent resources and benchmarks. We find that the size is on par with major existing KBs like DBpedia and Yago, as well as major text extraction projects. Only Wikidata is still an order of magnitude larger, although its biases are well documented, e.g., an undue weight on scholarly information (45.5M scholarly article entities vs. 77.3M other entities as of July, 2026) (Wikidata team, 2025).

Topical Analysis The most frequent classes and relations in GPTKB are shown in Table 3. *person* and *human* dominate the KB at 1.2M making up 19% of all entities, followed by *film* and *company* at just 4%. This lower-bound estimate of the number of persons the model has acquired data on is especially remarkable, as OpenAI is notoriously secretive about the amount of data seen or learned by their models, with their models consistently denying storing any data about individuals (see Tab. 1). [2] The remaining classes concern media artifacts (film, book, song, album), geopolitical entities (city, village), organizations (company). This is in stark contrast to Wikidata, where the dominating class, scholarly articles, makes up 38% of the KB, followed by humans (10%) and astronomical objects (7%). The mandatory *instanceOf* relation aside, the relation frequency show a more even distribution. In contrast to Wikidata, GPTKB puts much more emphasis on so-called *soft/subjective* relations that narrate achievements, like *notableWork, notableFor*, which are among the most frequent.

Taxonomy The seed taxonomy generated by the LLM, without priors from the resource KB, exhibited an architectural bias towards STEM. It established *Engineering*, *Mathematics*, *Technology* and *Science* as peer categories to *Humanities* and *Arts*. In stark contrast, when this STEM-centric taxonomy was populated, the taxonomy is overwhelmingly concentrated in the Humanities and Arts branches (60.0% and 23.8%), while Technology (4.0%), Engineering (1.3%) and Science (0.2%) contribute only a very small proportion (see Table 4). This uncovers a critical decoupling between the LLM's organizational logic and its knowledge content and demonstrates that the LLM's worldview is twofold.

Language We performed language detection on the triples using fastText (Joulin et al., 2017) to analyze the linguistic composition of GPTKB. Figure 1 shows the distribution of the six most frequent lan-

[2] Note that we do not claim violations of privacy here, as the data may be voluntarily published, and sourced from public resources like Wikidata. More on sensitive data below.

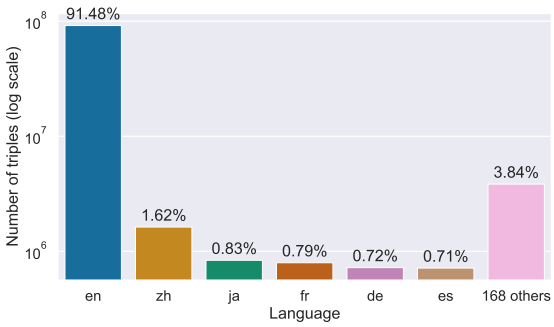


Figure 1: Language distribution in GPTKB.

guages, with the remaining languages aggregated. Given that GPTKB was constructed using English-based prompts, the fact that English dominates at 91.5% of the content is unsurprising. Crucially, in our analysis of the remaining 9.5% of the detected content, we found that non-English knowledge is largely self-contained. Only 22.6% of all knowledge paths that start from an English entity and move to a non-English one return to an English-language statement. Particularly, Japanese, Chinese and Hebrew entities return to English in less than 10% of the cases. This evidence strongly indicates that much of the LLM's non-English knowledge is organized into its own densely interconnected, language-specific clusters. The long tail of 168 languages collectively accounts for 3.84% of the KB's content. Such a substantial and diverse linguistic footprint, elicited from English-centric queries, suggests that the internal knowledge of the LLM contains a rich repository of multilingual content that remains largely unexplored.

Property	Count	Property	Count
logo	37,338	mapType	2,432
coverArt	33,717	photo	2,253
signature	32,443	portrait	1,337
coatOfArms	5,833	range	581
flag	3,424	hasPoster	517

Table 5: Top-10 predicates which come with objects being image file names.

Literals The are 42.9M triples with literal objects in GPTKB, making up 43% of all triples. *website* and *occupation* are the most frequent predicates with literals objects (frequency > 800K). This is followed by predicates denoting language *language*, notability *notableFor* and *knownFor*, and date-related predicates such as *birthDate* and *deathDate*. Interestingly, we also found image filenames as literal objects occurring most frequently with predicates, such as, *logo*, *coverArt* and *signature* (Table 5). This indicates that the LLM has memorized some image filenames, likely from resources such as Wikipedia or Wikidata.

Dimension	RegEx Heuristic	Predicates	# Unique Subjects
Gender	*gender*	gender	270,694
Race	*race*	race	395
Ethnicity	*ethnicity*	ethnicity	49,799
Religion	*religion*	religion	81,675
Disability	*disability*	disability	1,273
Sexual orient.	*sexualorient*	sexualOrientation	4,594
Contact details	*address*	-	0
	phone	phoneNumber	2
	email	hasEmail	1
Finan. info.	*income*	-	0
	networth	netWorth / netWorthAtDeath	2,253
Military info.	*coordinates*	coordinates	55,265
	planneddeploy	plannedDeploymentSite / plannedDeploymentYear / plannedDeploymentRegion	3
	baseloc	hasBaseLocation / majorUSBaseLocation / militaryBaseLocation	5

Table 6: Predicates related to potential sensitive information found in GPTKB and count of unique subjects. We filter for instanceOf: human/person for all the predicates except those related to military info.

Sensitive Data We heuristically identified predicates in GPTKB on eight potentially sensitive data dimensions, namely, gender, race/ethnicity, religion, disability, sexual orientation, contact details, financial and military information. Table 6 lists the identified predicates and the count of subjects associated with them. Notably, personal identifiers appear only for public entities (e.g., address and phone number for organizations, net worth for celebrities). Entities who are children. i.e. who have a birth date less than ten years from the time of writing (August 2025) also do not raise any privacy concerns on any of our predefined dimensions. Inspired by Dodge et al. (2021), which documented an unexpected amount of text from US military websites in the C4 corpus, in GPTKB we observe few military-related predicates, which contain only public domain information.

Gender, race/ethnicity, religion, disability, and sexual orientation information exist mainly for individuals where these dimensions are contextually relevant for their notability. For example, the most frequent race/ethnicity, "African-American", occurred almost exclusively for notable activists, jazz musicians, victims of hate crimes, and politicians from the Reconstruction era and the Civil Rights

Movement. The same finding is valid for religion (e.g. notable bishops, rabbis, monks, priests, and other religious leaders), disability (e.g. Paralympic athletes), and sexual orientation (e.g. activists and celebrities supporting LGBTQ rights).

Result 1: Content

GPTKB departs from Wikidata through diverse classes and subjective relations. The duality of the broad taxonomy and concentrated class population exposes the twofold worldview of GPT-4.1. Non-English content is organized in densely-connected, language-specific clusters.

4 Biases

In this section we address *RQ2. "What systematic biases does LLM-mined knowledge exhibit relative to real-world distributions?"*

Gender Bias *female* and *male* are the dominant gender values in GPTKB, occurring in 196K and 168K triples respectively, with few other values, such as, *non-binary* (589) and *transgender woman* (413). Notably, GPTKB has a female-to-male gender ratio of 1.16, i.e., *female* is more frequent. The classes with the highest gender statements are *person*, *fictional_character* and *human*, with a gender ratio of 1.31, 0.68, and 0.88, respectively. For comparison, in the real-world, this gender ratio, hovers close to but slightly below one (Ritchie and Roser, 2019), and in Wikidata it is about 0.2 (Yunus et al., 2025). It is notable that the gender is mentioned in just about 22% of the entities from the classes *person* and *human*, reinforcing our previous observation about gender, among other personal identifiers, being mentioned only when contextually relevant.

No female gender		No male gender	
Profession	**Frequency**	**Profession**	**Frequency**
sports shooter	127	gptkb:actress	11248
cowboy	74	gptkb:businesswoman	1307
gptkb:drag_queen	70	noblewoman	581
crime lord	68	gptkb:suffragist	469
gptkb:hurler	60	gptkb:feminist	323
gentleman	55	housewife	269
gptkb:cryptocurrency	48	gptkb:beauty_pageant	226
drug dealer	42	waitress	165
carpenter	42	homemaker	164
gptkb:casino	40	gptkb:softball	153

Table 7: Top 10 gender-specific professions.

This active gender debiasing is prominent among professions accessible via the *occupation* property. We find professions heavily skews towards females, for instance, *educator* and *activist*, with even gender-specific professions, such as, *actress* (only females) (Figure 3). In particular, GPTKB has 189 professions with no female entities and 203 professions with no male ones (top-10 list in Table 7). Gender-neutral professions occurring at lower frequencies exhibit skews reflecting traditional gender norms, such as *nurse* with a gender ratio of 39.5 (1,541 triples) and *software_engineer* with a gender ratio of 0.38 (696 triples).

Nationality	**GPTKB**	**Global***	**Global top-10**
American[a]	31.5	4.3	India (18.2)
British[b]	12.4	0.8	China (17.6)
French	3.9	0.8	United States (4.3)
German	3.6	1.0	Indonesia (3.5)
Indian	3.6	18.2	Pakistan (3.9)
Canadian	3.3	0.5	Nigeria (2.9)
Australian	2.5	0.3	Brazil (2.6)
Japanese	2.4	1.5	Bangladesh (2.1)
Italian	2.3	0.7	Russia (1.8)
Russian	1.4	1.8	Ethiopia (1.6)

[a] *includes United_States* [b] *includes English*

Table 8: Nationality distribution in percentage in GPTKB and the real-world. Nationalities in GPTKB that feature in the global top-10 are highlighted. *Real world data obtained from https://worldpopulationreview.com/countries as of 2025.

Nationality	**Top-5 Professions**
American	author, politician, actor, actress, musician
British	footballer, cricketer, politician, priest, rugby_union_player
French	politician, writer, actor, actress, footballer
German	footballer, politician, professor, writer, composer
Indian	politician, actor, actress, film_director, writer
Canadian	politician, ice hockey player, actor, actress, musician
Australian	politician, cricketer, actress, actor, footballer
Japanese	footballer, voice_actor, actor, singer, actress
Italian	footballer, politician, painter, actor, composer
Russian	footballer, writer, politician, actor, military_officer

Table 9: Top professions of most frequent nationalities.

Geographic Bias Table 8 shows the most frequent nationality distribution in GPTKB. While GPTKB nationalities do not exclusively belong to living humans, we still compare the percentage share with the global population distribution to obtain an estimate of the order of the distribution. The bias to English-speaking nations, making up almost 44% of all nationalities, is reflective of the training corpus and not alarming. However, in light of the somewhat even gender distribution observed previously, only three countries of the top ten forming global population come up in GPTKB's top ten nationalities. If we exclude American, their share, i.e.,

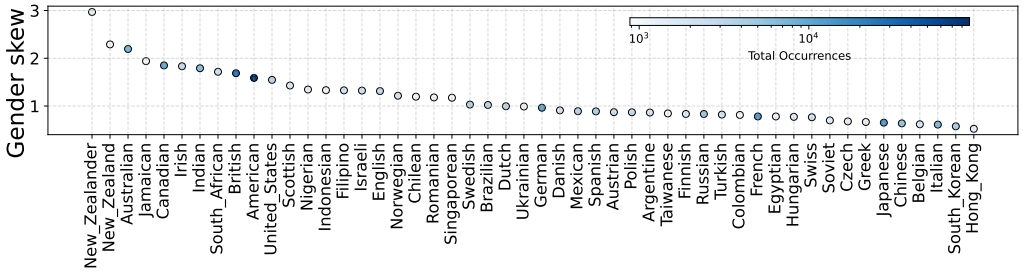


Figure 2: Female-to-male ratio (y-axis) in 50 most frequent nationalities (x-axis).

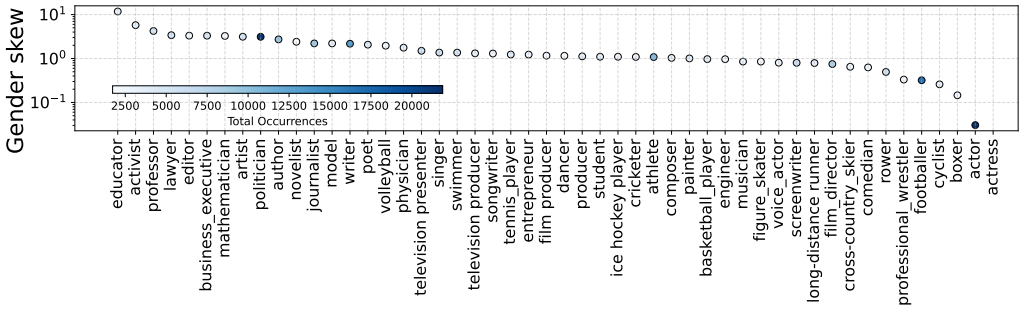


Figure 3: Female-to-male ratio (y-axis) in 50 most frequent professions (x-axis).

Indian and Russian is only 5%, while Chinese form only 1%. We find this over-representation of the United States in other geographic distributions as well. 32% of the 63K cities in GPTKB are located in the United States, followed by Japan (5.0%), India (3.2%), China (2.7%) and Russia (2.3%). 22% of the 73K companies are located in the United States, followed by the United Kingdom (10%), India (7%), Japan (5%) and China (4.5%).

Composite Biases In Table 9, we look at the most popular occupations by nationality. Politicians and footballers are by far the most frequent professions across nationalities. When we qualify by gender, politicians have a positive gender skew towards females with female-to-male ratio of 3.12 compared to actors (0.03) and footballers (0.31) (Figure 3). Figure 2 shows the gender skew in the 50 most frequent nationalities. 43% of the nationalities with a female-to-male ratio >1 belong to English-speaking countries, suggesting selective gender debiasing, as seen for professions.

Political Bias We find no strong evidence of a consistent left- or right-wing bias in GPTKB's factual coverage: U.S. Democrats have more recorded membership triples than that of Republicans (11,781 vs. 9,452), but after normalizing by the number of politician entities, positive facts occur at nearly identical rates (0.359 vs. 0.361 triples per member), while negative facts are only slightly more frequent for Republicans (0.0056 vs. 0.0042 explicit negative triples per member), based on a dictionary of positive (e.g., award) and negative (e.g., convictedOf) relations. For Germany, the corresponding left/right blocs show near parity in positive coverage (0.348 vs. 0.330 triples per member) and virtually no recorded negative facts on either side.

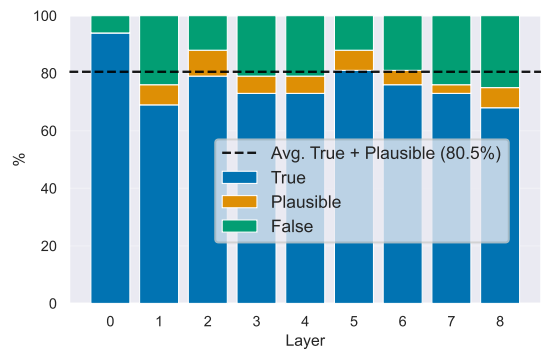
Result 2: Biases

GPTKB exposes a strong English country bias and selective gender debiasing via more female gender assertions and professions in GPT-4.1. The bias flips for non-English speaking countries and gender-neutral professions.

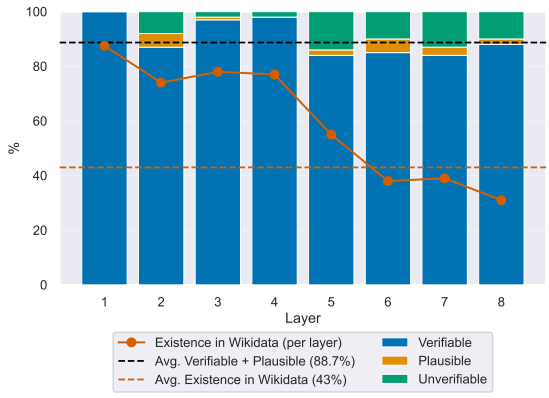
5 Accuracy and Hallucinations

In this section we examine *RQ3. "How does factual accuracy change with long-tailed entities?"* We further sample unverifiable triples to detect hallucinations versus long-tailed entities.

Accuracy GPTKB explores an LLM in a breadth-first manner, such that we get a set of new entities from parent entities of the previous exploration layer. We sampled and validated 100 triples and 100 entities from each layer in the source KB (all



(a) Trend of **triple accuracy** across layers.



(b) Trend of **entity verifiability** across layers.

Figure 4: GPTKB accuracy across layers.

triples and entity from seed layer). Following the paradigm in Hu et al. (2025b), we use Llama3.1-70B-Instruct as an judge to determine triple accuracy as true/plausible/false when provided with the top-5 webpage snippets for the search term "⟨subject⟩ ⟨object⟩" from a search engine API. Similarly, the judge model determines whether an entity is verifiable/plausible/unverifiable when provided with the top-5 search results for the subject entity. The results are presented in Figure 4.

For factual knowledge particularly relevant is the MMLU benchmark, widely used to test broad general knowledge and reasoning across academic disciplines, where GPT-4.1 is reported to achieve 90.2% (OpenAI, 2025). This starkly contrasts with the factual accuracy of only 75% and 80.5% when considering both true and plausible. Our analysis reveals a notable decoupling between entities' presence in the public domain and LLM's factual accuracy regarding it. Specifically, we observed that as the GPTKB exploration deepened, the percentage of entities verifiable in Wikidata showed a clear downward trend with an average per layer existence in Wikidata at 43%, while the accuracy of the LLM's elicited statements remained consistent, showing no significant degradation. This strongly indicates that the LLM possesses an internally consistent knowledge base that extends well into the long tail.

Domain	True	Plausible	False
Locations	75.7%	6.8%	17.5%
Persons	74.2%	6.5%	19.3%
Politics	72.7%	0%	27.3%
Organizations	72.2%	7.8%	20.0%
Science	70.6%	5.6%	23.8%
Business	68.9%	8.1%	23.0%
Sports	67.5%	8.4%	24.1%

Table 10: Accuracy of triples divided into 7 specific domains.

When categorized by domain in Table 10, the LLM's reliability is the highest for locations and persons. Conversely, the LLM performs worst in business and sports domains, where the knowledge is more numeric and time-sensitive. Notably, the politics domain showed a 0% plausibility rate and the highest error rate (27.3%), suggesting the LLM mirrors the polarized, binary nature of political discourse, leading to confident but often incorrect assertions.

Hallucinations We randomly sampled and manually verified triples from GPTKB until we collected 50 triples that could not be verified as correct. Of these 32 triples (64%) had incorrect objects, mostly belonging to related entities, for example, *Ohio's 7th congressional district - currentRepresentative - William McKinley* is incorrect, though he served the 17th congressional district of Ohio; 9 triples (18%) had either non-existent, ambiguous, or long-tail subjects, for example, *John Stewart (British Army officer, died 1811)* and *Chiara Ricciardone* are both long-tail and ambiguous entities; 9 triples (18%) had dubious predicates (likely results of relation clustering errors), for example, *Connections (TV series) - hasMethod* has *tt0078588* as its object, which is actually the IMDb ID of the series.

Result 3: Accuracy

GPTKB's accuracy suggest that GPT-4.1's factual accuracy is lower than that suggested by common factuality benchmarks, even though the accuracy does not degrade towards the tail. Most hallucinations occur via incorrect objects on real subjects.

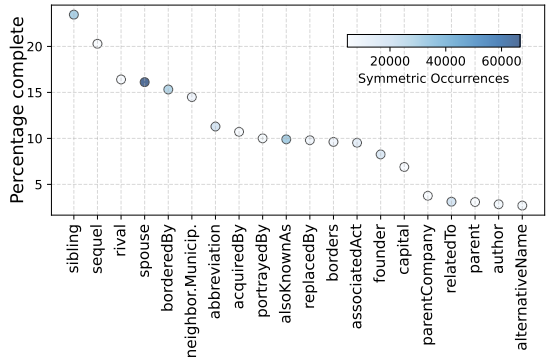


Figure 5: Percentage of total triples present symmetrically for the 20 most frequent symmetric relations.

6 Consistency

In this section and the next we address *RQ4. "How consistent and temporally coherent is GPTKB?"*

Paraphrased Subjects Predicates denoting aliases, such as, *alsoKnownAs*, *alias* and *realName*, are the most common for persons/humans and fictional characters, occurring for a combined total of 36K triples. These alias predicates, occur less frequently in other classes, such as, *city* (1.1K) and *film* (3K). GPTKB contains 112K subjects connected to 124K paraphrased entities via the *alsoKnownAs*, *alias* and *realName* predicates, overcounting at least 2% of the entities.

Symmetric Relations GPTKB has 4,841 relations with at least one reciprocal pair. Figure 5 shows the symmetric completeness of 20 most frequent such relations. The *sibling* relation is the most complete at 23%, followed by *spouse* at 16% completeness. We prompted GPT-4.1 for the missing symmetric counterparts of 100 spouse statements to determine whether the missing statements occurred due to incomplete elicitation. The model returned the missing symmetric object only 9% of the time, an empty response for 13% of the cases and hallucinated for the rest, showing that the LLM cannot judge relation symmetry. We also identify relations, such as, *founder*, *author* and *abbreviation*, that wrongly appear as symmetric relations due to incorrect relation clustering, such as clustering *founder* and *founder of*.

Type Mismatch Our analysis surfaces another consolidation challenge via disambiguation errors that lead to type inconsistencies in GPTKB for polysemous named-entities. In family relations, such as, spouse, parent, sibling, children, we notice a shift from fictional characters to humans/persons and vice-versa in more than 4K triples, where the object elicitation resolves to the more prominent entity. For instance, for the spouse of the character Alexander Hamilton, namely Eliza Hamilton, in the play Hamilton is later generated as a person. Type mismatch more indicative of internal LLM inconsistency is for the *children* relation, which has both entities and integer values as objects.

Result 4: Consistency

GPTKB overestimates parametric knowledge due to at least 2% paraphrased subjects, and symmetric relations are less than 25% complete. Consolidating paraphrased subjects can improve the coverage of symmetric relations.

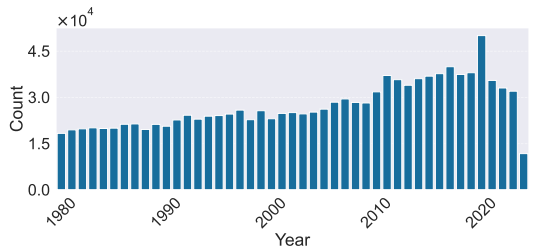
7 Timeliness

Timeliness is a crucial aspect of KBs. LLMs, without access to up-to-date information, are limited to knowledge available up to their training cutoff date. We assess the timeliness of GPTKB via triples whose objects are timestamps. We find them by filtering triples with predicates that contain either "year" or "date". The objects are parsed into YYYY-MM-DD format using regular expression. Any missing day or month is substituted with "00". This leaves us with 2.7M triples whose objects were successfully parsed into the date format.

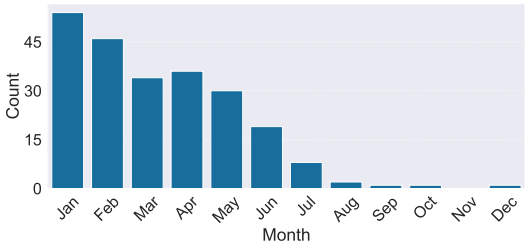
Latest Observed Years and Recency Bias Figure 6 (a) presents the year distribution between 1980 and 2030. There is a pronounced decline in distribution around the reported knowledge cutoff date of GPT-4.1 (June 1, 2024).[3] We zoom into the year 2024, in figure 6 (b), which shows the month distribution of 232 triples that have *deathDate* as their predicate and 2024 as their object year. The distribution declines drastically in the later months, with less than 20 occurrences in June and the following months. Furthermore, we observe that GPT-4.1 exhibits a clear recency bias, with more than 75% of all 2.7M triples having dates after 1917.

Years of Plants and Animals Figure 7 presents the distribution of years in the 18th century, highlighting two notable spikes in 1753 and 1758, with 2,934 and 4,233 occurrences, respectively. In both

[3] https://platform.openai.com/docs/models/gpt-4.1



(a) Year distribution with a peak at 2020.



(b) Month distribution of the death dates in 2024.

Figure 6: Timeliness distribution in GPTKB.

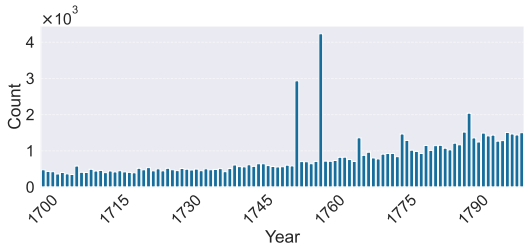


Figure 7: Year distribution (1700-1800) with observed spikes at 1753 and 1758.

cases, the most popular predicate is *describedYear* (more than 80% of all triples), and the subjects of these triples are scientific names of either plants or animals. Historically, these peaks correspond to pivotal milestones in taxonomy: the publication of Carl Linnaeus's Species Plantarum (Linnaeus, 1753) and the 10th edition of Systema Naturae (Linnaeus, 1758), which laid the foundations of modern botanical and zoological nomenclature.

Result 5: Timeliness

Dates in GPTKB enable a fine-grained tracing of GPT-4.1's knowledge cutoff. The LLM has a recency bias, with significant spikes found in historical editorial events.

8 Implication for Downstream Tasks

Our main findings from **RQ1** are that GPT's knowledge is vast, remembering over 1M persons, but at the same time well-filtered, virtually without any sensitive content. The broad taxonomy versus narrow knowledge content indicates a critical knowledge gap in the underlying LLMs. Also, multilingual knowledge clusters indicate complementary language-specific content that remain untapped.

Our bias quantification via nationality and gender ratios to examine **RQ2** reveals that the English knowledge is biased towards the English-speaking world and exhibits a bias towards female entities, though not for non-English locales. Given that only three of the global top-10 nationalities make it to the top-10 nationalities in GPTKB and just 22% of the entities from the classes *human* and *person* mention their gender reveals an imbalanced representation in GPT-4.1 surfaced by GPTKB.

The stability of the true assertions across layers (Figure 4a) and domains (Table 10) indicate consistent knowledge that extends into the long tail (**RQ3**). Yet, this average factual accuracy of 75% is significantly below other benchmark results.

Findings from **RQ4** confirm previous sample-based studies on consistency. We observe significant logical inconsistencies, like relation asymmetry, redundancy, and type mismatches, which can inform future knowledge consolidation strategies. Time-stamped triples help estimate the knowledge cut-off indicating where newer information is necessary. GPTKB preserves historical editorial events evident from significant spikes.

9 Conclusion

With this paper we provide the framework and the first multi-dimensional analysis of LLM knowledge at scale, based at over 100M facts extracted from GPT-4.1 into GPTKB v1.5 (Hu et al., 2025a). Due to cost, this analysis cannot be easily repeated with other LLMs, but our results stand exemplarily for the knowledge of a 2025 frontier model, and notably, could be performed even for closed-source API-only models.

10 Limitations

In this work, we analyze a closed-source, commercial model. Without access to the model's training data, we cannot estimate the LLM's capacity for retaining knowledge. Access to training data, as in

the case of open-source models, would provide a deeper insight into knowledge coverage and bias propagation from the data to the LLM parameters and generated knowledge. Nonetheless, the presented analysis paradigm stands out for enabling to gain insights into potentially any LLM, including closed source, commercial ones, as we indeed show in this work. This might help us to shed light on such models' training data and post-training practices e.g., active gender de-biasing.

While our analysis offers empirical evidence regarding the quality of knowledge stored in GPT-4.1—examining aspects such as accuracy, bias, and inconsistencies, as well as its similarities and differences compared to classical knowledge sources like Wikidata and the Web—these findings may not necessarily generalize to other LLMs.

11 Ethical Considerations

When analyzing very large amounts of data, such as GPTKB, there is a potential risk of encountering sensitive information. However, our findings indicate that such instances are absent in GPTKB. It is important to note that our analysis does not offer formal privacy guarantees and should be distinguished from research specifically focused on privacy leakage in LLMs under extraction attacks (Brown et al., 2022; Lukas et al., 2023).

References

Maryam Amirizaniani, Elias Martin, Tanya Roosta, Aman Chadha, and Chirag Shah. 2024. Auditllm: A tool for auditing large language models using multiprobe approach. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 5174–5179, New York, NY, USA. Association for Computing Machinery.

Anthropic. 2024. Claude 3.5 Sonnet model card addendum. Accessed: 2026-09-09.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. DBpedia: A nucleus for a web of open data. In *ISWC*.

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? In *Conference on Fairness, Accountability, and Transparency*, page 2280–2292.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023a. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicolas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023b. Extracting training data from diffusion models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5253–5270. USENIX Association.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1306–1313.

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling the internal knowledge-base of language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1856–1869.

Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *EMNLP*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545.

Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, Moritz Müller, and Simon Razniewski. 2025a. GPTKB v1.5: A massive knowledge base for exploring factual LLM knowledge. *arXiv preprint arXiv:2507.05740*.

Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, and Simon Razniewski. 2025b. Enabling llm knowledge analysis via extensive materialization. In *ACL*, pages 16189–16202.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *TACL*, 8.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomás Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*, pages 427–431.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *EACL*, pages 3250–3258.

Carl Linnaeus. 1753. *Species plantarum*. Laurentius Salvius, Holmiae (Stockholm).

Carl Linnaeus. 1758. *Systema naturae: per Regna Tria Naturæ, secundum Classes, Ordines, Genera, Species, cum Characteribus, Differentiis, Synonymis, Locis. Editio decima, reformata*. Impensis Direct. Laurentii Salvii, Holmiae (Stockholm).

Nils Lukas, Ahmed Salem, Robert Sim, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. 2023. Analyzing leakage of personally identifiable information in language models. In *Symposium on Security and Privacy*, pages 346–363.

Tuan-Phong Nguyen and Simon Razniewski. 2022. Materialized knowledge bases from commonsense transformers. In *Workshop on Commonsense Representation and Reasoning (CSRR)*.

OpenAI. 2025. Introducing GPT-4.1 in the API. Accessed: 2026-09-09.

Marinela Parović, Ze Li, and Jinhua Du. 2025. Generating domain-specific knowledge graphs from large language models. In *Findings of ACL*, pages 11558–11574.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP*.

Simon Razniewski, Shrestha Ghosh, Luca Giordano, Yujia Hu, Josua Kowalzik, and Tuan-Phong Nguyen. 2026. Epistemic twins: Enabling a symbolic science of language model knowledge. *Workshop on Automated Knowledge Base Construction (AKBC)*.

Hannah Ritchie and Max Roser. 2019. Gender ratio. *Our world in data*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *EMNLP*, pages 4222–4235.

Fabian M Suchanek, Mehwish Alam, Thomas Bonald, Lihu Chen, Pierre-Henri Paris, and Jules Soria. 2024. Yago 4.5: A large and clean knowledge base with a rich taxonomy. In *SIGIR*, pages 131–140.

Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-tail: How knowledgeable are large language models (LLMs)? A.K.A. will LLMs replace knowledge graphs? In *NAACL*.

Amos Tversky and Daniel Kahneman. 1973. Availability: A heuristic for judging frequency and probability. *Cognitive Psychology*, 5(2).

Blerta Veseli, Simon Razniewski, Jan-Christoph Kalo, and Gerhard Weikum. 2023. Evaluating the knowledge base completion potential of GPT. In *Findings of EMNLP*.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10).

Wikidata team. 2025. Wikidata: Sparql query service/wdqs graph split. Available at: `https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/WDQS_graph_split`. Accessed: 13 August 2025.

Qinyuan Wu, Mohammad Aflah Khan, Soumi Das, Vedant Nanda, Bishwamittra Ghosh, Camila Kolling, Till Speicher, Laurent Bindschaedler, Krishna Gummadi, and Evimaria Terzi. 2025. Towards reliable latent knowledge estimation in llms: Zero-prompt many-shot based factual knowledge extraction. In *WSDM*, page 754–763.

Yahya Yunus, Tianwa Chen, and Gianluca Demartini. 2025. Exploring wikipedia gender diversity over time – the wikipedia gender dashboard (wgd). *arXiv preprint arXiv:2501.12610*.

Danna Zheng, Mirella Lapata, and Jeff Z. Pan. 2024. Large language models as reliable knowledge bases? *CoRR*, abs/2407.13578.

Edge Knowledge Bases for Sustainable and Explainable AI

Aparna S. Varde[1], **Emil Njor**[1], **Francesco Daghero**[2], **Mahyar Tourchi Moghaddam**[2]
1. Center for Software Technology, SDU Vejle
2. Dept. of Software Engineering, SDU Odense
MMMI, University of Southern Denmark (SDU), Denmark
ORCID: 0000-0002-3170-2510 (Varde), 0000-0003-0219-1624 (Njor),
0000-0001-9595-7216 (Daghero), 0000-0001-5028-7546 (Moghaddam)
{apva,njor,fdag,mtmo}@mmmi.sdu.dk

Abstract

We propose a vision of **EdgeKBs** (Edge Knowledge Bases) integrating KBs into Edge AI systems. EdgeKBs can infuse commonsense and domain knowledge within Edge devices to guide them while learning from training data. By doing so, we not only bring explainability and performance benefits of KBs to the Edge, but also enable more efficient, sustainable Edge AI systems. We envisage three approaches to develop EdgeKBs: Slicing & Dicing; Tailoring to Fit; and Dynamic EdgeKBs. These EdgeKBs will bring Knowledge-Guided Machine Learning (KGML) to the Edge to support next-generation applications, e.g., Physical AI, cobots, and local natural language interfaces.

1 Introduction

Background: As AI expands beyond centralized cloud environments to embedded & network-edge settings, performing inference near data sources can deliver quality gains: reduced latency, greater bandwidth efficiency, and enhanced privacy (Varghese et al., 2016). Advances in knowledge-intensive natural language processing (NLP) show that explicit external memory can improve factuality & provenance compared to purely parametric models (well-known in Retrieval Augmented Generation (RAG)) (Lewis et al., 2020), (Wu et al., 2022). More generally, the paradigm *Knowledge-Guided Machine Learning (KGML)* "aims to use both scientific knowledge and data in ML frameworks to achieve better generalizability, scientific consistency and explainability" (Karpatne et al., 2026). KGML studies (Serra et al., 2025), (Austin-Gabriel et al., 2025) add interpretability and raise accuracy as well as efficiency (enhancing sustainability).

Related Work: In KBs, approaches such as the classical Cyc (Lenat, 1995) and WordNet (Fellbaum, 1998) as well as the recent ConceptNet 5.5 (Speer et al., 2017) for commonsense knowledge (CSK) and CANDLE for cultural CSK (Nguyen et al., 2023), indicate that structured knowledge can be curated at scale. CSK extraction, compilation and use in machine intelligence can be vital (Razniewski et al., 2021), (Tandon et al., 2018). Yet, KB design is rarely co-designed with Edge deployment constraints (memory, latency, maintenance). Emerging RAG frameworks well-optimize local vector retrieval in text generation, but are bounded by unstructured data representations and passive storage (Kang et al., 2026; Solanki et al., 2026; Qin et al., 2024; Zhu et al., 2024). Recent work (Liu et al., 2022) covers resource constraints in physics-informed neural networks.

Novelty and Contributions: Motivated by the state-of-the-art and excited by the gaps to be filled, this vision paper proposes to build **EdgeKB** (Edge Knowledge Bases) to enable **EdgeKGML** (Knowledge-Guided Machine Learning on Edge devices). Such EdgeKBs must provide a small, focused, practical system for managing knowledge that works well on Edge devices. Building these new EdgeKBs entails novel paradigms of Slice and Dice, Build from Scratch, and Dynamic EdgeKBs (elaborated in the paper). They contribute to sustainable and explainable AI for knowledge extraction, storage, reasoning, and updating for devices with limited resources, e.g. cobots, inbolts, NLP-based virtual assistants, and other next-generation systems. This initiative can enable researchers to address further issues such as:
(1). Model Compression : Can large-scale commonsense and domain knowledge be compressed in compact EdgeKBs without sacrificing relational information for reliable local inference?
(2). EdgeKB Maintenance: How should EdgeKBs be updated, valued, and orchestrated to enhance latency, privacy, robustness, and explainability under resource constraints?

We elaborate on our proposed approaches and experimental plans in this vision paper.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 19–25
October 28, 2026 ©2026 Association for Computational Linguistics

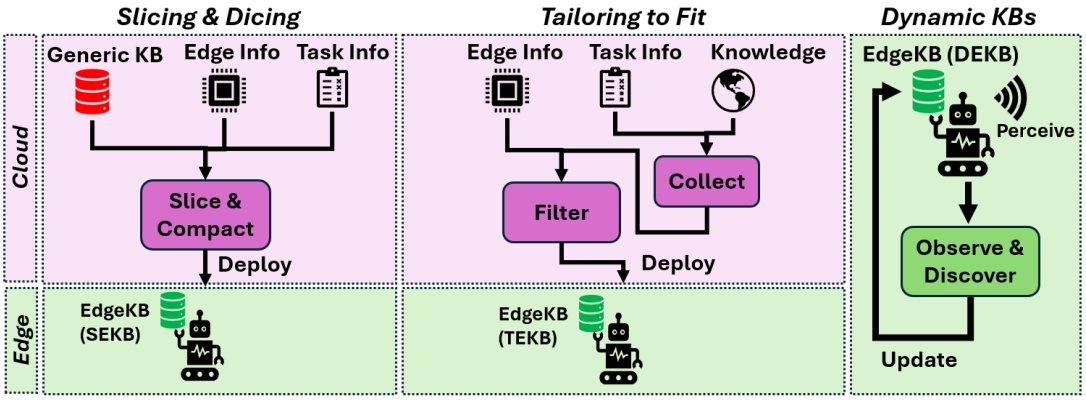


Figure 1: Overview of proposed approaches for EdgeKB development.

2 Proposed EdgeKB Approaches

We propose three archetype approaches to construct EdgeKBs. They are not meant to be mutually exclusive, e.g., some EdgeKBs might be created by their combination. See Figure 1 for a visualization of the approaches.

2.1 Approach 1: Slicing and Dicing

As a first approach, we propose to slice one or more existing KBs to derive an EdgeKB that fits within Edge device constraints and context-dependencies. We call the resulting KB a **Sliced EdgeKB (SEKB)**. It can be built by one of the following methods.

(i) **Deploy a Multifaceted KB:** A multifaceted KB such as Dice (Chalier et al., 2020b,a) can be applied to slice commonsense KBs (ConceptNet (Speer et al., 2017), WebChild (Tandon et al., 2014), Quasimodo (Romero et al., 2019), CANDLE (Nguyen et al., 2023) etc.) or domain KBs (ChEBI (Degtyarenko et al., 2007), GeneOntology (Aleksander et al., 2023) etc.) Dice has clauses (Plausibility, Typicality, Remarkability, Salience) helpful to curate knowledge from other KBs.

$$\rho(e_1, p) \wedge \lambda(e_1, e_2) \wedge \neg\rho(e_2, p) \implies \mu(e_1, p) \quad (1)$$

$$\tau(e, p) \wedge \mu(e, p) \implies \sigma(e, p) \quad (2)$$

Consider the above equations (1) and (2) from Dice, potentially useful to build the SEKB by slicing. Here, ρ means *plausible*, λ is *related*, μ is *remarkable*, τ is *typical*, σ is salient; e, e_1, e_2 are entities, p being properties. These equations can express facts, e.g., in domestic robotics, if it is *plausible* to find an *air-fryer* item as a property of a *kitchen* and *not plausible* to find it in a related entity *laundry room*, an *air-fryer* is *remarkable* for a *kitchen*. By calculating scores for each clause, aggregated scores can be computed to extract info from other KBs with reference to context, e.g., retain info on "salient" entities only or on all "plausible" ones.

While we exemplify the method with Dice, other multifaceted KBs, e.g., Octopus (Yang et al., 2002) can be used as well.

(ii) **Harness LLMs:** Large language models such as GPT-4o (OpenAI, 2024), and DeepSeek-R1 (Guo et al., 2025) can also be applied for slicing existing Knowledge Bases (KBs). We can conduct LLM slicing using either of the following options.

(a) Semantic Chunking: This is a method analogous to semantics-preserving RAG chunks (Puspitasari et al., 2026). Here, relevant KBs are passed to a given LLM with instructions to split them based on general concepts, e.g., "everything relevant to domestic robotics". Hence, this method can slice big chunks of knowledge for the SEKB.

(b) Knowledge Graph Extraction: This method is similar to Microsoft GraphRAG (Edge et al., 2024). Here, an LLM can be prompted to identify more specific entities and relationships, thus knowledge slices can be mapped into structured SPO (subject-predicate-object) triples. LLM prompts can mention specific entities, e.g., only "kitchen", "bedroom", or "living room", along with specific relationships, e.g., "containment". It can then extract SPO triples such as "refrigerator cools food", i.e., *S=refrigerator, P=cooling, O=food.* The most relevant SPO triples can then be stored in the SEKB.

2.2 Approach 2: Tailoring to Fit

Edge AI applications often work in unique environments which necessitate tailored models & data to achieve peak performance (Njor et al., 2024). Thus, while some EdgeKBs can be extracted from slices of existing traditional-scale KBs, there is often a need for an EdgeKB tailor-made from scratch for the Edge AI to fit unique characteristics of applica-

tions. We call this a **Tailored EdgeKB (TEKB)**.

One primary driver for tailoring a new EdgeKB from scratch will likely be the lack of an appropriate existing KBs to adapt. Perhaps the application is one that only exists for Edge AI, such as an AI system deployed on an IoT sensor, which judges whether captured data is simple enough to be processed locally, or whether it is so complex that it must be sent to a larger machine. Another example can be EdgeKBs for wearable devices in medicine (e.g., intelligent pacemakers) or drones on the Edge with natural language interaction.

In some cases, KBs suitable for the application may exist, but they are constructed or licensed in a way that prevents slicing them to fit Edge AI constraints. In other cases, designing an EdgeKB from scratch will allow for faster development. No matter the reason, tailoring an EdgeKB to fit the needs of the application enables highly domain-specific, specialized data structures and custom semantics to reduce data storage and resource consumption.

We aim to distinguish between two TEKB types:

(i) **Personalized EdgeKB:** This type of TEKB is bespoke to a specific Edge AI system. It tightly fits device constraints, and is designed to leverage any specialized hardware for maximum performance.

(ii) **General EdgeKB:** This TEKB type, on the other hand, is designed for the Edge, but not necessarily a specific Edge AI system. It will be resource-efficient, but will follow ecosystem-wide norms rather than system-specific constraints & hardware.

We propose that tailoring an EdgeKB from scratch (TEKB) entails the following. (1). Define constraints of the Edge AI system (Maximum memory use, energy consumption, latency). (2). Evaluate which data structures and custom semantics can support better resource efficiency for the EdgeKB. (3). Address whether the EdgeKB will be able to query other KBs on a network, or if it must act autonomously. (4). Consider how AI systems and users will query the EdgeKB, e.g., through natural language interfaces. (5). Decide whether the EdgeKB should be personalized or general-purpose.

2.3 Approach 3: Dynamic EdgeKBs

Rather than remaining static artifacts, some EdgeKBs should be able to adapt to environments in which they are deployed. Enabled by increasingly capable Edge AI models, e.g., Small Language Models (SLMs) (Wang et al., 2025) and lightweight foundation models, Edge devices may evolve by continuously identifying, deriving, organizing and refining knowledge to maintain up-to-date information for their goals. We call the evolved KB a **Dynamic EdgeKB (DEKB)**, and claim that this evolution can occur through two complementary paradigms as explained next.

(i) **KG refinement:** This paradigm of refining a knowledge graph (KG) entails dynamically adapting the respective KBs initially obtained offline, updating outdated information, refining existing relations, and incorporating context-dependent variations observed during operation. It preserves the original KB structure while ensuring that its content gets aligned with the current environment.

(ii) **KG derivation:** In contrast, this other paradigm of deriving a knowledge graph delegates management of the given KB entirely to the Edge AI system. The Edge model autonomously discovers new entities and relations, creates application-specific abstractions, validates knowledge against local observations, and continuously restructures the KB as per evolving requirements.

2.4 Cross-Approach Considerations

Rather than storing all acquired information locally, often unfeasible due to the devices' resource constraints, some Edge AI systems may be connected to larger KBs over a network. Knowledge that is often accessed or essential for real-time decision-making or privacy may be retained locally, while less relevant information may be compressed, summarized, or migrated to nearby fog nodes and cloud repositories.

Conversely, when new tasks emerge, relevant knowledge may be proactively retrieved from remote nodes and materialized on-device before it is required. Such adaptive placement strategies effectively transform knowledge into a managed resource, analogous to memory or computation in traditional edge systems.

3 EdgeKBs for EdgeKGML

By introducing EdgeKBs to Edge AI, we can bring many benefits that other AI paradigms have harvested to the Edge. Future Edge AI systems supported by EdgeKBs will be more trustworthy & explainable, as their decisions will be supported by verified information. We also expect that EdgeKBs will make Edge AI achieve better predictive performance. Uniquely for Edge AI, this increase in predictive performance can be traded for resource

Table 1: Evaluation Metrics for Edge Knowledge-Guided Machine Learning Systems

Category	Metric	Description	Unit
Hardware	Static Memory (Storage)	Disk/Flash space required to store AI model parameters and the KB index.	MB / GB
	Dynamic Memory (RAM)	Peak working memory consumption during concurrent KB retrieval and model inference.	MB / GB
	Energy Consumption	Total energy consumed per query/batch, critical for battery-constrained edge devices.	mJ / J
	Latency	End-to-end processing time from input reception to final output generation.	ms
AI Model	Accuracy	Proportion of correct predictions or classifications out of all evaluated samples.	%
	F1-Score / Precision	Performance metrics for robustness, especially useful for imbalanced edge datasets.	0–1
Unified	Storage Ratio (KB:AI)	Ratio of static memory footprint of the KB versus the AI Model parameters.	Ratio
	Compute Ratio (KB:AI)	Proportion of processing time (or FLOPs) spent on KB retrieval versus AI model inference.	Ratio
	KB Hit/Relevance Rate	Percentage of inferences where the KB successfully provided useful context.	%

efficiency by using smaller AI models (Banbury et al., 2024), allowing EdgeKBs to make Edge AI more sustainable. By pushing structured, adaptable knowledge directly to resource-constrained devices, EdgeKBs unlock a new class of applications that fundamentally improve performance, trustworthiness, and sustainability of local AI deployments. Such applications can potentially entail EdgeKGML, deploying knowledge from EdgeKBs. We synopsize some of these applications here.

Physical AI: While current Edge AI systems can accurately detect & track objects (Yrjänäinen et al., 2020), adequate interaction with the physical world needs a deeper understanding of affordances (Zhang et al., 2016), task constraints, and safety regulations. In domains like manufacturing, logistics, and warehouse automation, EdgeKBs can bridge this gap. Crucially, this enables next-gen Physical AI, e.g., domestic cobots & motion-tracking inbolts (Kolvig-Raun, 2026). Embedding EdgeKBs (commonsense, domain knowledge) in such systems can bring them closer to thresholds of human cognition.

Edge Natural Language Interfaces: Natural language interfaces operating in high-stakes domains, e.g. industrial machinery, cannot rely solely on generic LLMs (Zheng et al., 2025) that may be unavailable at critical moments. EdgeKBs can enable deployment of highly specialized conversational agents, intrinsically grounded in specific maintenance workflows and safety protocols.

Private Smart Assistants: Smartphone-based assistants (e.g., text-based / voice-operated (Sateli et al., 2013), (Kalvakurthi et al., 2023)) can demonstrate the immense utility that can be offered by dynamic EdgeKBs in the future. Phones can be initialized with a sliced commonsense knowledge base alongside a compact, domain-specific repository. Runtime adaptation can then build a deeply personalized knowledge layer that can capture user routines, preferences, and frequent interactions.

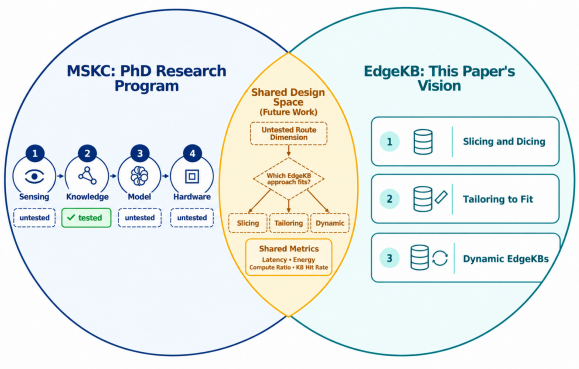


Figure 2: Experimentation Example — MSKC (minimum sufficient knowledge computation) in targeted applications, e.g., decision support of wind turbines for renewable energy, in conjunction with EdgeKB development: Knowledge routing is partly tested, the rest of the parts can be mounted with EdgeKB components.

4 Experimentation Plans

The intersection of Edge AI and knowledge bases can offer ideal opportunities to study the benefits they can achieve together. As EdgeKBs mature, we envision promising research trajectories as follows.

- Investigating how EdgeKBs improve Edge AI

system performance by grounding decisions in real information, reducing hallucinations.
- Applying EdgeKBs to improve resource efficiency, e.g., trade-off in performance gains achieved by EdgeKBs vs. efficiency due to less computationally intensive models.
- Investigating how EdgeKBs constructed on the Edge can pave the way for decentralized construction of larger Cloud KBs
- Exploring how large-scale commonsense & domain knowledge can be compressed into compact EdgeKBs without sacrificing relational info needed for reliable local inference.
- Studying how EdgeKBs updated over time offer privacy, robustness, explainability, and low latency, yet respecting resource constraints.

We aim to evaluate solutions across these trajectories, especially with respect to the metrics in Table 1. An example of an outline for our experimentation appears in Figure 2, indicating how ongoing work on the research sub-problem of a PhD student (Sengupta et al., 2026a) furthering from prior work (Sengupta et al., 2026b) can blend with the EdgeKB vision in this paper.

5 Conclusions and Roadmap

Summary: In this paper, have presented our proposed vision of EdgeKBs being compact, curated knowledge bases co-designed with the resource constraints of Edge devices. We have outlined three complementary paths to building them: *(i)* slicing existing large-scale KBs, *(ii)* tailoring new ones from scratch, and *(iii)* allowing deployed KBs to evolve dynamically. We have further illustrated how EdgeKBs can pave the way to potentially help in bringing Knowledge-Guided Machine Learning to the Edge (i.e., EdgeKGML) from Physical AI to local language interfaces.

The benefits of grounding local inference in explicit, verifiable knowledge are that Edge AI becomes more explainable and trustworthy, while the resulting accuracy gains can be traded for smaller models, thus lowering energy and memory demands; thereby making Edge AI more sustainable. We see offline EdgeKB construction as a stepping stone towards inference-time KBs being assembled on the fly, as time progresses.

Usefulness: It is well-known that data is ever-growing, e.g., a Quettabyte (QB), i.e., a decillion bytes (10^{30} bytes) has already been officially declared through SI Units in 2022 to future-proof global data measurement. Much of the data is multimodal with natural language text, images, video etc. While the big data community aims to build accurate, efficient methods to analyze enormous volumes of data, there are parallel efforts in sustainable AI to achieve the best results with minimal resources, e.g. operating on resource-constrained devices. Likewise, there are trends in adding explainability in learning for twofold contributions: better interpretability and resource-saving. The latter occurs by infusing explanations in learning, derived from KBs (one-time process vs. recurrent learning from huge data). Hence, with the proposed approaches in this paper, we aim to promote KGML along with Edge AI with EdgeKBs.

Our vision paper can be useful in terms of introducing the EdgeKB problem setting, proposing development entailing knowledge base slicing, Edge-native KB construction, and adaptive maintenance. It can envision managing knowledge as a system resource together with computation and memory in next-generation Edge AI, keeping in mind the constantly increasing data and knowledge as we head towards next-generation systems. Our work identifies high-impact application domains in cyber-physical and conversational contexts so as to outline a roadmap from conceptualization to practical feasibility.

We invite the NLP, KB and Edge Computing communities to pursue our research agenda. We anticipate that the approaches, experiments and metrics proposed in this vision paper will be exciting to these research communities.

6 Acknowledgements

Aparna Varde acknowledges a research grant NF25OC011003 from NNF, Denmark. Francesco Daghero and Mahyar Tourchi Moghaddam acknowledge the REINS project, supported by Next-Generation AI (Forskningsreserven) and Digital Research Centre Denmark (DIREC). We thank Purbak Sengupta, a PhD candidate at CST, MMMI, SDU Vejle, for his inputs related to an illustration in this paper. All the authors are at the Maersk McKinney Moller Institute (MMMI) in the Faculty of Engineering (TEK) at the University of Southern Denmark (SDU).

References

Suzi A Aleksander, James Balhoff, Seth Carbon, J Michael Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris, and 1 others. 2023. The gene ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031.

Blessing Austin-Gabriel, Aparna S Varde, and Hao Liu. 2025. Geoinformatics-guided machine learning for power plant classification. In *AAAI Conference on Artificial Intelligence (Bridge Program)*.

Colby Banbury, Emil Njor, Andrea Mattia Garavagno, Mark Mazumder, Matthew Stewart, Pete Warden, Manjunath Kudlur, Nat Jeffries, Xenofon Fafoutis, and Vijay Janapa Reddi. 2024. Wake vision: A tailored dataset and benchmark suite for tinyml computer vision applications. *arXiv preprint arXiv:2405.00892*.

Yohan Chalier, Simon Razniewski, and Gerhard Weikum. 2020a. Dice: A joint reasoning framework for multi-faceted commonsense knowledge. In *International Semantic Web Conference (ISWC Demo*, volume 2721, pages 16–20.

Yohan Chalier, Simon Razniewski, and Gerhard Weikum. 2020b. Joint reasoning for multi-faceted commonsense knowledge. In *Conference on Automated Knowledge Base Construction, AKBC 2020, Virtual, June 22-24, 2020*.

Kirill Degtyarenko, Paula De Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. 2007. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36(suppl_1):D344–D350.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Christiane Fellbaum. 1998. Wordnet. *WordNet An Electronic Lexical Database*, page 69.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Vishesh Kalvakurthi, Aparna S Varde, and John Jenq. 2023. Hey dona! can you help me with student course registration? In *AAAI Conference on Artificial Intelligence (Workshop on AI for Education)*.

Dong Ho Kang, Hyunjoon Lee, Hyeonjeong Cha, Minkyu Choi, and Sungsoo Lim. 2026. Pocket rag: On-device rag for first aid guidance in offline mobile environment. *arXiv preprint arXiv:2602.13229*.

Anuj Karpatne, Xiaowei Jia, and Vipin Kumar. 2026. Knowledge-guided machine learning: A paradigm shift in ai for science. In *AAAI Conference on Artificial Intelligence*, page 40(46):39715–39717.

Emil Stubbe Kolvig-Raun. 2026. Motion guidance tracking inbolt.

Douglas B Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Ziyue Liu, Xinling Yu, and Zheng Zhang. 2022. Tt-pinn: A tensor-compressed neural pde solver for edge computing. *Preprint*, arXiv:2207.01751.

Tuan-Phong Nguyen, Simon Razniewski, Aparna S. Varde, and Gerhard Weikum. 2023. Extracting cultural commonsense knowledge at scale. In *ACM Web Conference 2023, WWW*, pages 1907–1917. ACM.

Emil Njor, Mohammad Amin Hasanpour, Jan Madsen, and Xenofon Fafoutis. 2024. A holistic review of the tinyml stack for predictive maintenance. *IEEE Access*, 12:184861–184882.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Fachrina Dewi Puspitasari, Chaoning Zhang, Jiaquan Zhang, Zhicheng Wang, Hafiz Shakeel Ahmad Awan, Rizwan Qureshi, Jewon Lee, Tae-Ho Kim, and Yang Yang. 2026. Efficient rag with intent-aware retrieval and semantics-preserving chunking. *arXiv preprint arXiv:2606.01240*.

Ruiyang Qin, Zheyu Yan, Dewen Zeng, Zhenge Jia, Dancheng Liu, Jianbo Liu, Ahmed Abbasi, Zhi Zheng, Ningyuan Cao, Kai Ni, and 1 others. 2024. Robust implementation of retrieval-augmented generation on edge-based computing-in-memory architectures. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9.

Simon Razniewski, Niket Tandon, and Aparna S Varde. 2021. Information to wisdom: Commonsense knowledge extraction and compilation. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 1143–1146.

Julien Romero, Simon Razniewski, Koninika Pal, Jeff Z. Pan, Archit Sakhadeo, and Gerhard Weikum. 2019. Commonsense properties from query logs and question answering forums. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1411–1420.

Bahar Sateli, Gina Cook, and René Witte. 2013. Smarter mobile apps through integrated natural language processing services. In *International Conference on Mobile Web and Information Systems*, pages 187–202. Springer.

Purbak Sengupta, Aparna Varde, and Emil Njor. 2026a. Sustainable event-triggered knowledge-guided intelligence for decision support. Technical report, University of Southern Denmark.

Purbak Sengupta, Aparna S. Varde, and Emil Njor. 2026b. Carbon-aware adaptive knowledge injection for sustainable, trustworthy ai in resource-constrained predictive modelling. In *Nordic AI Meet*.

Angela Serra, Michele Fratello, Antonio Federico, and Dario Greco. 2025. An update on knowledge graphs and their current and potential applications in drug discovery. *Expert opinion on drug discovery*, 20(5):599–619.

Om Solanki, Lopamudra Praharaj, Deepti Gupta, and Maanak Gupta. 2026. Policy-aware edge llm-rag framework for internet of battlefield things mission orchestration. *arXiv preprint arXiv:2604.09493*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, No. 1.

Niket Tandon, Gerard De Melo, Fabian Suchanek, and Gerhard Weikum. 2014. Webchild: Harvesting and organizing commonsense knowledge from the web. In *ACM WSDM international conference on Web search and data mining*, pages 523–532.

Niket Tandon, Aparna S Varde, and Gerard de Melo. 2018. Commonsense knowledge in machine intelligence. *ACM SIGMOD Record*, 46(4):49–52.

Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S Nikolopoulos. 2016. Challenges and opportunities in edge computing. In *2016 IEEE international conference on smart cloud (SmartCloud)*, pages 20–26. IEEE.

Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qiuhao Lu, Wanjing Wang, Rui Li, Junjie Xu, Xianfeng Tang, and 1 others. 2025. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with llms, and trustworthiness. *ACM Transactions on Intelligent Systems and Technology*, 16(6):1–87.

Yuxiang Wu, Yu Zhao, Baotian Hu, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2022. An efficient memory-augmented transformer for knowledge-intensive NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5184–5196, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jun Yang, Qing Li, and Yueting Zhuang. 2002. Octopus: aggressive search of multi-modality data using multifaceted knowledge base. In *Proceedings of the 11th international conference on World Wide Web*, pages 54–64.

Jukka Yrjänäinen, Xingyang Ni, Bishwo Adhikari, and Heikki Huttunen. 2020. Privacy-aware edge computing system for people tracking. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 2096–2100. IEEE.

Zuo Zhang, Yaoru Sun, and Glyn W Humphreys. 2016. Perceiving object affordances through visual and linguistic pathways: A comparative study. *Scientific reports*, 6(1):26806.

Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. 2025. A review on edge large language models: Design, execution, and applications. *ACM Computing Surveys*, 57(8):1–35.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2024. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577.

A Temporal Knowledge Graph for Music Festival Lineup Forecasting

Julia Gastinger, Thilo Dieing, Christian Meilicke, Heiner Stuckenschmidt
University of Mannheim, Mannheim, Germany
first.last@uni-mannheim.de

Abstract

Music festival lineups emerge from complex relationships among artists, genres, releases, labels, and past performances, making the prediction of future lineups a natural fit for temporal knowledge graph (TKG) forecasting. In this work, we present a TKG covering 380 festivals over 55 years, comprising more than 90K festival performance quadruples along with information on festivals, artist tours, and artist metadata, and release it as a resource for TKG forecasting evaluation. We formalize festival lineup forecasting as temporal link prediction between artists and festivals at future timestamps. We evaluate six TKG forecasting models on this task, analyze their capabilities and limitations, and compare them against Large Language Models applied zero-shot. Our resource complements existing TKG benchmarks by grounding evaluation in a concrete, real-world application domain.

1 Introduction

Temporal knowledge graphs (TKG) incorporate temporal information into knowledge graphs (KG) (Han et al., 2021a). The task of TKG forecasting, which aims to predict future links from historical information, has recently gained increasing attention and motivated the development of a variety of approaches (Li et al., 2021; Liu et al., 2022; Chen et al., 2025).

Alongside these modeling advances, recent work has begun to address the evaluation of TKG forecasting, proposing a unified evaluation protocol, a heuristic baseline, large-scale datasets, and an evaluation framework (Gastinger et al., 2023, 2024b,a).

Yet evaluation has so far relied on benchmarks not tied to any concrete application scenario, and the resulting prediction tasks do not necessarily correspond to questions a practitioner would meaningfully want to answer. Most established benchmarks are extracted from political event data and news sources via automated NLP pipelines (e.g., ICEWS (Boschee et al., 2015) and GDELT (Leetaru and Schrodt, 2013)) or based on general-purpose knowledge bases (e.g., Wikidata (Vrandečić and Krötzsch, 2014), YAGO (Mahdisoltani et al., 2015)). Creating TKGs from event extraction pipelines can result in queries that cannot be interpreted without additional contextual knowledge. The TKGs derived from knowledge bases often include facts that remain stable over long period, which results in a task that primarily requires repeating previously stated facts rather than forecasting (Gastinger et al., 2024b).

These characteristics reflect the gap between general-purpose evaluation and the needs of application scenarios. This motivates the development of complementary resources grounded in concrete use cases, with well-defined entities, interpretable relations, and realistic temporal dynamics.

Music festival lineups are shaped by complex interactions among artists, festivals, genres, labels, and past performances. Hiller (2016) shows that hiring decisions depend on factors such as an artist's past festival appearances, album quality, and touring history across multiple years. This makes music festival lineups and the related interactions a natural fit for temporal relational modeling. Beyond their suitability as a research domain, music festivals represent a significant cultural and economic sector. In Germany alone, the festival landscape generates estimated annual revenues of €551 Mio., with artist fees accounting for approximately 38% of expenditures (Sommer et al., 2025). Consequently, predicting future festival lineups is directly relevant to different groups within the festival ecosystem: It can support organizers in curating lineups, help artists and managers identify suitable events, and assist fans in planning attendance decisions, especially since many festivals sell early-bird tickets before full lineups are revealed.

In this work, we construct a TKG of music fes-

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 26–44
October 28, 2026 ©2026 Association for Computational Linguistics

tival lineups as benchmark for TKG forecasting models. We focus on the following contributions:

- We provide a TKG spanning 55 years and 380 festivals, integrating festival lineups, concerts, venues, and artist metadata, along with a documented, reproducible construction pipeline.
- We introduce the task of lineup forecasting. It focuses on relations of interest for the given use case, and covers two query directions (artist-prediction and festival-prediction). In this context, we argue for two metrics better suited to the many-correct-answer structure of lineup queries.
- We conduct an empirical evaluation of six forecasting models and zero-shot LLMs, analyze behavioral differences, and discuss strengths and open issues.

Resource Availability: Source code, including dataset creation, evaluation, tested baseline implementations, a lightweight ontology file, an RDF-star version of our dataset, and Croissant metadata are available on `https://github.com/JuliaGast/festival-lineup-tkg`. The final TKG datasets are available on `https://madata.bib.uni-mannheim.de/822/`.

2 Prediction Task

The task in this work is to predict future lineups of music festivals based on historical information. We model this historical information as a temporal knowledge graph, and the task of predicting future festival lineups as a TKG forecasting task, i.e., the task of predicting future links in a TKG.

A temporal knowledge graph G is a set of quadruples (s, p, o, t) with entities (or nodes) $C(G) = \{s \mid (s, p, o, t) \in G\} \cup \{o \mid (s, p, o, t) \in G\}$, relations $P(G) = \{p \mid (s, p, o, t) \in G\}$, and timestamps $T(G) = \{t \mid (s, p, o, t) \in G\} \subset \mathbb{N}$. Timestamps may represent hours, days, years, or any other temporal granularity, depending on dataset and use case. In our case, timestamps are years. The semantic meaning of a quadruple (also referenced as a temporal triple) (s, p, o, t) is that s is in relation p to o at time t. An example is *(Arlo Parks, performs at festival, MS Dockville, 2023)*.

Given a TKG G, TKG forecasting or extrapolation is the task of predicting quadruples $(s, p, o, t^\star)$ for future timestamps $t^\star > max(T(G))$ with $p \in P(G)$ and $s, o \in C(G)$. In our project, we will focus on the task of entity forecasting, that is, predicting object or subject entities for queries $(s, p, ?, t^\star)$ or $(?, p, o, t^\star)$. Akin to static knowledge graph completion, TKG forecasting is approached as a ranking task (Han, 2022). For a given query, which is usually derived from a quadruple in a test or validation set, a model needs to rank entities in $C(G)$ using a scoring function that assigns plausibility scores (Gastinger et al., 2025).

In our work, we ask two specific types of queries: *(A, performs at festival, ?, $t^\star$)*, i.e. at which festival will a given artist A play at in a future timestamp $t^\star$, and *(?, performs at festival, F, $t^\star$)*, i.e. which artist will perform at a given festival F at a future timestamp $t^\star$. We also refer to these two types of queries as festival prediction and artist prediction.

We focus on single-step prediction, which means we predict one timestamp into the future. Since our dataset has yearly granularity, this means we predict the festival lineup for the next year.

3 Related Work

3.1 TKG Forecasting Models

A wide range of methods has been proposed for temporal knowledge graph forecasting. One prominent line of work combines graph-based message passing with sequential modeling to capture structural and temporal patterns; representative models include RE-Net (Jin et al., 2020), RE-GCN (Li et al., 2021), xERTE (Han et al., 2021a), TANGO (Han et al., 2021b) and CEN (Li et al., 2022b). Further approaches in this direction include history-based prediction (CyGNet (Zhu et al., 2021)), contrastive learning (CENET (Xu et al., 2023)), and combined local and global graph encoding (TiRGN (Li et al., 2022a)). A second direction uses reinforcement learning to discover temporal reasoning paths, with TimeTraveler (Sun et al., 2021) as representative example.

Rule-based methods aim to learn interpretable temporal logic rules. TLogic (Liu et al., 2022) learns such rules via temporal random walks, while CountTRuCoLa (Gastinger et al., 2025) learns temporal confidence functions that account for frequency and recency. Hybrid approaches such as INFER (Li et al., 2025) and CognTKE (Chen et al., 2025) combine rule-based and embedding-based methods. More recently, LLM-based models have been proposed to incorporate semantic knowledge into forecasting, including zrLLM (Ding et al., 2024) and GenTKG (Liao et al., 2024). Finally, the Recurrency Baseline (Gastinger et al., 2024b) is a heuristic baseline that predicts fact recurrence by incorporating temporal distance and frequency.

3.2 TKG Forecasting Benchmark Datasets

Most established benchmark datasets are derived from political and news event data, including the Integrated Crisis Early Warning System (ICEWS) (Boschee et al., 2015; Shilliday et al., 2012), the Global Database of Events, Language, and Tone (GDELT) (Leetaru and Schrodt, 2013), and POLECAT (Scarborough et al., 2023). In addition, some datasets are constructed from knowledge bases such as Wikidata (Vrandečić and Krötzsch, 2014) and YAGO (Mahdisoltani et al., 2015).

Limitations of Benchmarks: While recent work has improved the evaluation of TKG forecasting methods (Gastinger et al., 2023, 2024b,a), the resulting tasks can remain difficult to interpret from an application perspective: evaluation is typically framed as generic future link prediction aggregated over all relations (Gastinger et al., 2024a), often spanning a wide variety of domains within a single dataset. Several issues compound this:

1. Test queries are sometimes unintuitive without access to the full contextual knowledge. For example, `tkgl-polecat` contains triples such as *(Ben Cardin, THREATEN, Federation)* that are impossible to interpret in isolation.
2. Automatic extraction from news text produces relations with heterogeneous domains and ranges. For example, in ICEWS-based datasets, the relation *Arrest, detain, or charge with legal action* connects countries to demographic groups (e.g., *Children (France)*), abstract roles (e.g., *Citizen (Australia)*, *Criminal (Philippines)*), and individual persons (e.g., *Andal Ampatuan Sr*), complicating interpretation and model analysis.
3. Wikidata-derived benchmarks such as `tkgl-smallpedia` and `WIKI` include many queries on facts that are stable across decades (e.g., religion, administrative capitals). More than 70% of test facts in these benchmarks were already true at their previous timestep and a simple recurrency-based baseline is competitive with or superior to most TKG forecasting methods on them (Gastinger et al., 2024b,a). This suggests that high stability leaves limited room above simple persistence.
4. Some datasets contain relations that cannot be reliably inferred from available history. `YAGO`, for instance, provides insufficient context to forecast prize awards or deaths (Gastinger et al., 2024b).
5. The news-based `GDELT` contains not only real-world dynamics, but also media reporting cycles: Its 15-minute resolution leads to repeated mentions of the same event across adjacent timestamps (Gastinger et al., 2025), making it difficult to determine what models actually learn.

These points suggest that benchmark scores may not always reflect a model's ability to answer practically relevant forecasting queries. This motivates the development of resources grounded in concrete use cases with well-defined entities, understandable relations, and realistic temporal dynamics.

4 TKG Creation

4.1 Data Sources

We extract information from two sources: `setlist.fm` (Live Nation Entertainment, Inc., 2026) and `MusicBrainz` (MetaBrainz Foundation, 2026). **`setlist.fm`** is a community-maintained wiki-style platform for collecting and sharing concert setlists. Beyond individual concert setlists, it contains information on venues, festivals, and the artists who play there. The platform provides a public API for data extraction, with all artists, festivals, concerts, and venues assigned unique identifiers. In `setlist.fm`, artists are represented by MusicBrainz identifiers. This enables straightforward cross-referencing between the two data sources. **`MusicBrainz`** is an open music encyclopedia that collects and publishes music metadata, maintained by a global community of contributors. Like `setlist.fm`, it offers an API for straightforward data retrieval.

4.2 Data Mining and Reduction

We initially fetch information on all festivals, their editions, venues, countries, and associated artists from `setlist.fm`. To ensure the resulting TKG remains computationally tractable for current models, we restrict our scope to festivals in Germany[1]. Germany is suitable due to its large, economically significant, and culturally diverse festival sector (Sommer et al., 2025). Moreover, it belongs to the countries with the highest number of entries on `setlist.fm`, ensuring sufficient data coverage.

To focus on established festivals, we apply the following filtering criteria: a festival must have occurred at least 5 times and featured at least 30 unique artists overall, counting only editions with a lineup of at least 10 artists. These thresholds are pragmatic cutoffs to exclude small-scale events. These filters reduce our dataset to 380 festivals.

[1]We provide scripts to mine data for any country/continent.

	Domain	Relation	Range
Performance Information	ARTIST	**performs at festival**	FESTIVAL
	FESTIVAL	happens in venue	VENUE
	ARTIST	performs concert at	VENUE
	VENUE	venue has location	AREA
	AREA	location has country	AREA
Release Information	ARTIST	releases album	RELEASE
	ARTIST	releases EP	RELEASE
	RELEASE	has genre	GENRE-TAG
Meta Data	ARTIST	has begin area	AREA
	ARTIST	has area	AREA
	ARTIST	has type	ARTIST-TYPE
Label Relationships	ARTIST	label founder	LABEL
	ARTIST	recording contract	LABEL
	ARTIST	personal label	LABEL
	ARTIST	artists and repertoire position at	LABEL
	ARTIST	personal publisher	LABEL
	ARTIST	owner	LABEL
	ARTIST	producer position at	LABEL
	ARTIST	creative position at	LABEL
	ARTIST	executive position at	LABEL
	ARTIST	engineer position at	LABEL
	ARTIST	named after label	LABEL
	ARTIST	named after artist	LABEL

Table 1: Data schema for the temporal KG. Each temporal triple also comes with a timestamp.

Given that the earliest festival in our filtered set dates to 1971, we include temporal information from that year onward.

For each artist appearing at these festivals, we extract additional concert data from `setlist.fm`, including all concert dates, venues, and venue locations. To prevent the TKG from being too large for many TKG forecasting models, and to focus on relevant venues, we exclude venues hosting fewer than five concerts, as such venues typically contribute little structural information while substantially increasing graph sparsity.

Finally, we enrich the artist information with metadata from `MusicBrainz` via their API. We extract founding (and, if applicable, ending) as well as geographic information, artist types, label relationships, and release history where available.

We started fetching festival information from `setlist.fm` on Dec 16th, 2025, concert performance data on Feb 9th, 2026, and metainformation from `MusicBrainz` on Feb 14th, 2026.

Table 1 presents the schema for our TKG. The base schema comprises 23 relations organized into four categories: (1) performance information, (2) release information, (3) artist metadata, and (4) label relationships. We provide this schema as a lightweight ontology in the GitHub repository.

4.3 Temporal Information

Our dataset spans from 1971 to 2025. At the time of construction, lineups for 2026 were not yet complete, leading us to conclude the range at 2025.

We employ yearly granularity for the TKG, meaning timestamps represent calendar years. This choice is driven by practical considerations: typical user queries in our use case (e.g., "Which artists will perform at the Southside Festival next year?") do not require finer temporal resolution.

Following common approaches in TKG forecasting (Li et al., 2021; Gastinger et al., 2024a), we encode all temporal information as TKG quadruples, where each triple, e.g., *(Arlo Parks, performs at festival, Melt! Festival)*, has a timestamp, e.g., 2022. Temporal annotation varies by relation:

- Performance events: For triples such as *(ARTIST, performs at festival, FESTIVAL)*, we assign the year of the performance as the timestamp. The same applies to individual concert performances.
- Releases: Release dates (specifically the year) serve as timestamps for album/EP triples and Genre information.
- Artist metadata: Meta-information (e.g., origin area, artist type) is included only when `MusicBrainz` provides a lifespan start date. For persons, this typically corresponds to birth date and place; for groups, to founding date and place. The relation *has begin area* is timestamped with the artist's start date. In contrast, temporal triples with the relations *has area* and *has type* are valid throughout the artist's active period, defined by the `MusicBrainz` lifespan start and end dates. If no end date is specified, we assume the artist remains active through the dataset's endpoint.
- Labels: For label relationships, we use the label relation start and end dates from `MusicBrainz`. When label relation temporal information is unavailable, we default to the artist's active period.
- Geographical information: These triples are typically static. Annotating them at every timestamp would unnecessarily inflate the dataset. Instead, we include them only at timestamps where the corresponding node participates in another relation (e.g., venue location is recorded only in years it hosts a concert).

Representing validity as repeated yearly instances rather than time intervals results in a larger graph size. For example, an artist active for 20 years generates 20 distinct instances of the same temporal triple, each with a different timestamp. We accept this trade-off to maintain compatibility with standard TKG forecasting models (see Section 3), which typically expect this representation instead of an interval-based one.

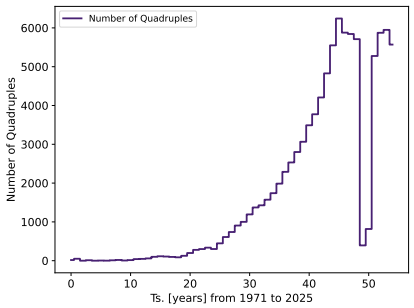


Figure 1: `concert`$_{\text{p}}$, number of quadruples over time.

4.4 Dataset Variants

We present two variants of the TKG, each designed to evaluate specific model capabilities. First, `concert` is the default variant, which encompasses all performance information, release information, artist metadata, and label relationships described in Table 1. And second, `concert`$_{\text{p}}$ is a subset that contains only the quadruples with the relation *performs at festival*. This variant offers a significantly reduced graph size. It serves two purposes: evaluating models constrained by high memory requirements and testing the extent to which the models can leverage additional information by comparing results from this limited version to the default variant, which contains richer information.

Importantly, the prediction task and the ground truth are consistent across both variants. The quadruples corresponding to *performs at festival* exist identically in `concert` and `concert`$_{\text{p}}$, ensuring that test set performance is directly comparable regardless of the background knowledge.

In the GitHub repository, we also provide an RDF serialization of the `concert` dataset. We use RDF-star to encode the temporal triples.

4.4.1 Data Splits

We partition the TKG chronologically into train (1971–2023), validation (2024), and test set (2025). We chose this split to mitigate the impact of the COVID-19 pandemic, which caused a severe disruption in the occurrence of cultural events. Figure 1 depicts the number of quadruples over time for `concert`$_{\text{p}}$, i.e., the number of festival performances. From 1971 to 2019, it shows a steady growth. A sharp drop is visible during the COVID-19 pandemic (2020–2021). Quadruples count returns to nearly pre-pandemic numbers by 2023. To prevent this atypical period from dominating the evaluation, we use 2024 as the first validation year. This ensures at least one year of near-typical festival activity (2023) before evaluation.

4.4.2 Dataset Statistics

Table 2 summarizes statistics for both variants. We report the Recurrency Degree (Rec), which is defined as the fraction of test quadruples (s, p, o, t^+) for which there exists a $k < t^+$ such that $(s, p, o, k) \in G$, and the *Direct Recurrency Degree (DRec)* which is the fraction of temporal triples (s, p, o, t^+) for which it holds that $(s, p, o, t^+{-}1) \in G$ (Gastinger et al., 2024b). Both metrics are computed exclusively for the quadruples containing the relation *performs at festival*.

For forecasting, a particularly relevant measure is the DRec of `concert`$_{\text{p}}$, which quantifies how often triples to be predicted recur. In contrast to the DRecs of established TKG datasets reported by Gastinger et al. (2024b) (e.g., DRec of 92.7% for YAGO), the DRec of `concert`$_{\text{p}}$ is substantially lower. One possible explanation is that consecutive-year re-bookings are relatively uncommon in festival lineups, which leads to fewer directly recurring triples. This low DRec highlights the difficulty of predicting future lineups from past observations: Models must capture dependencies beyond simple recurrence patterns in order to be successful.

Compared to commonly used TKG forecasting benchmarks, the `concert` dataset contains a relatively large number of nodes and quadruples and comparatively few timestamps. Thus, individual temporal snapshots are substantially larger than those of most existing datasets. Appendix A.2 provides a comparison with existing datasets, together with additional dataset statistics, including distributions of relations, genres, and geographic areas.

Constructing and utilizing a TKG for festival lineup prediction involves several design choices and assumptions. We discuss these considerations, along with their limitations, in Appendix A.1.

5 Experiments

5.1 Evaluation Protocol

We follow the evaluation protocol discussed in Gastinger et al. (2023). For evaluation, we use and adapt the TGB 2.0 framework (Gastinger et al., 2024a). We evaluate on single-step prediction. We predict entities in both directions, namely $(s, r, ?, t^\star)$ and $(?, r, o, t^\star)$, achieved by introducing inverse relations, where the head and tail of an existing relation are inverted. As explained in Section 2, we only evaluate on quadruples with the relation *performs at festival* and its inverse.

TKG forecasting is commonly evaluated using

Dataset	#N	#R	#Train	#Valid	#Test	Time Int.	#Tr/Val/ Te TS	DRec [%] for rel. of interest	Rec [%] for rel. of interest
concert	278.936	23	5.068.265	222.507	214.578	1 y.	53/1/1	2.9	29.4
concert_p	25.399	1	83.889	5.949	5.570	1 y.	51/1/1		

Table 2: Dataset statistics: The number of nodes (#N), relations (#R), train, valid, and test quadruples, the time interval, number of train/valid/test timestamps, as well as Direct Recurrency Degree (DRec) and Recurrency Degree (Rec) for the *performs at festival* relation.

time-aware filtered Mean Reciprocal Rank (MRR) and Hits@k (Gastinger et al., 2023). Following the widely adopted filtered ranking protocol introduced by Bordes et al. (2013) for KG completion, MRR and Hits@k are computed per individual test (temporal) triple: for each test triple, the corresponding correct entity is ranked against a set of candidate entities to obtain a reciprocal rank. MRR is the average of these reciprocal ranks across all such queries, and Hits@k measures the fraction of these queries for which the correct entity appears in the top-k ranked scores. If a query has n correct answers, this leads to n test (temporal) triples, each ranked independently. The time-aware filtered variant removes other entities known to be correct at the query time from the ranking, to avoid penalizing correct predictions (Gastinger et al., 2023).

Our use case differs from common TKG evaluation, where usually each query has only a few correct answers: A single festival may feature over 200 performing artists, meaning each query of the form $(?, \textit{performs at festival}, F, t^\star)$ can have hundreds of correct answers. We argue that MRR and Hits@k yield misleadingly low scores in this setting, and therefore report two additional metrics.

To illustrate, consider a festival F with 100 ground truth artists, yielding 100 test quadruples of the form $(A_i, \textit{performs at festival}, F, t^\star)$. Suppose a model produces a ranking in which every entity at an odd-ranked position (1,3,...,199) is correct, and every entity at an even-ranked position is incorrect. This is not a bad ranking, since half of all predictions are correct. For the test quadruple of the artist ranked at position 199, the time-aware filter removes the 99 other correct artists from the ranking, leaving this artist at filtered rank 100. The MRR across all 100 test quadruples for this festival is then $(1{+}1/2{+}1/3{+}...{+}1/100)/100 = 5.2\%$, creating the misleading impression of a near-useless model. Yet a ranking in which every other prediction is correct would intuitively warrant a score near 50%.

We therefore report two additional metrics: The **R-Precision** (Manning et al., 2008) measures how many of the top R_i predicted candidates were correct, where R_i is the number of ground-truth entities for query i: $P_R = \frac{1}{N}\sum_{i=1}^{N}\frac{C_{R_i,i}}{R_i}$ with $C_{R_i,i}$ being the number of correct predictions among the top R_i ranked predictions for query i, and N the number of queries. The **Normalized 10-Precision** measures how many of the top-10 predictions are correct, normalized by the minimum of R_i and 10 to account for queries with fewer than 10 correct answers: $P_{N10} = \frac{1}{N}\sum_{i=1}^{N}\frac{C_{10,i}}{\min(R_i,10)}$ where $C_{10,i}$ is the number of correct predictions in the top 10 ranked predictions for query i.

We consider these metrics more reflective of the real-world use: A user asking for festival prediction cares primarily about how many of the model's top-predicted artists (or festivals) actually appear in the lineup (or tour plan), which is exactly what these metrics capture. In our example above, both metrics would result in a score of 0.5.

5.2 Methods

5.2.1 TKG Forecasting Methods

We compare the following methods: CountTRuCoLa, TLogic, RE-GCN, CEN, CognTKE, and the Recurrency Baseline, since they represent the best performing methods reported across varying datasets in (Gastinger et al., 2025).

We adapt the rule-based methods to only mine rules whose head relation is among the relations of interest, and all methods to produce predictions only for the queries of interest. We describe the hyperparameter selection and the hardware used for our experiments in Appendix A.3.

5.2.2 LLMs

We also test whether a TKG and specialized forecasting pipeline are even necessary, or whether zero-shot LLM prompting can perform competitively without any KG data. This is not a comprehensive evaluation of LLM capabilities, but a check on the necessity of the TKG-based approach.

We evaluate four LLMs spanning both open-source and proprietary model families and covering a range of model sizes: Llama 3.1 8B (Meta, 2024), a quantized Llama 3.3 70B (ibnzterrell, 2026), Mix-

tral 8x7B (Mistral AI, 2023; Jiang et al., 2024), and GPT-4o-mini (OpenAI, 2024). All evaluated models have documented knowledge cutoffs no later than late 2023, ensuring that festival lineups from the 2025 test period cannot have been observed during pretraining. Since our goal is to assess whether broadly accessible general-purpose LLMs can compete with specialized TKG forecasting methods, rather than to benchmark the latest frontier models, we focus on models that are relatively efficient, are largely locally deployable, and practical to reproduce within typical academic compute budgets.

Each model is prompted to produce a ranked list of artists or festivals for a given query; find full prompts in Appendix A.3.2. We request lists of up to 210 artists and 50 festivals per query, thresholds which slightly exceed the maximum number of artists per festival and festivals per artist in the test period. To verify that the training cutoff does not substantially bias performance, we evaluate all open-source models on queries from 2023, 2024, and 2025. We find no evidence that the knowledge cutoff substantially degrades 2025 performance, see full study in Appendix A.5.2. We set the Temperature to 0 throughout to maximize reproducibility. Since LLMs produce a candidate list, instead of a full ranking over all entities, it would be unfair to compute MRR and Hits@k. We therefore report only the P_{N10} and P_R for LLM results.

We do not provide models with the set of TKG entities, reflecting a realistic deployment scenario in which the full candidate set is unavailable. This has two consequences. First, predictions may include entities absent from the TKG, either due to hallucination or because the entity is not present in the dataset. And second, the predictions must be mapped back to TKG nodes before evaluation.

This mapping is conducted as follows: Since LLMs generate free-text entity names that may differ from the TKG node labels, each predicted name is normalized, e.g., by lowercasing, applying Unicode normalization, and removing punctuation. The resulting string is matched against three reference resources: the artist and entity mapping from the TKG, a worldwide festival mapping from `setlist.fm`, and an artist mapping containing all artists and their identifiers from `MusicBrainz`. Matching is performed over a set of string variants generated using simple heuristics such as prefix removal (e.g., “the”, “dj”), suffix stripping (e.g., "open air", "festival"), and comma reordering (e.g., “Franklin, Aretha” to “Aretha Franklin”). Predictions that cannot be resolved against any of the three resources are treated as unmatched. To ensure that we did not miss too many correct predictions, we randomly selected 50 unmatched names and tried to match them manually. In none of these cases did we detect a missing match that would have resulted in a correct prediction.

	predicting artists					predicting festivals				
	MRR	H1	H10	P_R	P_{N10}	MRR	H1	H10	P_R	P_{N10}
no TKG provided										
GPT4-o-m.	-	-	-	0.8	0.7	-	-	-	4.1	14.2
Llama 8B	-	-	-	0.4	0.5	-	-	-	3.1	10.2
Llama 70B	-	-	-	1.3	1.7	-	-	-	7.0	17.2
Mixtral 8x7b	-	-	-	0.6	0.6	-	-	-	5.5	15.9
$\texttt{concert}_\texttt{p}$										
CEN	1.3	0.4	2.5	4.5	5.8	18.0	9.9	34.3	13.1	33.5
Rec B.	2.4	**1.2**	4.2	7.1	9.4	18.3	10.9	33.8	13.4	31.9
RE-GCN	1.4	0.3	3.0	5.1	6.5	16.8	8.9	32.5	11.3	30.5
CognTKE	**2.7**	1.0	**5.5**	**9.6**	**11.9**	21.3	12.5	40.1	14.6	34.4
CountTR.	2.3	0.9	4.3	7.5	10.2	**23.0**	12.5	**44.3**	15.6	**41.4**
TLogic	0.8	0.2	1.5	3.2	3.3	11.7	6.9	22.5	9.5	21.5
concert										
CountTR.	2.4	0.9	4.4	7.7	10.5	**23.0**	**13.5**	42.5	**16.5**	40.4
TLogic	0.9	0.3	1.5	3.8	4.3	11.3	6.5	22.9	9.3	21.6

Table 3: Test results per direction, all values show %: predicting artists *(?, performs at festival, F, 2025)*, predicting festivals *(A, performs at festival, ?, 2025)*. Best results are **bold**, second best are underlined.

5.3 Results and Discussion

Table 3 reports results per direction: predicting artists *(?, performs at festival, F, 2025)* for given festivals F, predicting festivals *(A, performs at festival, ?, 2025)* for given artists A, and aggregated over both directions. RE-GCN, CEN, and CognTKE consistently produced CUDA out-of-memory errors on the full `concert` dataset even after reducing embedding dimension and batch size, and are therefore only evaluated on $\texttt{concert}_\texttt{p}$.

In general, scores are higher for predicting festivals than for predicting artists. This is expected, as there are significantly less festivals than there are artists, and the set of suitable festivals for a given artist is more constrained, e.g., by genre compatibility, than vice versa. The best-performing model differs by direction: CognTKE achieves the highest scores for artist prediction, while CountRuCoLa performs best for festival prediction.

Does information beyond performance history help? CountRuCoLa and TLogic show slight improvements on `concert` over $\texttt{concert}_\texttt{p}$ for artist prediction, but no consistent pattern emerges for festival prediction. The enriched data appears to provide value in some cases, but also to introduce noise that current models struggle to cope with.

Is there a benefit in using specialized TKG

methods over out-of-the-box LLMs? Comparing TKG methods to LLMs applied zero-shot, TKG methods show a clear and consistent advantage across both prediction directions. For artist prediction, the best TKG model CognTKE achieves a P_{N10} of 11.9%, compared to only 1.7% for the strongest LLM (Llama 3.3 70B). This gap holds across all metrics and both prediction directions. This suggests that the implicit knowledge of a general-purpose LLM cannot replace specialized forecasting methods on a domain-specific TKG.

Are the results sufficient for real-world application? Artist prediction scores (P_R values of 8-12%) are too low to serve as standalone recommendations, but festival prediction is more encouraging: around 16% of predicted festivals are correct under P_R, and 40% within the top-10 predictions. A realistic deployment could provide model outputs as a ranked shortlist for human review. The limited gain from $\texttt{concert}_\texttt{p}$ to `concert` further suggests that current TKG models do not yet capture the full complexity of festival booking. Closing this gap represents a clear direction for future work.

We present additional analyses, including whether prediction quality depends on the activity level of the query subject and the potential benefit of auxiliary shortcut relations, in Appendix A.4.

5.4 Discussion of Model Strengths, Limitations, and Identified Failure Modes

LLMs: All four models exhibit a strong popularity bias. They frequently predict globally prominent artists rather than artists relevant to the German festival landscape. For example, GPT-4o-mini names Billie Eilish in the top 3 predictions in 47.1% of queries, even though she did not perform at a German festival in 2025. In addition, LLM-generated rankings contain duplicate and near-duplicate entries (e.g., Llama 3.1 8B predicts Robin Schulz an average of 23 times per query across 24 queries), and show autoregressive effects, where consecutive predictions share properties, such as artist gender or genre, rather than reflecting independent judgments. Detailed examples are provided in Appendix A.5.1.

Embedding Based Methods: All three methods encounter out-of-memory errors on the full `concert` dataset, likely due to large snapshots and number of entities. RE-GCN and CEN underperform the Recurrency Baseline, consistent with previous observations (Gastinger et al., 2024b,a). CognTKE shows more promise: it achieves competitive scores even when trained solely on $\texttt{concert}_\texttt{p}$, suggesting that research into improved scalability is a worthwhile direction.

Rule-Based Methods: CountTRuCoLa is restricted to rules of length 1, thus cannot capture multi-hop dependencies. Despite this, it uncovers meaningful connections that reflect plausible real-world connections between artists, venues, festivals, and record labels. E.g., performing at Mayfair Ballroom is predictive of playing at Rock am Ring, and holding a contract with Spinefarm Records is predictive of playing at Wacken Open Air. TLogic, in contrast, performs substantially below other TKG methods, despite being competitive on standard benchmarks (Gastinger et al., 2023). It only learns cyclic rules without constants, meaning that rule bodies must begin and end with the same entities as rule heads. This prevents it from learning rules with specific entities, such as artists who perform at Hurricane will likely perform at Southside Festival. ($< 0.2\%$ relative MRR improvement).

6 Future Work

Our work opens several directions for future research. The current TKG could be enriched in multiple ways: Unstructured news data could capture artist availability (tour cancellations, recording schedules). Acoustic attributes could help capture stylistic similarity. Economic factors such as venue capacities and booking budgets reflect practical constraints. Moreover, finer-grained geographic information, such as inter-city distances, could improve predictions given that German festival organizers place great importance on including local artists (Sommer et al., 2025); future work could account for this via spatial embeddings or proximity-based scores. More broadly, the scalability limitations we observed motivate research into more efficient TKG reasoning approaches suited to the large real-world event data. Finally, future work could explore other real-world predictions, such as partial lineup queries (predicting remaining performers given a partially confirmed lineup) and partial schedule queries (predicting additional bookings for a given artist). Since partial lineup queries expose some ground-truth information about the target festival, they may substantially change task difficulty; understanding this effect is a useful direction for future work. Both settings would require adapted evaluation protocols.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data.

Wei Chen, Yuting Wu, Shuhan Wu, Zhiyu Zhang, Mengqi Liao, Youfang Lin, and Huaiyu Wan. 2025. Cogntke: A cognitive temporal knowledge extrapolation framework. In *39th Conference on Artificial Intelligence (AAAI)*, pages 14815–14823.

Zifeng Ding, Heling Cai, Jingpei Wu, Yunpu Ma, Ruotong Liao, Bo Xiong, and Volker Tresp. 2024. zrllm: Zero-shot relational learning on temporal knowledge graphs with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1877–1895.

Julia Gastinger, Shenyang Huang, Michael Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, Reihaneh Rabbany, and Guillaume Rabusseau. 2024a. TGB 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. In *38th Conference on Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.

Julia Gastinger, Christian Meilicke, Federico Errica, Timo Sztyler, Anett Schuelke, and Heiner Stuckenschmidt. 2024b. History repeats itself: A baseline for temporal knowledge graph forecasting. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*.

Julia Gastinger, Christian Meilicke, and Heiner Stuckenschmidt. 2025. Counttrucola: Rule confidence learning for temporal knowledge graph forecasting. *arXiv preprint arXiv:2509.09474*.

Julia Gastinger, Timo Sztyler, Lokesh Sharma, Anett Schuelke, and Heiner Stuckenschmidt. 2023. Comparing apples and oranges? On the evaluation of methods for temporal knowledge graph forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 533–549.

Zhen Han. 2022. *Relational learning on temporal knowledge graphs*. Phd thesis, Ludwig–Maximilians–University, Munich, Germany.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2021a. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *9th International Conference on Learning Representations (ICLR)*.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021b. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8352–8364.

R Scott Hiller. 2016. The importance of quality: How music festivals achieved commercial success. *Journal of Cultural Economics*, 40(3):309–334.

ibnzterrell. 2026. Meta-llama-3.3-70b-instruct-awq-int4. https://huggingface.co/ibnzterrell/Meta-Llama-3.3-70B-Instruct-AWQ-INT4. Hugging Face model repository. Quantized 4-bit version of Meta Llama 3.3 70B Instruct model using AWQ INT4.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, pages 1–49. Citeseer.

Ningyuan Li, E Haihong, Tianyu Yao, Tianyi Hu, Yuhan Li, Haoran Luo, Meina Song, and Yifan Zhu. 2025. Infer: A neural-symbolic model for extrapolation reasoning on temporal knowledge graph. In *19th International Conference on Learning Representations (ICLR)*.

Yujia Li, Shiliang Sun, and Jing Zhao. 2022a. TiRGN: Time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2152–2158.

Zixuan Li, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. 2022b. Complex evolutional pattern learning for temporal knowledge graph reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2: Short Papers*, pages 290–296.

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal knowledge graph reasoning based on evolutional representation learning. In *The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Ruotong Liao, Xu Jia, Yangzhe Li, Yunpu Ma, and Volker Tresp. 2024. Gentkg: Generative forecasting on temporal knowledge graph with large language models. In *Findings of the Association for Computational Linguistics: NAACL*, pages 4303–4317.

Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. 2022. TLogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *36th Conference on Artificial Intelligence (AAAI)*, pages 4120–4127.

Live Nation Entertainment, Inc. 2026. setlist.fm, the setlist wiki.

Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. 2015. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Evaluation in information retrieval. In *Introduction to Information Retrieval*, chapter 8, pages 151–175. Cambridge University Press.

Meta. 2024. Meta-llama-3.1-8b. Hugging Face model repository. `https://huggingface.co/meta-llama/Meta-Llama-3.1-8B`.

MetaBrainz Foundation. 2026. Musicbrainz.

Mistral AI. 2023. Mixtral of experts (8x7b).

OpenAI. 2024. Gpt-4o mini.

Grace I. Scarborough, Benjamin E. Bagozzi, Andreas Beger, John Berrie, Andrew Halterman, Philip A. Schrodt, and Jevon Spivey. 2023. POLECAT Weekly Data.

Andrew Shilliday, Jennifer Lautenschlager, et al. 2012. Data for a worldwide icews and ongoing research. *Advances in Design for Cross-Cultural Activities*, 455.

Michael Sommer, Mathias Lorch, Lisa Andersohn, Karsten Schölermann, Stephan Schulmeistrat, Timo Varelmann, and Nelly Welskop. 2025. Festivalstudie. Musikfestivals in Deutschland: Vielfalt, Strukturen und Herausforderungen. Editorial team: Michael Sommer and Mathias Lorch (IfD Allensbach).

Haohai Sun, Jialun Zhong, Yunpu Ma, Zhen Han, and Kun He. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8306–8319.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10).

Yi Xu, Junjie Ou, Hui Xu, and Luoyi Fu. 2023. Temporal knowledge graph reasoning with historical contrastive learning. In *37th Conference on Artificial Intelligence (AAAI)*, pages 4765–4773.

Cunchao Zhu, Muhao Chen, Changjun Fan, Guangquan Cheng, and Yan Zhang. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *35th Conference on Artificial Intelligence (AAAI)*, pages 4732–4740.

A Appendix

A.1 Assumptions and Limitations

Constructing and utilizing this TKG for festival lineup prediction introduces several challenges and is based on certain assumptions:

Assumption of Completeness: The forecasting models and evaluation protocols operate under the assumption that the TKG is sufficiently complete regarding the festival lineups. Both setlist.fm and MusicBrainz are crowdsourced and thus are prone to coverage gaps. We performed sanity checks by checking a sample of lineups against official festival websites. These checks revealed no major omission.

Dataset Scope: The current schema is limited to performances, releases, metadata, and label information. It excluded additional contextual information that human experts might leverage, such as unstructured data from news and announcements regarding, e.g., artist breaks, recording schedules, or tour cancellations, music-specific details like instrumentation or acoustic characteristics of the music, and economic factors like ticket prices and venue capacities. Incorporating such multimodal data could significantly enhance predictive performance but lies outside the scope of this work.

Temporal Dynamics and Announcement Waves: Festival lineups are typically not released at once, but in waves over several months leading up to the event. This dynamic release schedule introduces two challenges for our evaluation: The exact moment when a festival's lineup is considered complete is ambiguous on setlist.fm. For this reason, we excluded the year 2026 from our evaluation, even though partial lineups are available at the time of writing. And second, comparing against models with access to real-time web search, e.g., Large Language Models (LLMs), is complicated by the fluid information availability. For instance, asking an LLM at the end of 2025 to forecast lineups for 2026 festivals risks test set leakage, as partial lineup data may already exist online.

Scalability and Memory Constraints: As illustrated in Figure 2, the resulting TKG is large when compared to standard datasets in terms of node and quadruple count, and especially snapshot size. This presents a practical problem: many existing TKG forecasting models encounter out-of-memory errors when processing TKG of this size (Gastinger et al., 2024a). We already restricted the dataset to festivals located in Germany. Expanding the TKG to a global scale would increase its size and likely make current state-of-the-art models inapplicable without architectural modifications or sampling strategies. While testing on a global festival TKG was beyond the scope of this work, we provide the code to enable the construction of such expanded TKGs for future research.

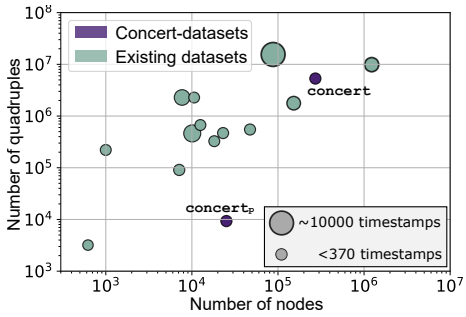


Figure 2: Comparison of `concert` dataset statistics to existing TKG datasets. Figure adapted from (Gastinger et al., 2024a).

A.2 Additional Dataset Details

Figure 2 contextualizes the size of our dataset by comparing it to datasets commonly used for evaluating TKG forecasting methods (Gastinger et al., 2024a). The dataset `concert` contains a comparatively high number of nodes and quadruples. At the same time, despite its overall size, it comprises relatively few timestamps. This implies that individual temporal snapshots are larger than in most existing datasets. `concert`$_p$ exhibits a comparatively small number of quadruples since it is a subset containing only the quadruples of one relation.

Figure 3 illustrates additional dataset statistics. Figures 3(a) and (b) depict the number of quadruples over time. Both datasets show steady growth from 1971 to 2019, followed by a sharp drop during the COVID-19 pandemic (2020–2021), which is particularly pronounced in `concert`$_p$. Quadruple counts recover to near pre-pandemic levels by 2023.

The distribution of relations (Figure 3(c)) in `concert` reveals that concert performances dominate the TKG with 57% of all quadruples (relation *performs concert at*). The high occurrence of the *has type* relation is caused by our temporal encoding strategy: as detailed in Section 4.3, this information is replicated for every year of an artist's active period, inflating the count of these specific triples.

Regarding the distribution of node types (Fig-

ure 3(d)), nodes that represent releases constitute the largest part. This is intuitive: while a single venue hosts a multitude of artists, a single artist may release many albums or EPs, each represented as a distinct node. Figure 3(e) shows the 12 most common release genres: Rock, Electronic, and Pop are the dominant categories, reflecting the mainstream focus of the festivals in the dataset. Figure 3(f) visualizes the geographic distribution of artist areas: although all festivals take place in Germany, the artists are predominantly from Europe and North America, with Germany, the United States, and the United Kingdom contributing the largest shares.

A.2.1 Selected Dataset Facts

Beyond aggregate statistics, the dataset enables extracting interesting facts regarding artists and festivals, for example:

- The three largest festival editions in terms of artist line-up size are Reeperbahn Festival 2024 (390 artists), Wacken Open Air 2023 (228 artists), and Wave-Gotik-Treffen 2024 (185 artists).
- The artists who performed at the highest number of festivals in Germany overall are Donots (63 festivals), Emil Bulls (58 festivals), and Itchy (56 festivals).
- The artists who performed at the highest number of festivals in a single year in Germany are Kraftklub (2019; 19 festivals), AnnenMayKantereit (2014; 18 festivals), and Gurr (2016; 17 festivals).
- The artists with the longest festival history in Germany (highest number of festival years) are Guru Guru (31 years; first: 1971, last: 2025), Subway to Sally (30 years; first: 1992, last: 2025), and Project Pitchfork (30 years; first: 1993, last: 2025).
- The venues hosting the highest number of distinct artists are Grote Zaal (2420 artists), Markthalle Hamburg (2310 artists), and Ancienne Belgique (2225 artists).
- The most common origin areas of artists are London (457 artists), Berlin (323 artists), and Los Angeles (267 artists).
- The countries with the highest number of artists (relation *has area*) are Germany (3424 artists), the United States (2877 artists), and the United Kingdom (1799 artists).
- Among artists with available type information (relation *has type*), 73.2% are classified as Group and 26.7% as Person. Additional types (in negligible numbers) include Orchestra, Choir, Other, and Character.

A.3 Experimental Details

A.3.1 Details on TKG method Experiments

Hardware Information: The methods RE-GCN, CEN, and CognTKE require execution on a GPU. We run them on an NVIDIA RTX 6000 Ada (48GB VRAM) with 500GB system RAM. The other experiments were run on a SLURM-managed CPU cluster. Nodes featured Intel Xeon (E5-2640 v2/v3/v4, Silver 4114) and AMD EPYC (7413, 7713P, 9474F) processors, with up to 96 cores and 1.5 TB RAM per node. We restricted parallelization to a maximum of 20 concurrent threads and limited available RAM to 500 GB.

For each method, we select the hyperparameters from the ranges specified in the original papers and code repositories. We conduct grid search on the validation set. Table 4 reports the values of selected hyperparameters for each method.

A.3.2 Details on LLM Experiments

LLM Prompts: The prompts used for the LLMs were evaluated and tuned on an iterative approach, generating two prompts for artists and festivals repeatedly[2].

For receiving artists playing at a festival, the LLMs were prompted like in Figure 4.

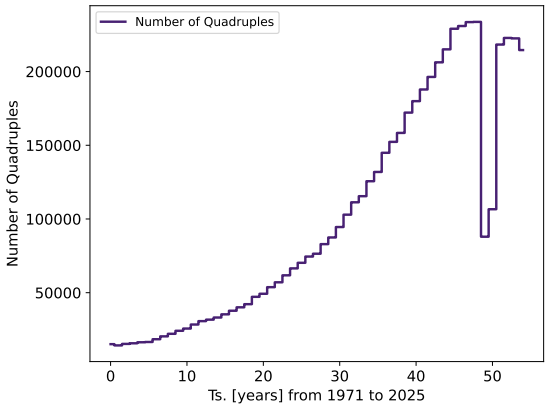
For receiving festivals where an artist will play, the LLMs were prompted like in Figure 5.

A.4 Additional Experiments and Results

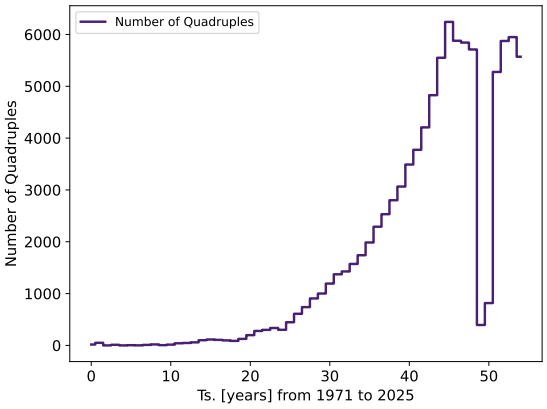
A.4.1 Results by Subject Activity Level

To investigate whether prediction quality depends on the activity level of the query subject, we analyze P_R across the number of distinct festivals an artist performed in, and the number of distinct artists a festival hosted (Figure 6). A clear positive trend is visible in both directions: prediction quality increases with activity level. The Recurrency Baseline performs comparably to other methods for infrequent subjects but falls increasingly behind as activity grows. This suggests that for infrequent query subjects, all methods are limited to simple distribution heuristics, and the more expressive models offer little advantage. For more active artists and festivals, richer temporal and relational

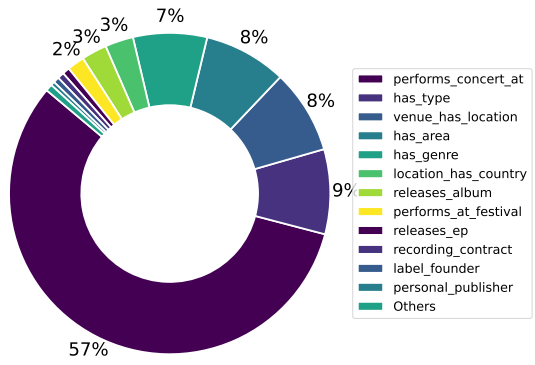
[2]The prompts, as well as input and evaluation code are also available on GitHub: https://github.com/JuliaGast/festival-lineup-tkg/tree/main/forecasting/llm.



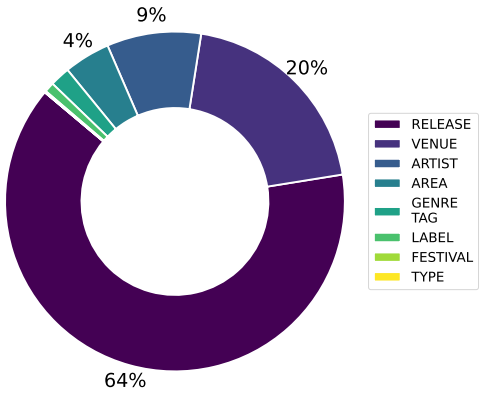
(a) concert



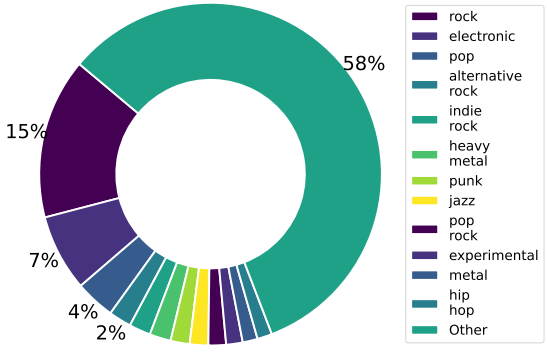
(b) $\texttt{concert}_{\text{p}}$



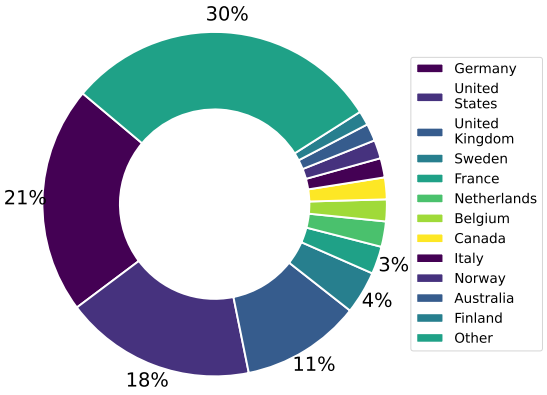
(c) Relations



(d) Node types



(e) Release Genres



(f) Artist Areas

Figure 3: Top: Number of quadruples over time for `concert` and $\texttt{concert}_{\text{p}}$. Bottom: Pie charts representing distribution of relations, node types, release genres, and artist areas for`concert`.

Method	Hyperparameters selected
RE-GCN	n-layers $= 2$, train-history-len$= 4$, test-history-len$= 4$
CEN	start-history-len$= 1$ dropout$= 0.2$ n-layers$= 2$
CognTKE	n-layers $= 3$, window-size $= 15$, dropout=0.25 (default hyperparameters, since no ranges or tuning procedure were specified, and results on all datasets in original paper were reported on the same hyperparameter values)
CountTRuCoLa	$\mathcal{P} = 30$, $\mathcal{P}_{f\text{-rules}} = 0$, $\mathcal{C} = True$, $\mathcal{W} = 20$, $\mathcal{M} = 50$, $\mathcal{Z} = 0.01$, $\mathcal{H} = 10$, $\mathcal{D} = 0.4$, $\mathcal{C}_{Xcount} = 3$; for $\texttt{concert}_{\texttt{with shortcuts}}$ we set $\mathcal{C}_{Xcount} = 10$, due to larger TKG size.
TLogic	num-walks$= 10$, transition-distr$= exp$, window$= 30$, top-k$= 10$, $\alpha = 1$, $\lambda = 0.01$, rule-length$= [1, 2, 3]$
Rec.	No Hyperparameters

Table 4: Hyperparameter Values selected for each method.

```
You are estimating likely performers for the \{name\} \{year\}
in \{city\}, \{country\}.
Task:
Generate a ranked list of 210 artists most likely to perform.
Output requirements:
- Provide exactly 210 artist names
- Rank them from 1 (most likely) to 210 (least likely)
- Output ONLY the ranked list
- Format: each line should be "1. Artist Name",
  "2. Artist Name", etc.
- Do NOT include explanations, notes, or extra text
If uncertain, still provide your best estimate based on
the criteria above.
```

Figure 4: LLM Prompts for receiving artists playing at a festival.

```
You are estimating which festivals in Germany an artist
named "\{name\}" is most likely to perform at in \{year\}.
Task:
Generate a ranked list of 50 German festivals
where "\{name\}" is most likely to perform in \{year\}.
Output requirements:
- Provide exactly 50 festival names
- Rank them from 1 (most likely) to 50 (least likely)
- Output ONLY the ranked list
- Format: each line should be "1. Festival Name",
  "2. Festival Name", etc.
- Do NOT include explanations or any extra text
If uncertain, still provide your best estimate based on
the criteria above.
```

Figure 5: LLM Prompts for receiving festivals where an artist will play.

patterns become available, which specialized TKG methods are better able to exploit. Finally, for less active artists, CountRuCoLa benefits more from the full `concert` dataset. This fits the intuition that additional information like venue patterns adds value when performance histories are sparse.

A.5 Addition of shortcut relations

To address the challenge that multi-hop inference across distant nodes may pose for certain TKG models, in this study, we introduce auxiliary shortcut relations under the category *Additionally Added Shortcuts*. These relations do not introduce new information but provide direct edges between artists and attributes (e.g., countries where they perform, genres of their releases) thereby offering an alternative path to information that would otherwise require traversing multiple hops. While this creates redundancy and increases graph size, it may benefit models with limited hop capacity.

We treat shortcut relations as optional input and conduct experiments under two configurations: with and without shortcuts. Table 5 details the schema extensions, and Table 6 reports dataset statistics for $\texttt{concert}_{\texttt{shortcuts}}$ alongside the default `concert` and the reduced $\texttt{concert}_{\texttt{p}}$. While `concert` and $\texttt{concert}_{\texttt{shortcuts}}$ share identical node counts, due to the additional relations that we injected in $\texttt{concert}_{\texttt{shortcuts}}$, the number of quadruples increases. Importantly, the prediction task as well as the ground truth, are consistent across both variant: the *performs at festival* quadruples exist unchanged in all three datasets, ensuring that test-set results are directly comparable.

Table 7 reports results for CountTRuCoLa and TLogic on both dataset variants, $\texttt{concert}_{\texttt{shortcuts}}$ and `concert`. The remaining methods could not be evaluated due to out-of-memory errors. No consistent improvement is observed when adding shortcut relations, and we therefore conclude that, at least under the current model capabilities, they do not provide a measurable benefit.

A.5.1 LLM Ranking Quality

Detailed Examples on LLM failures: All evaluated LLMs exhibit a tendency to favor highly popular artists regardless of their relevance to the German festival landscape. Llama 3.1 8B predicts Taylor Swift among the top three candidates in 23.9% of queries, Mixtral 8×7B in 36.1%, and GPT-4o-mini predicts Billie Eilish in 47.1% of queries, although neither artist performed at a German festival in 2025. Llama 3.3 70B instead tends to predict prominent German artists, such as Rammstein (48.3%) and Die Toten Hosen (23.5%), but these predictions are almost always incorrect; only Die Toten Hosen appearing at Jamel rockt den Förster in 2025 was a valid prediction.

The generated rankings also frequently contain duplicate and near-duplicate predictions. For example, Llama 3.1 8B predicts Robin Schulz an average of 23 times per query across 24 queries. Mixtral 8×7B produces near-duplicates such as "Taylor Swift" and "Taylor Swift & Ed Sheeran" within the same ranking.

Ranking Example (Autoregressive Bias): LLM-generated rankings exhibit autoregressive bias. Since predictions are generated sequentially, previous tokens can influence subsequent outputs, leading to rankings where neighboring predictions share properties such as artist gender or genre rather than representing independent candidate evaluations.

In Figure 7, we show a ranking produced by Llama 3.3 70B when asking who will perform at the *Keep It True Festival* in 2025. Female artists and female-fronted bands are marked in green. Rather than being spread throughout the ranking as knowledge of the festival would suggest, these acts form a contiguous block. Artist gender is not a meaningful criterion for ranking bands at a traditional heavy metal festival, so the observed clustering exemplifies the auto-regressive bias described in Section 5.4: once the model generates one female-fronted act, subsequent token probabilities shift toward similar acts.

A.5.2 Effect of LLM Knowledge Cutoff

Since the knowledge cutoff of the evaluated LLMs falls around late 2023, queries about 2025 festival lineups may fall outside their reliable knowledge window. To assess whether this introduces a systematic bias, we compare model performance across three years, 2023, 2024, and 2025, using a random subset of 50 artists and 50 festivals sampled from quadruples occurring in all three years. To limit expenses, these tests were only conducted on the three open-source models.

Figure 8 (top) reports Mean R-Precision per year and model. CountTRuCoLa consistently outperforms all LLM-based approaches across all years, consistent with the main results in Section 5.3. To analyze year-to-year differences, we compute the

	Domain	Relation	Range
Additionally Added Shortcuts	ARTIST	releases album of genre	GENRE-TAG
	ARTIST	releases ep of genre	GENRE-TAG
	ARTIST	performs festival in country	AREA
	ARTIST	performs festival in location	AREA
	ARTIST	performs concert in country	AREA
	ARTIST	performs concert in location	AREA

Table 5: Data schema, extended triples, for the temporal KG. Each temporal triple also comes with a timestamp.

Dataset	#N	#R	#Train	#Valid	#Test	Time Int.	#Tr/Val/ Te TS	DRec [%]	Rec [%]
concert	278.936	23	5.068.265	222.507	214.578	1 y.	53/1/1	26.9	44.2
$\texttt{concert}_{\texttt{shortcuts}}$	278.936	29	8.927.809	423.314	409.911	1 y.	53/1/1	25.0	53.1
$\texttt{concert}_{\texttt{p}}$	25.399	1	83.889	5.949	5.570	1 y.	51/1/1	2.9	29.4

Table 6: Dataset statistics for all three dataset variants: The number of nodes (#N), relations (#R), train, valid, and test quadruples, the time interval, number of train/valid/test timestamps, Direct Recurrency Degree (DRec), and Recurrency Degree (Rec).

	predicting artists					predicting festivals					both directions				
	MRR	H1	H10	P_R	P_{N10}	MRR	H1	H10	P_R	P_{N10}	MRR	H1	H10	P_R	P_{N10}
concert															
CountTRuCoLa	2.4	0.9	4.4	7.7	**10.5**	**23.0**	13.5	**42.5**	16.5	**40.4**	**12.7**	7.2	**23.5**	16.0	**38.5**
TLogic	0.9	0.3	1.5	3.8	4.3	11.3	6.5	22.9	9.3	21.6	6.0	3.2	12.2	8.7	20.6
$\texttt{concert}_{\texttt{with shortcuts}}$															
CountTRuCoLa	**2.5**	**1.0**	**4.5**	**8.2**	**10.5**	22.5	**13.6**	40.9	**16.7**	39.0	12.5	**7.3**	22.7	**16.2**	37.3
TLogic	0.9	0.1	1.5	3.1	2.9	11.5	6.6	22.8	9.5	21.4	6.1	3.3	12.3	8.8	20.4

Table 7: Test results per direction: predicting artists *(?, performs at festival, F, 2025)*, predicting festivals *(A, performs at festival, ?, 2025)*, and predicting both directions, i.e. aggregated over all test queries. Best results are **bold**. Comparing the default concert dataset to the extended $\texttt{concert}_{\texttt{with shortcuts}}$ dataset version.

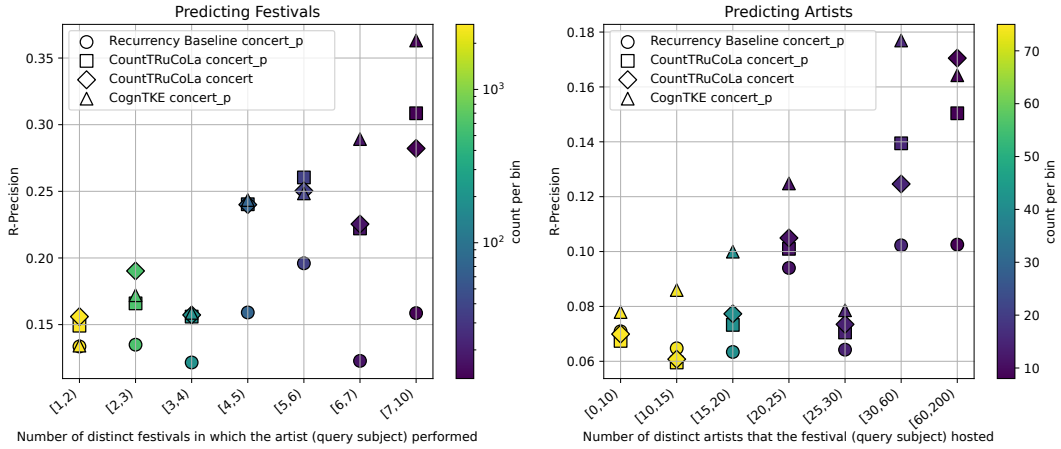


Figure 6: Mean P_R binned by artist activity (number of distinct festivals per year, left) and festival activity (number of distinct artists hosted per year, right) for the four best-performing methods. Colors indicate counts per bin.

relative deviation of each model's per-year score from its three-year mean, shown in Figure 8 (bottom). The deviations are comparable in magnitude across LLMs and CountTRuCoLa alike: Llama 3.3 70B, Mixtral, and CountTRuCoLa peak in 2024, while Llama 3.1 8B reaches its lowest score in that year. Crucially, no model exhibits systematically stronger performance in 2023 or 2024 relative to 2025, and the LLMs do not show unusually large temporal fluctuations compared to CountTRuCoLa.

We find no evidence that the knowledge cutoff substantially degrades 2025 performance, and therefore proceed with 2025 data in all subsequent experiments.

1. Saxon
2. Judas Priest
3. Iron Maiden
4. Scorpions
5. Helloween
6. Blind Guardian
7. Accept
8. U.D.O.
9. Sabaton
10. Powerwolf
11. Rhapsody of Fire
12. Hammerfall
13. Stratovarius
14. Sonata Arctica
15. Dragonforce
16. Alestorm
17. Korpiklaani
18. Eluveitie
19. Ensiferum
20. Finntroll
21. Turisas
22. Sabaton
23. Amon Amarth
24. Children of Bodom
25. In Flame s
26. Dark Tranquillity
27. At the Gates
28. Arch Enemy
29. Nightwish
30. Epica
31. Within Temptation
32. Delain
33. Leaves' Eyes
34. Xandria
35. Lacuna Coil
36. Evanescence
37. Kamelot
38. Symphony X
39. Evergrey
40. Threshold

Figure 7: A ranking produced by Llama 3.3 70B when asking who will perform at the *Keep It True Festival* in 2025. Female artists and female-fronted bands are marked in green.

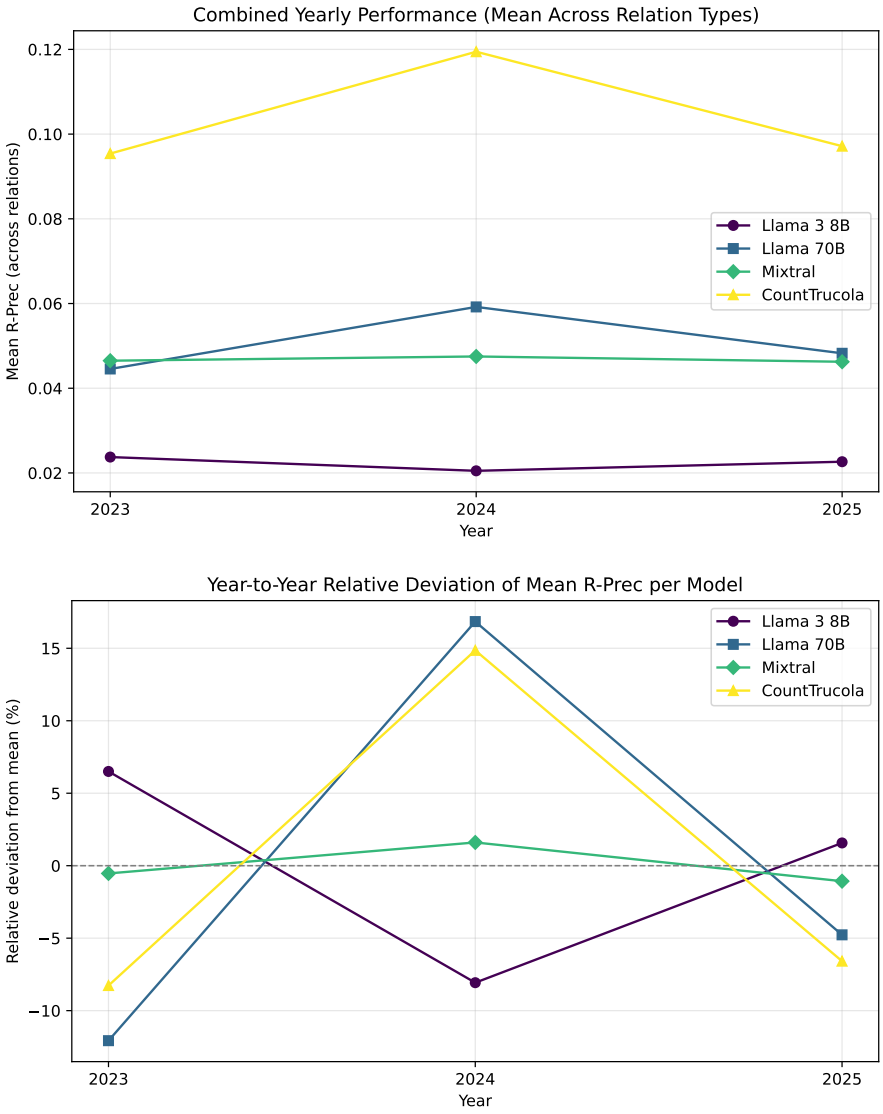


Figure 8: Per-year performance on Mean R-Precision (top) and relative deviation from each model's three-year mean (bottom), evaluated on a subset of 50 artists and 50 festivals appearing in all three years.

The Canonical Order Problem: When Large Language Models Are Unreliable Knowledge Bases for Multi-Valued Relations

Timo Pierre Schrader[1] **Annemarie Friedrich**[1] **Simon Razniewski**[2] **Lukas Lange**[3]
[1]University of Augsburg, Augsburg, Germany
[2]ScaDS.AI & TU Dresden, Dresden, Germany
[3]Bosch Center for Artificial Intelligence, Renningen, Germany
lukas.lange@de.bosch.com

Abstract

Large language models (LLMs) are increasingly used as knowledge bases (KBs) due to the vast amount of knowledge they acquire during pre-training. While many works focus on extracting single relational triples, most real-world relations are multi-valued and require generating sets of entities.

In this paper, we investigate how LLMs represent and generate multi-valued relations. We identify the **canonical order problem**: The probabilistic distributions inside LLMs organize many multi-valued relations according to a canonical ordering (e.g., alphabetical or chronological). Through mechanistic analysis, we show that set generation in LLMs can be thought of in terms of three phases: **(1)** retrieval of candidate entities, **(2)** internal sorting, and **(3)** selection of the next element. As a result, prompts aiming to construct KBs that deviate from this internal canonical ordering lead to a markedly reduced reliability of LLMs when aiming to generate complete sets for multi-valued relations.

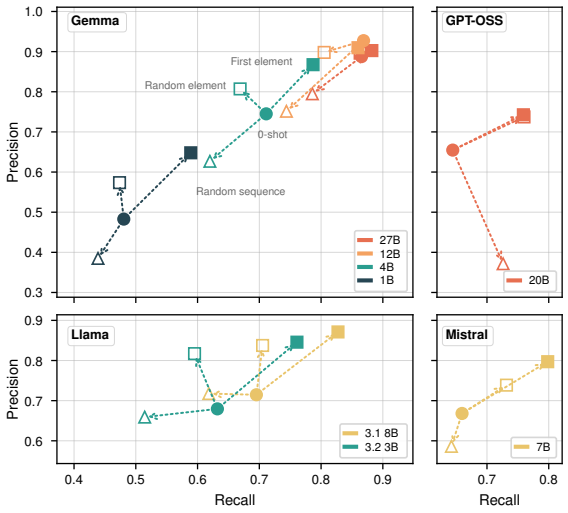


Figure 1: Precision-recall curves for LLM set generation across 37 topics. We compare four distinct settings: • without any examples; ■ with the first element in canonical order; □ with a single random element; and △ with randomly shuffled elements. Providing the first element as an example always achieves the best results.

1 Introduction

Due to their pre-training on huge amounts of text, large language models (LLMs) possess broad knowledge implicitly stored in their parameters. This renders LLMs particularly interesting for the extraction of structured knowledge to construct knowledge bases (KBs, Hu et al., 2024; Pinto et al., 2025). Most existing work focuses on single-valued relations (Petroni et al., 2019; Jiang et al., 2020; Singhania et al., 2023). In practice, much relational knowledge is organized into sets such as "all Formula 1 world champions," "all official EU languages," or "all Turing Award winners." To extract this knowledge from an LLM, the model has to generate a set of entities. In practice, this leads to two main issues: (1) elements are *omitted* during generation (*Recall problem*) and (2) the generation of additional wrong elements (*Precision problem*).

Despite the practical importance of multi-valued relations, little is known about how LLM parameters store such information, or what happens when we attempt to retrieve it via prompting and generation. In this paper, we study the internal mechanisms and challenges when extracting this type of relational knowledge from LLMs. Our findings reveal previously overlooked limitations of using LLMs for KB construction: whether or not they succeed in retrieving multi-valued relations depends strongly on the examples provided in the prompt.

We identify the **canonical order problem**: LLMs internally organize multi-valued relations according to a canonical order (e.g., numerical, alphabetical, or chronological), which causes poor reliability when the generation process deviates from this order. This finding shows that in this context, LLMs do not "store" relational information consistently but rather that their output depends on whether prompts reflect canonical orders observed during pretraining. As an example, we consider retrieving all 16 states of Germany. If we ask a model such as Gemma-3 12B to come up with all 16 states, it will most likely generate all of them in alphabetical order. However, if we ask it to complete a random subset of 9 states, it for example tends to forget *Bremen* in its final

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 45–55
October 28, 2026 ©2026 Association for Computational Linguistics

output. Figure 1 illustrates this effect across 37 topics and multiple model families. We compare four prompting strategies: no example at all, the first element in canonical order, a single randomly chosen element, and a set of randomly shuffled elements. In all cases, providing the first element in canonical order yields the most complete and accurate sets, while forcing the model to continue from shuffled elements leads to substantially more omissions and errors.

To better understand this model behavior, we perform a mechanistic analysis. We analyze the autoregressive generation process by inspecting the outputs of each Transformer block and how expected entities behave in the hidden states. Our mechanistic analysis suggests that activation in earlier Transformer layers reflect all set entities equally (reflecting the overall topic similar to a **retrieval step**) and that the following layers increasingly reflect canonical orderings (**sorting**), which becomes problematic when prompts propose example set entities that are not at the beginning of the canonical ordering. This implies that multi-valued knowledge extraction from LLMs is more reliable when they do not have to deviate from a canonical order. We argue that the observed behavior limits the reliability of zero-shot LLMs as KBs, since the correctness of generated sets depends not only on the stored facts but also on the generation order.

2 Related Work

In this section, we review related work on using LLMs for KB construction, using LLMs in combination with structured knowledge and multi-valued relations, ordering problems in LLMs, and approaches to mechanistic analyses of LLMs.

Language Models for KB Construction The knowledge implicitly stored in language models is widely used for constructing KBs (Petroni et al., 2019; AlKhamissi et al., 2022). GPTKB (Hu et al., 2024) extracts 105 million relations from GPT-4o mini, greatly exceeding the number of entries in well-known KBs such as Wikidata (Vrandečić and Krötzsch, 2014) and YAGO (Suchanek et al., 2007). A multi-turn scheme for extracting relational triples is introduced by Zhang and Soh (2024), where free-form LLM-based extraction is followed by canonicalization of generated relations into a general schema.

Another branch of KB construction uses the language understanding capabilities of LLMs to extract relational triples from documents. A pipeline for extracting relational triples from text given a priori retrieved context from an existing KB is proposed by Papaluca et al. (2024). Applications in materials science demonstrate how LLMs can extract knowledge from tabular data for KG construction (Dreger et al., 2025). Earlier approaches leverage encoder-based language models to treat relation extraction as a classification task (Wei et al., 2020).

We position our work as an analysis of the intermediate step of knowledge elicitation from LLMs before actually building up a KB. Although the generated sets of tail entities could be used for constructing a KB, we aim to understand how LLMs internally represent and generate relational knowledge.

LLMs and Multi-Valued Relations There are two main facets when dealing with multi-valued relations in the context of LLMs: *extraction* and *editing*. On the extraction side, Singhania et al. (2023) probe BERT encoders (Devlin et al., 2019) using masked language modeling to generate candidate entities. Different heuristics for selecting candidate sets are explored, such as top-k selection or probability thresholds derived from model logits. In contrast, our work focuses on analyzing set generation in autoregressive decoder-based models rather than proposing selection heuristics. Decoder-based LLMs benefit from broader general-purpose knowledge due to next-token prediction objectives and typically larger parameter counts. On the editing side, encoder-decoder models are fine-tuned to generate either individual instances or complete sets (Nagasawa et al., 2023). While high output accuracy can be achieved, this often comes at the cost of reduced generalizability.

LLMs and Structured Knowledge Research in this area investigates how structured knowledge can be processed, generated, or integrated with LLMs. This includes tabular data (Agarwal et al., 2025; Zhou et al., 2025a; Liu et al., 2024b), KGs (Zhou et al., 2025b; Regino and dos Reis, 2025; Zhu et al., 2025), databases (Qin et al., 2024; Li et al., 2023), natural language documents via retrieval-augmented generation (Wahidur et al., 2025; Arslan et al., 2024), and ontologies (Mai et al., 2024; Liu et al., 2025). Our work focuses on structured knowledge in the form of lists, which may be stored in documents or generated directly as responses.

Influence of Orderings in Language Models The ordering of few-shot demonstrations has been shown to strongly affect the accuracy of autoregressive language models, for example, the ordering in which demonstrations are presented to models in few-shot prompts (Lu et al., 2022; Zhao et al., 2021). It has also been shown that learning relational knowledge of the form $A \equiv B$ does not necessarily imply learning the reverse relation $B \equiv A$ (Berglund et al., 2024). The effect of input ordering on reasoning performance is further explored in multi-document question answering and logical reasoning tasks (Liu et al., 2024a; Chen et al., 2024). Performance degrades when relevant information appears later in the prompt but improves again when placed near the end, forming a U-shaped curve (Liu et al., 2024a). Similarly, deviations from natural premise ordering negatively impact reasoning accuracy, especially as the number of premises increases (Chen et al., 2024). While existing work focuses on input ordering, we investigate the extent to which ordering is inherent in stored

knowledge.

Mechanistic Analysis of LLMs Understanding the internal mechanisms of language models has been a long-standing goal in interpretability research. A major line of work focuses on attribution methods that assign importance scores to input features. Integrated Gradients (Sundararajan et al., 2017) is a widely used method based on axiomatic principles such as sensitivity and implementation invariance. Second-order methods have also been proposed to analyze interactions between modalities in dual encoder models (Moeller et al., 2025), such as CLIP (Radford et al., 2021).

In language models, attention analysis is commonly used to study information flow in Transformer architectures (Abnar and Zuidema, 2020; Clark et al., 2019; Jain and Wallace, 2019). In this work, we employ the logit lens technique, which projects hidden states onto the vocabulary space by applying the final normalization and language modeling head. This method has been used to analyze where factual knowledge is stored in multilingual LLMs (Wang et al., 2025), to study social biases (Prakash and Lee, 2023), and to investigate visual feature representations in vision-language models (Huo et al., 2024).

3 Theoretical Concepts

Background We define multi-valued relations as sets of entities $\mathcal{S}(t) = \{e_1, \ldots, e_n\}$ that are in an "isA" relation with head node h. Such relations can be represented as sparse or dense KGs with one node for each entity connecting it to the head as illustrated in Figure 2.

Following standard relation extraction terminology, we also call the head node the relations *subject* and the dependent node its *entities*.

Subsets and Prefixes We define *set prefixes* to be an unordered true subset $\mathcal{P}(t') \subset \mathcal{S}(t)$. Although we use the term "prefix", these subsets carry no ordering information; the term reflects their role as a starting point that the model must continue during generation. For example, a prefix for the states in Germany might be $\mathcal{P}(t') = \{\text{Bremen}, \text{Saxony}\}$. Set prefixes will be used in this paper to test how robustly LLMs can complete true subsets of the expected set while deviating from a canonical order. Starting with such prefixes reflects a realistic scenario in knowledge base curation: practitioners typically have partial knowledge about the multi-valued relations they aim to collect. For example, a knowledge engineer might already know that Bavaria is a state in Germany, and this partial information naturally suggests exploring related queries, such as "Besides Bavaria, what other states are located in Germany?" This approach is motivated by few-shot learning methods where providing explanatory examples significantly improves model performance and guides the completion process (Brown et al., 2020; JIANG et al., 2025). By starting with k prefix elements, we simulate this workflow while testing how robustly LLMs can complete true subsets despite deviating from canonical orderings. In our experiments, set prefixes enable us to analyze **(1)** how well LLMs can extend partial knowledge, and **(2)** whether the presence of non-canonical orderings in the prefix disrupts the completion of remaining set elements.

Auto-Regressive LLM Generation Process LLMs are statistical distributions that model the probability of the next token at each decoding step j:

$$LLM : V \mapsto [0, 1], \qquad \sum_{t \in V} \mathbb{P}(T_j = t) = 1$$

This mapping assumes a normalization function that is applied on the LLM logits such as softmax. The joint distribution over a sequence of tokens $(T_1, \ldots, T_n)$ can be decomposed using conditional probabilities:

$$\mathbb{P}(T_1, \ldots, T_n) = \mathbb{P}(T_1) \prod_{j=2}^{n} \mathbb{P}(T_j \mid T_1, \ldots, T_{j-1})$$

This decomposition allows us to inspect each auto-regressive decoding step j individually. Hence, we can inspect how likely each token $t \in V$ is to be generated at each step. We inspect these probabilities to learn where "missed" elements that are never accurately verbalized are ranked in the probability distribution during generation. Furthermore, due to the discrete generation process, we can determine "points of interest" at which we look deeper into the hidden states of a Transformer-based LLM, which is defined as the output of one Transformer block (attention combined with two linear layers). We provide details in Section 5.

4 Methodology

This section describes the creation of our dataset for prompting LLMs to extract sets of entities for given multi-valued relations. We then analyze the LLMs' probability distribution while generating particular set items and analyze the relationship between missed items and canonical orderings.

Dataset We first create a dataset that consists of 37 distinct multi-valued relations for which we can prompt an LLM to come up with all entities that occur as entities. These relations, alongside the canonical orderings of the entities, are listed in Table 1. We ensure that each relation has a well-defined number of entities without frequent changes over time. We leave the analysis of non-stable sets for future work. We sort the topics according to the size of the corresponding set of entities $\mid \mathcal{S}(t) \mid$ and categorize them into *small* ($\mid \mathcal{S}(t) \mid < 15$), *regular* ($15 \leq \mid \mathcal{S}(t) \mid \leq 50$), and *large* ($\mid \mathcal{S}(t) \mid > 50$). Furthermore, we assign a **canonical ordering** to each $\mathcal{S}(t)$. In this context, we refer to a canonical ordering as a commonly used way of ordering a list of entities, such as alphabetically, chronologically for temporal events, or according to other domain-specific criteria (e.g., the distance of planets from their sun). We assign

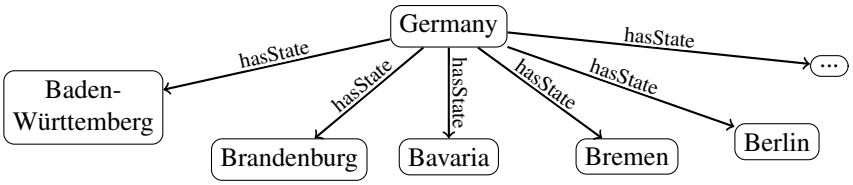


Figure 2: Example demonstrating a multi-valued relation in a KB. The dependent nodes share the same head node and are connected to it via the same relation type.

the canonical ordering based on the most common convention found in common literature or web sources (e.g., Wikipedia).

Evaluation Framework When analyzing how reliably LLMs can generate complete sets of entities for multi-valued relations, we treat them as **unordered** sets and use precision (P), recall (R), and F1 score for evaluation. To understand what happens inside an LLM during the autoregressive text generation process, we look at two distinct parts of the text generation process: **(1)** a **single forward pass** including all hidden states produced by the Transformer blocks, and **(2)** the evolution **across all autoregressive forward passes**, i.e., all steps j at which the probability $\mathbb{P}(T_j = t \mid T_{j-1} \dots T_1)$ is computed for all tokens $t \in V$ in the vocabulary. To perform a mechanistic analysis, we use the *logit lens*[1] for interpreting the outputs of hidden states in LLMs. For each hidden layer, the method takes the k-dimensional output vector, applies the model's final normalization layer followed by the language-modeling head (a linear layer) that produces the $\mid V \mid$ logits for all tokens in the vocabulary, thus allowing for a language-level interpretation of the hidden states. We can then search for all expected entities in the (sorted) logits by retrieving the position of the first subtoken of every entity.[2] When performing this search across sorted logits, we obtain a relative ranking $\hat{R}$ as well as an **absolute** rank between 0 and $\mid V \mid -1$ for each entity. We compare R, the ranking induced by the canonical ordering (i.e., sorting the entities by a fixed convention such as alphabetical or chronological), with $\hat{R}$, the ranking implied by the order in which the LLM generates the entities, using Spearman's rank correlation coefficient ρ to quantify agreement between the two orderings. For the mechanistic analysis, we treat these sets as **ordered**.

Prompt Templates The base instruction for the model for every set generation experiment and the simulated start of the model answer are as follows:

> **User:**
> *Give me a list of all* `<SUBJECT_NODE>`*. The list output should be in bullet point format with the dash sign ('-') used as bullet point style.*
>
> **Assistant:**
> *The following list contains all entities that belong to* `<SUBJECT_NODE>`*:* `\n`
> `- <OPTIONAL_PREFIX>`

For k-prefix experiments, we replace `<OPTIONAL_PREFIX>` with the bullet point list that holds all k prefix elements.

String Comparison Finding matching strings in free-text generation is difficult since exact matches are not guaranteed due to the model having the freedom of casing, adding punctuation marks and markdown markers, spacing, etc.

In order to make the set comparison between $\mathcal{S}(t)$ and $\hat{\mathcal{S}}(t)$ as robust as possible, we use the `difflib.SequenceMatcher` implementation from Python, which builds upon the algorithm originally presented in (Ratcliff et al., 1988). Our threshold is set to a strict value of 0.95. During our experimentation phase, this value has been shown to be a good trade-off. We additionally perform some pre-processing steps that remove some special signs such as quotation marks and parentheses that do not contribute to any entity in a meaningful way. The procedure is shown in Algorithm 1.

Algorithm 1 String Preprocessing

Require: Input string S_i
Ensure: Normalized output string S_o
  $S_o \leftarrow strip\_whitespace(S_i)$
  $S_o \leftarrow extract\_from\_within\_md\_indicators(S_o)$
  $S_o \leftarrow remove\_enumeration\_signs(S_o)$
  for each $delimiter$ in ["(", ":", " - ", " – ", " — ", "\n"] **do**
    $S_o \leftarrow S_o.split(delimiter)[0]$
  end for
  $S_o \leftarrow S_o.lower()$
  $S_o \leftarrow remove\_quotation\_marks(S_o)$
  $S_o \leftarrow squeeze\_multiple\_spaces\_into\_one(S_o)$

[1] https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens

[2] Some entities may share the same initial subtoken, which can lead to overlapping matches. This is a limitation of the tokenizer design and introduces ambiguities when mapping entities to vocabulary tokens. Furthermore, we search for subtokens with and without leading whitespace and select whichever is higher ranked.

Topic	Count ↓	Ordering	Size
Nobel prize categories	6	other	small
Ancient wonders	7	chronological	small
Newton spectrum colors	7	numerical	small
Crystal systems	7	other	small
Seven seas	7	other	small
Moon phases	8	chronological	small
Solar system planets	8	numerical	small
Beethoven symphonies	9	chronological	small
Poker hands	10	numerical	small
German chancellors	10	chronological	small
Zodiac signs	12	chronological	small
Countries in SA	13	alphabetical	small
Eclipse versions	13	chronological	small
German states	16	alphabetical	regular
MBTI 4-letter types	16	other	regular
Fib. numbers ($\leq$ 1000)	17	numerical	regular
Standard amino acids	20	alphabetical	regular
MS Office versions	22	chronological	regular
India: official languages	23	alphabetical	regular
EU official languages	24	alphabetical	regular
Bond films	25	chronological	regular
Primes ($\leq$ 100)	25	numerical	regular
Mammal orders	27	numerical	regular
NBA teams	30	alphabetical	regular
F1 world champions	34	chronological	regular
Shakespeare First Folio	36	chronological	regular
NVIDIA RTX GPUs	36	chronological	regular
U.S. presidents	45	chronological	regular
HTTP status codes	63	numerical	large
U.S. national parks	63	alphabetical	large
Bible books	66	chronological	large
Constellations	88	alphabetical	large
Periodic table elements	118	numerical	large
Primes ($\leq$ 1000)	168	numerical	large
Physics Nobel laureates	230	chronological	large
Stable nuclides	251	numerical	large
Catholic popes	267	chronological	large

Table 1: The 37 distinct topics in our dataset along with their number of entities $|\ \mathcal{S}(t)\ |$, type of canonical ordering, and categorization into the three set sizes.

5 Results

We start by conducting multi-valued relation generation experiments to understand how canonical orderings influence the robustness. We then provide mechanistic insights into the models to better understand why canonical orderings play a crucial role inside the LLMs. For set generation experiments in Section 5.1, we mainly focus on models from the Gemma-3 model family (Gemma-Team et al., 2025). We confirm the generalizability of our findings in an aggregated statistic for models from the Llama family (Grattafiori et al., 2024), Mistral-0.3 7B (Jiang et al., 2023), and GPT-OSS 20B (OpenAI et al., 2025). For the mechanistic analyses in Sections 5.2 and 5.3, we use the instruction-tuned Gemma-3 12B model.

5.1 Robustness of Set Generation

We start with the question of how the choice of start element affects the reliability of set generation. This provides a first indication of how LLMs are influenced by orderings.

To do that, we compare four settings: **(1)** no start element, **(2)** a start element from the first position, **(3)** a single start element from any position, and **(4)** an average across 25 runs with randomly shuffled start elements. We embed these start elements by pre-filling them in the model's response so that the model is asked to generate all remaining entities while being forced to work with a prefix of 1 or k start elements. Contrasting settings **(3)** and **(4)** isolates the effect of the *amount* of deviation from the canonical order: a single misplaced element already perturbs generation, and the disruption grows as more shuffled elements are forced into the prefix.

Table 2 shows the influence of different types of start elements across model sizes in the Gemma-3 family and across set sizes categories. From the results, we observe that providing the first element according to a canonical ordering is beneficial compared to having no start element at all as it provides a strong signal about the expected ordering and composition of the multi-valued relation.[3] Furthermore, forcing the model to continue from randomly shuffled multiple start elements substantially decreases the robustness of set generation by a large margin, indicating that "disrupting" the canonical ordering leads to instabilities in the generation. Figure 1 displays the same tendencies across more model families aggregated over all list size categories.

	1B			4B			12B			27B		
Example(s)	S	R	L	S	R	L	S	R	L	S	R	L
0-shot	56.9	36.2	36.3	78.7	79.1	50.2	**92.5**	**89.7**	80.9	86.4	88.4	**85.1**
First	**61.3**	**51.8**	**36.4**	**84.7**	**85.5**	**66.2**	92.2	87.4	**82.6**	**89.7**	**89.9**	83.4
1 Random	53.0	46.8	24.6	77.5	74.7	49.6	90.1	83.3	72.8	88.6	87.7	81.6
k Random	42.2	34.7	13.0	69.1	58.3	34.6	84.7	73.9	48.2	83.9	77.9	57.5

Table 2: Average F1 (%) for models in the Gemma-3 family, split by set sizes and prefix strategy. S = small, R = regular, L = large.

We further observe that the negative effect of shuffled prefixes grows with the size of the target set: the gap between the canonical (first-element) and shuffled settings is largest for large sets across all model sizes. We hypothesize that this occurs because larger sets leave more room for the model to deviate from the canonical order, and the autoregressive pre-training can bias LLMs toward storing multi-valued relations as ordered sequences rather than independent triples.

[3] In the case of 0-prefixes, the LLMs often tend to be more verbose in their output, e.g., adding additional information in parentheses or writing full sentences after the bullet points.

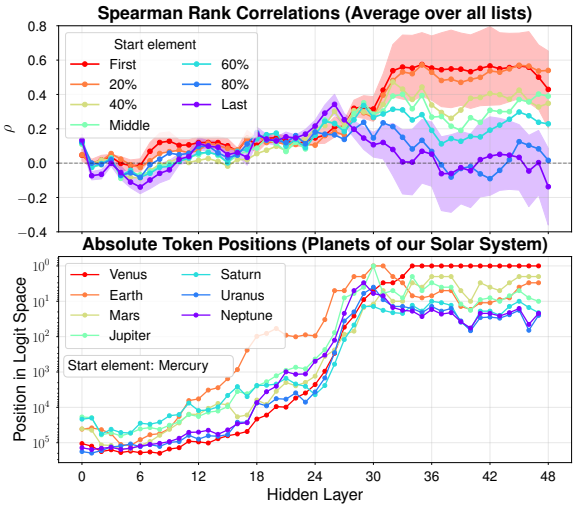


Figure 3: **(a)** Evolution of the Spearman correlation between expected and generated positions in each hidden layer of the first forward pass. We evaluate different choices for the 1-prefix position. Shaded areas mark $\pm$ 1 standard deviation. **(b)** Evolution of the absolute ranks in the sorted logit space for the 1-prefix *Mercury*. Both demonstrate that the canonical ordering emerges in the middle layers of the model.

In sum, providing a single start element following the canonical ordering can elicit multi-valued relations from LLMs more robustly, while deviating from this order hurts the robustness. In the following, we analyze mechanistically why sticking to the canonical order leads to better results.

5.2 Analysis of Prevalence of Canonical Orderings

In this section, we address the question of in which layers the canonical ordering of the target set is visible in the hidden states, and how this ordering behaves across consecutive forward passes.

Single Forward Pass We begin by investigating the initial forward pass. We instruct the model to generate the set of entities $\hat{\mathcal{S}}(t)$ for a given topic. Furthermore, we extend the prompt with a short introductory phrase followed by a bullet point, ensuring that the first generated token corresponds to an entity in the set.

The top plot of Figure 3 shows the evolution of the correlation coefficients ρ in all 48 hidden layers of Gemma-3-12B. As explained in Section 4, we compute the correlation between the expected canonical order of entities and their actual relative ranks. These values are averaged across the 12 topics in our dataset for which computation of all logit lens values is tractable.[4]

It becomes evident that up to approximately layer 25-30, there is almost the same correlation between the expected (inferred by the canonical order) and actual relative positions regardless of the input element. Beginning from layer 30, the model begins to be influenced by which entity is provided as the starting example. When using an element from the first position there is a medium-high correlation of around 0.5, consistent with the model internally retrieving an ordered representation of the target set. This correlation gradually drops the more the starting element is taken from the end of the set. Interestingly, for a start element from the last position (e.g., *Neptune*), there is no highly negative correlation. The visualized standard deviations of the red and purple lines further show that this finding holds across a variety of topics. We interpret this as evidence that the model internally stores a representation aligned with a canonical ordering, while forcing it to deviate from it shows signs of the model losing any structured order. We hypothesize that this behavior arises primarily from the pre-training data distribution, where entities belonging to such sets are most often encountered in their canonical ordering.

Exemplary Topic To further understand the inner workings of set generation during the first forward pass, we examine the multi-valued relation "planets of our solar system" as an example using *Mercury* as the starting element, following the canonical ordering. The bottom plot of Figure 3 depicts the absolute ranks of the logits corresponding to the remaining seven planets. Beginning around layer 20, the model starts to "pull" all necessary entities to top positions in the sorted logit space.

Referring to the bottom plot of Figure 3, it becomes clear that once entities have been pulled to the top of the logit space, the model begins to sort them according to its internal ordering. Summarizing the findings of both plots, we can define the following phases of multi-valued relation generation in LLMs: **(1) retrieval** of relevant entities, **(2) sorting** of the retrieved elements, and **(3) selection** of the next element to be generated.

Spearman Correlation Values for different Models Figure 4 shows the difference across hidden layers of the first forward pass between the correlation values ρ when using the first element and the last element as starting element of the set. We can see that our findings generalize to a variety of multi-valued relations and that approx. the last third of layers is where the internalized canonical order plays the most important role.

Across Forward Passes Figure 5 displays the average values of ρ for the different starting elements described above but this time across the last hidden states of all decoding steps, i.e., the hidden states that are used to compute the model's output predictions. We observe very similar tendencies, with later start elements leading to low correlation with the canonical ordering. Note that with an increasing number of generated set elements, the correlation with the canonical order for the remaining elements increases as there are fewer degrees of freedom for the model.

[4] ancient world wonders, books of the Bible, Nobel prize categories, countries in South America, distinct F1 world champions, MBTI personality types, official languages in India, official languages in the EU, phases of the moon, planets of our solar system, poker hands, states in Germany

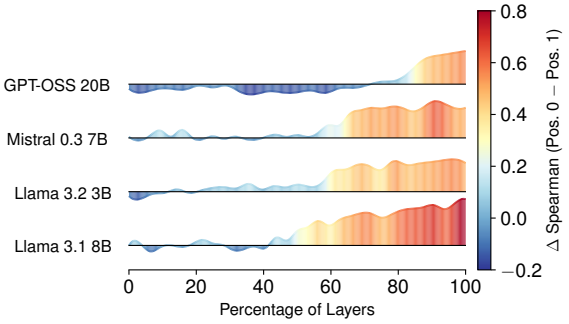


Figure 4: Difference of correlation values ρ when using the first and last element as start element across the hidden layers of the first forward pass.

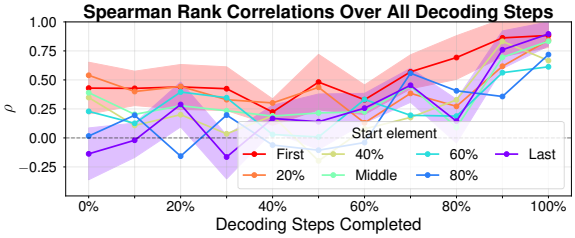


Figure 5: Evolution of the Spearman correlation values between expected positions and actual positions across the last hidden states of all decoding steps. Shaded areas indicate ± 1 standard deviation.

5.3 Evolution of Absolute Ranks and Missing Elements

In this section, we look at how entities that are ultimately missing from the generated output behave in the logit space during decoding. For this, we examine the much larger set of all "Formula 1 world champions" as an example and compare two runs: one with the first-ever world champion (*Giuseppe Farina*) as start element and one with *Max Verstappen* as start element, who is the most recent world champion w.r.t. the knowledge cutoff dates of most LLMs. For all champions, we track the absolute rank of their first subtoken in the sorted logit space across all forward passes in which the generation of a new entity begins.[5]

Figure 6 shows the set generation process of all F1 world champions, including the absolute ranks at each entity generation start. The rows are sorted in ascending order, starting with the earliest world champion (depending on the prefix used) in the first row. A cell marked with a star indicates that the driver of this row was chosen at the corresponding generation step. White cells indicate absolute positions larger than 50. We can see that drivers that have been generated only play a minor role in subsequent decoding steps, regardless of the canonical ordering. What we can see is that deviating from the canonical ordering by inputting *Max Verstappen* as first element has the result that the model does not stick to a generation order anymore. Furthermore, deviating from the canonical ordering leads to fewer drivers being generated by the model.

Rank Developments of Missing Elements In Figure 7, we show the relative rank development of three different drivers that are missing from the final output when inputting the first ever champion as start element. We can see that they ranked at high relative spots, especially in the early stages of the list generation process. However, they were never actually selected since they never ranked top-1 at all. This shows that LLMs can actually miss elements even though they played an actual role during decoding. As a result, future research should focus on detecting missing elements from the logit scores to improve recall.

We can conclude three major findings when comparing the two runs. **(1)** Using the first champion as the start element allows the LLM to generate subsequent drivers mostly in order, as indicated by the diagonal pattern. **(2)** In contrast, using the most recent champion leads to generation of drivers out of order rather than in reverse chronological order. **(3)** Sticking to the order leads to more drivers being successfully generated, while deviating from the canonical ordering, in this case chronological, results in more omitted elements. This aligns well with our previous findings that LLMs are more reliable when generating following the canonical ordering of the topic.

6 Conclusion & Outlook

We position our work as a starting point for further research on multi-valued relation generation and relational knowledge extraction from LLMs as KBs. Our mechanistic analysis suggests that multi-valued relation generation in LLMs follows three phases: **(1)** "retrieval" of relevant entities in the form of assigning higher probabilities to tokens that are part of entities, **(2)** internal sorting according to a relation-specific canonical ordering, and **(3)** selection of the next element during decoding. Ultimately, our findings and illustration highlight a general issue with the perception of LLMs: while they encode a vast amount of useful knowledge, they do not structure it in the way principled KBs are designed or in the way humans reason about this knowledge. Instead, our experiments clearly demonstrate that autoregressive transformer architectures learn probability distributions reflecting canonical orderings seen during pretraining rather than actually incorporating the relationships inherent to multi-valued relations. To mitigate this issue, we recommend incorporating shuffled and perturbed variants of multi-valued relations during pre-training. More broadly, our findings suggest that using LLMs as knowledge bases requires accounting for their internal ordering biases.

[5] Note that rank collisions can occur when driver names share the same first subtoken.

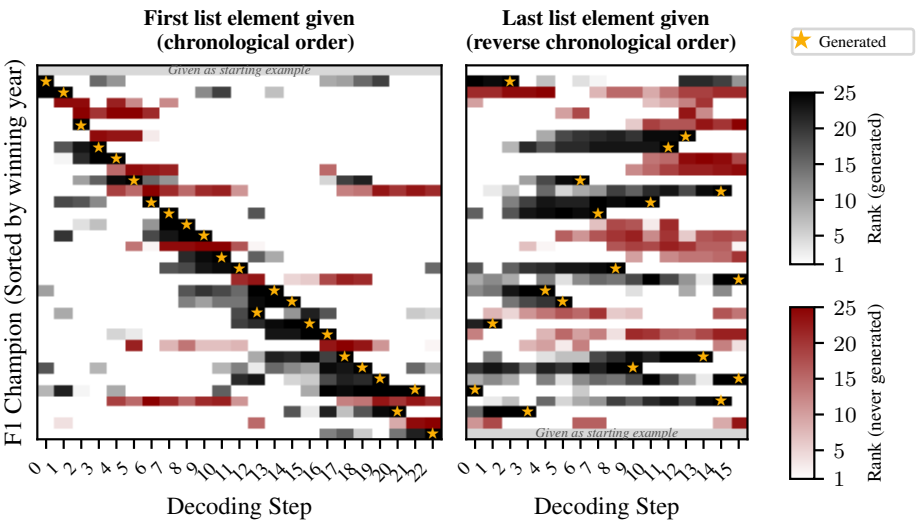


Figure 6: Evolution of absolute ranks when generating the set of all distinct Formula 1 world champions given the first (left, *Giuseppe Farina*) and most recent champion (right, *Max Verstappen*) as start element.

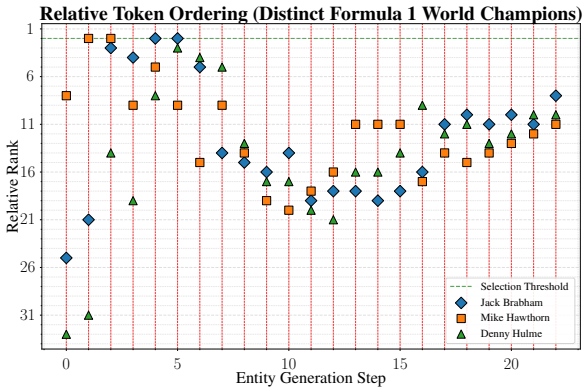


Figure 7: Relative rank development of three different drivers that were not generated into the final output.

Limitations

A limitation of our work arises from the fact that we rely on subtokens when identifying entities in the logit space. These subtokens cannot always be unambiguously mapped to their corresponding entities, especially when multiple entities share the same initial syllables or characters. To mitigate this issue, we primarily select entities that are distinguishable in the subtoken space for our mechanistic analyses. Nevertheless, some collisions may still occur, in which case we cannot reliably determine which entity the model would generate if it continued generation from that subtoken.

Furthermore, in this work, we only deal with sets that do not change too frequently. We leave the analysis of frequently changing sets and the LLM behavior on them for future work.

Ethical Considerations

In this work, we focus exclusively on publicly accessible and well-known topics. We do not incorporate any personally sensitive data that could be used to harm individuals. Furthermore, we do not aim to elicit secret or confidential knowledge that could be used in a harmful way. Instead, our goal is to analyze how publicly available knowledge can be elicited from LLMs.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Ankush Agarwal, Chaitanya Devaguptapu, and Ganesh S. 2025. Hybrid graphs for table-and-text based question answering using LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 858–875, Albuquerque, New Mexico. Association for Computational Linguistics.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *Preprint*, arXiv:2204.06031.

Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. 2024. A survey on rag with llms. *Procedia computer science*, 246:3781–3790.

Lukas Berglund, Meg Tong, Maximilian Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2024. The reversal curse: Llms trained on "a is b" fail to learn "b is a". In

The Twelfth International Conference on Learning Representations.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6596–6620. PMLR.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Max Dreger, Kourosh Malek, and Michael Eikerling. 2025. Large language models for knowledge graph extraction from tables in materials science. *Digital Discovery*, 4(5):1221–1231.

Gemma-Team et al. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Aaron Grattafiori et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yujia Hu, Shrestha Ghosh, Tuan-Phong Nguyen, and Simon Razniewski. 2024. Gptkb: Building very large knowledge bases from language models. *arXiv preprint arXiv:2411.04920*.

Jiahao Huo, Yibo Yan, Boren Hu, Yutao Yue, and Xuming Hu. 2024. MMNeuron: Discovering neuron-level domain-specific interpretation in multimodal large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6801–6816, Miami, Florida, USA. Association for Computational Linguistics.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

YUE JIANG, Yile Chen, Xiucheng Li, Qin Chao, SHUAI LIU, and Gao Cong. 2025. FSTLLM: Spatio-temporal LLM for few shot time series forecasting. In *Forty-second International Conference on Machine Learning*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Jinyang Li, Binyuan Hui, GE QU, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024b. Rethinking tabular data understanding with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 450–482, Mexico City, Mexico. Association for Computational Linguistics.

Zhiqiang Liu, Chengtao Gan, Junjie Wang, Yichi Zhang, Zhongpu Bo, Mengshu Sun, Huajun Chen, and Wen Zhang. 2025. Ontotune: Ontology-driven self-training for aligning large language models. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 119–133, New York, NY, USA. Association for Computing Machinery.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Huu Tan Mai, Cuong Xuan Chu, and Heiko Paulheim. 2024. Do llms really adapt to domains? an ontology learning perspective. In *International Semantic Web Conference*, pages 126–143. Springer.

Lucas Moeller, Pascal Tilli, Thang Vu, and Sebastian Padó. 2025. Explaining caption-image interactions in CLIP models with second-order attributions. *Transactions on Machine Learning Research*.

Haruki Nagasawa, Benjamin Heinzerling, Kazuma Kokuta, and Kentaro Inui. 2023. Can LMs store and retrieve 1-to-n relational knowledge? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 130–138, Toronto, Canada. Association for Computational Linguistics.

OpenAI et al. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Andrea Papaluca, Daniel Krefl, Sergio Rodríguez Méndez, Artem Lensky, and Hanna Suominen. 2024. Zero- and few-shots knowledge graph triplet extraction with large language models. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 12–23, Bangkok, Thailand. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Tomás Cerveira Da Cruz Pinto, Hugo Gonçalo Oliveira, and Chris-Bennet Fleger. 2025. Exploring medium-sized llms for knowledge base construction. In *Proceedings of the 5th Conference on Language, Data and Knowledge*, pages 221–232.

Nirmalendu Prakash and Roy Ka-Wei Lee. 2023. Layered bias: Interpreting bias in pretrained large language models. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 284–295, Singapore. Association for Computational Linguistics.

Zongyue Qin, Chen Luo, Zhengyang Wang, Haoming Jiang, and Yizhou Sun. 2024. Relational database augmented large language model. *Preprint*, arXiv:2407.15071.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

John W Ratcliff, David E Metzener, et al. 1988. Pattern matching: The gestalt approach. *Dr. Dobb's Journal*, 13(7):46.

André Gomes Regino and Julio Cesar dos Reis. 2025. Can LLMs be knowledge graph curators for validating triple insertions? In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, pages 87–99, Abu Dhabi, UAE. International Committee on Computational Linguistics.

Sneha Singhania, Simon Razniewski, and Gerhard Weikum. 2023. Extracting multi-valued relations from language models. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 139–154, Toronto, Canada. Association for Computational Linguistics.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 697–706, New York, NY, USA. Association for Computing Machinery.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pages 3319–3328. PMLR.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Rahman S. M. Wahidur, Sumin Kim, Haeung Choi, David S. Bhatti, and Heung-No Lee. 2025. Legal query rag. *IEEE Access*, 13:36978–36994.

Mingyang Wang, Heike Adel, Lukas Lange, Yihong Liu, Ercong Nie, Jannik Strötgen, and Hinrich Schuetze. 2025. Lost in multilinguality: Dissecting cross-lingual factual inconsistency in transformer language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5075–5094, Vienna, Austria. Association for Computational Linguistics.

Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. A novel cascade binary tagging framework for relational triple extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488, Online. Association for Computational Linguistics.

Bowen Zhang and Harold Soh. 2024. Extract, define, canonicalize: An llm-based framework for knowledge graph construction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9820–9836.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *Preprint*, arXiv:2102.09690.

Wei Zhou, Mohsen Mesgar, Annemarie Friedrich, and Heike Adel. 2025a. Efficient multi-agent collaboration with tool use for online planning in complex

table question answering. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 945–968, Albuquerque, New Mexico. Association for Computational Linguistics.

Yigeng Zhou, Wu Li, Yifan Lu, Jing Li, Fangming Liu, Meishan Zhang, Yequan Wang, Daojing He, Honghai Liu, and Min Zhang. 2025b. Reflection on knowledge graph for large language models reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23840–23857, Vienna, Austria. Association for Computational Linguistics.

Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. 2025. Knowledge graph-guided retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8912–8924, Albuquerque, New Mexico. Association for Computational Linguistics.

FALCON: A Model and Dataset Agnostic Framework for Synthetic Data Generation for NL2SQL Pairs

Darian Lee*, **Shannon Rumsey***, **Jack St. Clair***, **Xinyi Tang***
University of California, Santa Cruz
1156 High Street, Santa Cruz, CA 95064
{daeilee, sarumsey, jrstclai, xtang21}@ucsc.edu

Aditya Bansal, Yuanming Shi†
Adobe
{jeremyshi, adibansal}@adobe.com

Abstract

Relational databases are among the most widely deployed forms of structured knowledge, and natural language access to them requires grounding language onto schema entities and relations while handling the ambiguity inherent in how people phrase requests. Existing synthetic NL-to-SQL data generation methods largely ignore this ambiguity and produce oversimplified queries that fail to prepare models for the complexity of real-world structured knowledge access. We present FALCON, a framework that generates realistic, ambiguity-aware NL-to-SQL data matching the complexity of challenging real-world benchmarks, at low cost using compact open models. Our approach combines reserved-word SQL seeding and persona-based prompting to generate structurally complex queries, while alignment-based filtering preserves difficulty by distinguishing genuinely incorrect examples from complex but valid queries. Human evaluation confirms consistent high quality across model sizes, and our generated data exceeds existing benchmarks in both SQL complexity and natural language richness. Difficulty-stratified analysis shows models trained on FALCON data increasingly outperform baseline-trained models as query complexity increases, validating our pipeline's success in generating challenging training data. When combined with a small proportion of existing benchmark data, mixed training recovers performance on simpler queries while preserving these advantages on complex ones. The model- and database-agnostic design enables organizations to generate high-complexity NL-to-SQL training data locally without external APIs.

1 Introduction

As large language model (LLM) capabilities advance, natural language interfaces for databases are becoming increasingly accessible to non-SQL experts, making NL-to-SQL parsing an increasingly consequential knowledge-grounding task (Yang et al., 2024; Dai et al., 2025; Yu et al., 2018; Li et al., 2023; Gao et al., 2024; Li and Jagadish, 2014). This requires resolving natural language references onto specific schema entities and relations, much as entity linking and relation extraction do for other forms of knowledge base construction. In practice, this grounding is complicated by real-world database users, who employ diverse linguistic styles, pose ambiguous requests, and work with complex SQL operations that existing benchmarks underrepresent (Li et al., 2025; Saparina and Lapata, 2024; Li et al., 2023; Floratou et al., 2024).

While challenging datasets such as BIRD (Li et al., 2023) have driven progress toward more realistic benchmarks, they remain human-authored, limiting scalability and adaptability to new domains. Existing synthetic data generation approaches face a critical trade-off: they either rely on expensive frontier model APIs, making them cost-prohibitive for many organizations, or produce oversimplified queries that fail to capture production-level complexity (Lupidi et al., 2025; Guo et al., 2025).

We introduce **FALCON**, a **F**ramework for synthetic **A**mbiguity-aware **L**inguistically-diverse **C**ost-efficient **O**ptimized **N**L-to-SQL data generation. FALCON addresses three core challenges: (1) generating BIRD benchmark-level query complexity, (2) capturing realistic linguistic diversity including ambiguity, and (3) achieving low cost using only compact open models (e.g. Qwen-2.5-3B). The framework is designed to complement existing datasets by targeting the complex end of the query distribution where existing resources fall short, and is model- and database-agnostic, allowing organizations to generate domain-specific training data for their internal schemas without sending sensitive information to external APIs.

*Equal contribution.
†Work performed while at Adobe.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 56–87
October 28, 2026 ©2026 Association for Computational Linguistics

2 Related Work

2.1 The Academic-Industry Gap in NL-to-SQL

Early NL-to-SQL datasets established foundational benchmarks but with significant limitations for real-world deployment. WikiSQL (Zhong et al., 2017; Yavuz et al., 2018) focused on single-relation queries, while Spider (Yu et al., 2018) and BIRD (Li et al., 2023) introduced cross-domain complexity and professional-domain examples. However, recent work has identified critical gaps between benchmark performance and production viability, including ambiguous queries from informal language use and underspecification, and database complexity at enterprise scale (Floratou et al., 2024).

2.2 The Cost Barrier to Industry Deployment

Beyond benchmark-to-production gaps, cost efficiency and privacy present fundamental barriers to enterprise adoption. While frontier LLMs demonstrate strong performance, their API costs make continuous deployment prohibitive (Hoang et al., 2025). Fine-tuning smaller models offers significant cost and privacy advantages (Luo et al., 2025; Li et al., 2025) and can outperform prompting (Li et al., 2024a), but requires substantial high-quality training data. Organizations need production-ready systems handling complex queries and diverse language (Li et al., 2024a), yet generating training data via expensive frontier models or human annotators undermines local deployment benefits.

2.3 Synthetic Data Generation for NL-to-SQL

Researchers have explored automated synthesis to overcome scalability limitations. Source2Synth (Lupidi et al., 2025) combines WikiSQL tables with LLM-generated facts using execution-based verification, but requires fine-tuning multi-billion parameter models for filtering and focuses only on single-table queries. OmniSQL (Li et al., 2025) expands to more difficult queries and diverse linguistic styles, yet lacks ambiguity handling and requires larger model ensembles. SQLForge (Guo et al., 2025) introduces SQL template parsing and iterative domain exploration but relies on expensive closed-source models (GPT-4) with declining execution rates on complex queries.

2.4 Handling Query Ambiguity

Addressing ambiguous queries is critical for production reliability (Floratou et al., 2024), yet most synthetic datasets generate only unambiguous pairs. Know What I Don't Know (Wang et al., 2023a) categorizes unanswerable questions but focuses strictly on schema-level ambiguity, excluding broader linguistic ambiguities. Ambrosia (Saparina and Lapata, 2024) identifies database-independent ambiguities (scope, attachment, vagueness) that reflect real user behavior but remain underrepresented in training data (Floratou et al., 2024).

Implications While these existing methods address specific facets of NL-to-SQL generation, there remains a need for a unified, cost-efficient framework that handles complexity diversity and ambiguity; we address these gaps with the FALCON pipeline described in the following section.

3 Pipeline

Our pipeline follows a multi-stage generation and filtering process: (1) SQL generation with reserved word and persona seeding, followed by executability filtering; (2) NL generation with LLM-as-a-judge candidate selection; (3) ambiguity detection and multiple query generation for questions that are ambigious; (4) alignment-based filtering to remove misaligned NL-SQL pairs; and (5) final prompt-based filtering to remove any remaining low-quality instances (Figure 1). We evaluate generation components by their contribution to SQL complexity and NL richness, and filtering components by their ability to retain high-quality examples while removing low-quality ones (validated against human annotations).

3.1 SQL Query and NL Question Generation

We adopt SQL-then-NL generation (Hu et al., 2023; Wu et al., 2021; Lupidi et al., 2025), which empirically increases both SQL and NL complexity (Table 1). The LLM is prompted with a table schema to generate a SQL query, including foreign key relationships when present. Each query is executed against the database with up to three attempts to ensure validity before discarding.

The LLM is then prompted with the validated query and schema to generate four candidate NL questions via chain-of-thought prompting (Prompt 19). An LLM-as-a-judge selects the optimal candidate based on our rubric (Bai et al.,

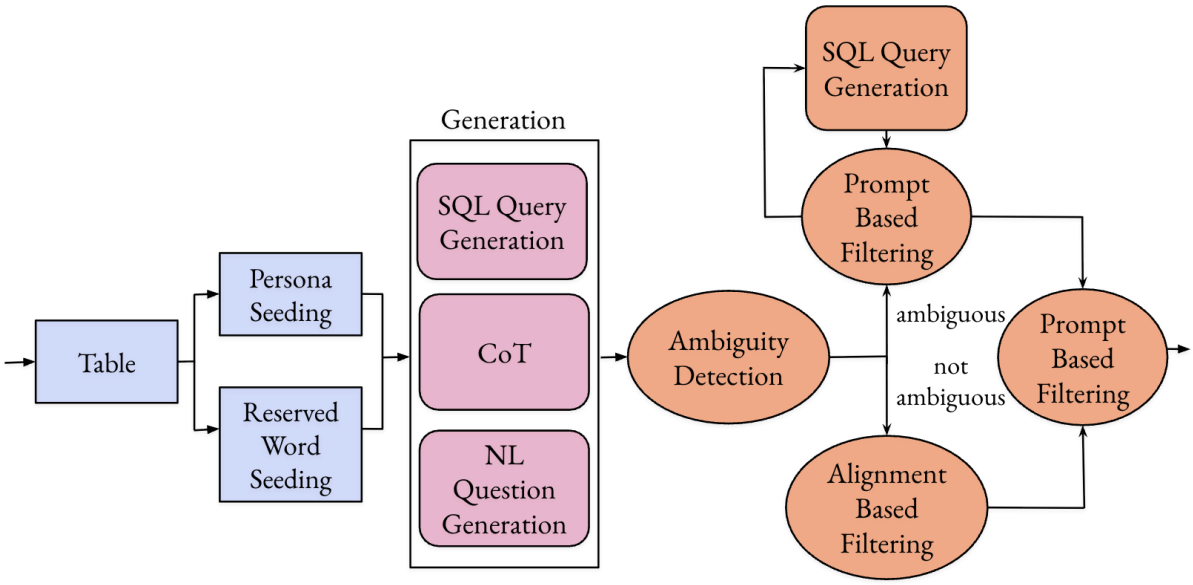


Figure 1: Simplified architecture of the FALCON pipeline. It integrates systematic seeding, SQL and NL generation with CoT, and a multistage filtering process (Ambiguity, Reward, and Prompt-based), to produce high-quality NL-to-SQL pairs. To see a more detailed diagram of our pipeline, see Figure 6

2022). Requiring higher variation among candidates forces the model to produce more diverse linguistic options, improving selection quality (Section 5.1).

3.1.1 Seeding Techniques

To mitigate the inherent repetitiveness of synthetic data (Wang et al., 2023b), we interleave two seeding techniques to encourage the LLM to generate unique results (Table 1).

Persona Seeding. Inspired by Jandaghi et al. (2024); Ge et al. (2025), for each table, we prompt the LLM to generate a realistic user persona with specific goals and example questions. We sample between two prompts to ensure a mix of common personas (e.g., "a software engineer") and creative ones (e.g., "an indie tabletop racing game designer"). This improves the complexity and diversity of generated questions while ensuring varied use cases (Table 1, Prompt 16, Prompt 17, Prompt 18, Prompt 20).

Persona: TV producer revitalizing a reality show using 'The Amazing Race' data.
Goals: Identify optimal timeslots and premiere dates.
Example: How do fall vs. spring premiere ratings compare?

Figure 2: Persona seeding example included in SQL generation prompts to introduce linguistic variety.

Reserved Word Seeding. To align our synthetic SQL with human-authored distributions, we specify target reserved word counts sampled from BIRD's distribution (Li et al., 2023), a widely-accepted challenging benchmark (Liu et al., 2025; Wretblad et al., 2024; Gao et al., 2024; Pourreza et al., 2025). The model achieves reasonable adherence (Pearson corelation $r = 0.6$, median deviation of 3 words) and ablations confirm reserved word seeding improves SQL complexity despite small deviations (Table 1, Prompt 15).

3.1.2 Ambiguity Detection

After generating candidate NL-SQL pairs, we assign each example a continuous ambiguity score using a DistilBERT model fine-tuned on Ambrosia data (Saparina and Lapata, 2024). We define ambiguity as occurring whenever an NL question can be correctly interpreted as several distinct SQL queries. Predicted scores $\hat{y}$ trigger two pipeline stages: (1) filtering removes extremely ambiguous questions, and (2) multi-interpretation generation (when $\hat{y} > 7$) produces alternative SQL queries for genuinely ambiguous examples using the prompt-based filters shown in Appendix Figures 23 and 24 to confirm genuine ambiguity and validate each alternative interpretation. Since this step may generate more than one SQL query for a single NL question, it also slightly increases the diversity of FALCON (see Appendix A.8 for training de-

Persona (shortened): Amanda Thompson, a travel blogger who uses storm record data to create safety guides for her readers. *NL:* "What are the 5 regions that have experienced the most storms in the past 10 years, and what are their region names and codes, so I can warn my readers about these storm-prone areas?"

```
SELECT `region`.`region name`, `region`.`region code`,
       COUNT(`storm`.`storm id`) AS `storm frequency`
FROM `region`
JOIN `affected region`
     ON `region`.`region id` = `affected region`.`region id`
JOIN `storm`
     ON `affected region`.`storm id` = `storm`.`storm id`
WHERE `storm`.`storm date` > DATE('now', '-10 years')
GROUP BY `region`.`region name`, `region`.`region code`
ORDER BY `storm frequency` DESC LIMIT 5;
```

The question is correctly grounded in the schema, joining `storm` and `region` through `affected_region` to compute per-region storm counts, while the trailing context ("so I can warn my readers") reflects how real users frame requests around goals rather than abstract retrieval.

Figure 3: Example NL-SQL pair generated via persona seeding.

tails). Appendix Figures 7 and 8 provide a full pipeline walkthrough for an ambiguous and a non-ambiguous example respectively, and Appendix Figure 13 gives a further illustration.

3.1.3 Alignment-Based Filtering

Generative models frequently produce semantically plausible questions misaligned with SQL logic. We train an alignment prediction model on 9,000+ BIRD-derived examples (50% systematically corrupted) using a novel hybrid metric combining semantic similarity (40%) and schema entity matching (60%) with corruption penalties. This scoring metric achieves Pearson correlation $r = 0.70$ to human judgments (Appendix A.6). ModernBERT trained on this metric achieves MSE: 0.0203 and MAE: 0.1129, outperforming other models (Appendix Table 11) and filtering methods (Table 2). This filtering step verifies that the generated query is faithfully grounded in the schema entities and relations the question implies, rejecting semantically plausible but ungrounded pairs.

3.2 Multi-Interpretation Generation and Filtering

Because ambiguity resolution is central to reliably constructing structured knowledge from natural language, we explicitly model and generate alternative interpretations rather than filtering ambiguous examples away. For queries flagged as ambiguous, we prompt the model to identify the ambiguity type (Saparina and Lapata, 2024) and generate an alternative SQL interpretation (Prompt 25). The alternative is validated with an ambiguity-specific prompt filter (Prompt 23, 24); the model has three attempts to produce an executable alternative that passes this filter, otherwise the example proceeds as unambiguous.

All examples then undergo a final prompt-based filter verifying that: (1) the SQL validly interprets the NL, (2) the SQL is sensible and non-redundant, and (3) the NL resembles something a human would actually ask (Prompt 26). While the reward model achieves high precision (Table 2), it only reaches 71% specificity and was trained solely on alignment, missing aligned examples with poor NL or SQL quality. Prompt-based filtering addresses these limitations and adapts to ambiguity-specific criteria beyond the reward model's scope. Examples passing all checks are added to the final dataset.

4 Evaluation

4.1 Difficulty Metrics

We measure NL Richness as a weighted combination of linguistic complexity features (Balyan et al., 2020; Byrd and Srivastava, 2022): mean tokens before main verb (15%), constituents per word (30%), subordinate clauses (30%), and coordinate clauses (20%). SQL complexity is quantified using SQLGlot to parse query structures, with weighted features: words and columns (10% each), and joins, sub-queries, operators, and splits (20% each) (Yu et al., 2018; Li et al., 2023; Vashistha and Jain, 2014).

4.2 Human Evaluation

We conduct human evaluation with 4 annotators using the rubric in Appendix A.9, which separately grades NL quality, SQL quality, and alignment to distinguish failure modes (e.g., inefficient queries vs. misaligned NL-SQL pairs). Each annotator independently scored 100 disjoint examples, with disagreements resolved through structured group discussion following standard adjudication protocol.

4.3 Downstream Task Evaluation

We finetune Qwen 2.5 3B and 1.5B (Appendix A.11, A.12) on three training sets: (1) 5,500 synthetic examples, (2) 5,500 Spider examples, and (3) mixed data (95% synthetic, 5% Spider). Models are evaluated on our synthetic test set ($n = 2{,}000$) and Spider test set ($n = 2{,}147$) with consistent hyperparameters.

We employ three automatic metrics: **Exact Match (EM)**, direct string equivalence (Yu et al., 2018); **Fuzzy Score (FS)**, average string similarity; and **Soft Execution Accuracy (Soft-EX)**, F1 score of result set overlap (Li et al., 2024b). Fuzzy metrics capture semantically equivalent queries with minor syntactic variations. We conduct difficulty-stratified analysis across query complexity levels to characterize model performance on queries of varying difficulty.

5 Ablations

5.1 Generation Order and Component Ablations

To maximize SQL complexity and NL richness, we ablated generation order (SQL-then-NL vs. NL-then-SQL), seeding methods (persona and reserved words), and LLM-as-a-judge selection strategies for NL: a Compare Judge (selecting the best candidate NL, Prompt 21) and a Score Judge (iterative regeneration of NL until meeting a quality threshold, Prompt 22). This NL candidate selection occurs before the final alignment-based filtering (Section 3.1.3).

Table 1 shows that SQL-then-NL outperforms the reverse order. Reserved word seeding improves SQL complexity while persona seeding improves NL richness. The Compare Judge yields the highest NL richness and unexpectedly improves SQL complexity by pruning simpler queries associated with low-quality questions. We therefore adopt SQL-then-NL generation, interleave both seeding methods, and use the Compare Judge in our final pipeline.

Configuration	SQL Comp.	NL Rich.
Generation Order		
SQL→NL	**0.42**	**0.31**
NL→SQL	0.40	0.21
Seeding & Filtering (SQL→NL)		
Baseline	0.42	0.31
+ Reserved Words	**0.48**	0.31
+ Personas	0.42	**0.34**
+ Compare Judge	0.46	**0.36**
+ Score Judge	0.46	0.28

Table 1: Ablation results for generation order, seeding methods, and NL filtering strategies on SQL complexity and NL richness (n=1000) using Qwen-2.5-3B.

5.2 Filtering Ablation

Filtering Method Comparison We evaluate our reward model against three filtering methods on human-annotated data (n=181, 84% aligned): **(1) Teacher-based**, where a larger model (Llama-70B) regenerates SQL (k=3 tries), discarding non-reproducible examples; **(2) Source2Synth** Lupidi et al. (2025), which fine-tunes the generator (Qwen-2.5-3B) on unfiltered data and rejects examples it cannot reproduce in k=3 attempts. This performed poorly, suggesting limited generalization to smaller models; and **(3) Prompt-based LLM-as-judge**, which uses Llama-3-70B to score alignment via chain-of-thought reasoning.

Our reward model substantially outperforms baselines (Table 2), achieving 94% F1 and 71% specificity at threshold 0.6. It reduces misaligned examples from 16% to 4% while retaining 92.81% of correct examples (8-9× higher recall than competing methods). Filtered examples' SQL and NL complexity metrics match BIRD benchmarks, confirming our filtering preserves difficulty without distributional bias (Appendix 9).

5.3 Ambiguity Ablation

We evaluate the ambiguity pipeline on 100 human-annotated examples from our final dataset. For each example, human annotators judged whether the query was genuinely ambiguous and whether the generated alternative SQL queries were valid. An example was counted as a true positive only when both conditions held. Those judgments were aggregated into a confusion matrix from which we

Method	F1	Specificity	Recall
Reward Model	**93.73**	71.43	92.81
Prompt Filter	91.54	28.57	**95.42**
LLaMA Teacher	17.65	92.86	9.80
Source2Synth	7.55	**100.00**	3.92

Table 2: Filtering results across different methods. F1 measures overall filtering performance. Specificity shows the percentage of low-quality examples correctly filtered out. Recall shows the percentage of high-quality examples retained. Results are shown at the best threshold for each model.

compute accuracy, precision, recall, F1, and specificity. Table 3 shows 94% accuracy with 98.9% specificity with 28.6% recall. This conservative behavior is intentional: false positives generate noisy multi-interpretation pairs, while false negatives simply omit optional enhancements. High specificity ensures triggered multi-interpretation generation produces genuine alternatives.

Accuracy	Precision	Recall	F1	Specificity
94.0%	66.7%	28.6%	40.0%	98.9%

Table 3: End-to-end ambiguity pipeline performance on human-annotated samples from final synthetic dataset (n=100), evaluating both detection accuracy and validity of generated alternative queries.

6 Results

Human evaluation with four annotators (100 examples each using the rubric in Appendix A.9) shows consistently high quality across all models (Table 4), with SQL, NL, and alignment scores all exceeding 0.92. Quality holds across seed base databases (Appendix 12), and complexity remains consistent across models without declining after filtering—in fact, it improves (Appendix 9).

Table 5 shows that our FALCON-generated datasets outperform the original datasets in terms of SQL complexity and NL richness, despite using the same table information. With the exception of the WikiSQL-seeded pipeline's SQL complexity, the FALCON-generated data is comparable, if not exceeding the complexity of human generated data in BIRD.

Our downstream results validate FALCON's core objective: generating challenging training data. Difficulty-stratified analysis demonstrates that FALCON-trained models significantly outperform Spider-trained models as query complexity increases, with the performance advantage growing in harder quartiles (Figure 4). Spider-trained models achieve 36% EM on Spider but only 0.05% on FALCON, while FALCON-trained models achieve 13.8% on FALCON but 0.8% on Spider (Table 7), reflecting Spider's over-representation of simpler queries (Appendix A.5). The low raw EM on Spider is consistent with known string match underestimation when training and test distributions diverge significantly. Our error analysis finds approximately a third of incorrect predictions are acceptable variations undetected by string match, likely amplified here relative to Spider-trained baselines due to the larger distribution gap (Appendix A.2). When supplemented with 5% Spider data, mixed training achieves 27% EM on Spider and 13.6% on FALCON (raw string match). Figure 5 shows that real data seeding primarily benefits easier queries (first octile) while FALCON-only training maintains advantages on harder queries (last octile). The most common genuine failure modes are spurious predicate hallucination and result boundary errors, attributable to the overrepresentation of complex queries in our training data; mixing with 5% real data is expected to reduce these artifacts (Section 6.1).

Model	SQL	NL	Alignment
Qwen-3B	0.96	0.95	0.94
Llama-8B	0.92	0.94	0.95
Qwen-32B	0.98	0.97	0.93
Llama-70B	0.93	0.97	0.94

Table 4: Human evaluation scores across model sizes for the FALCON-generated data that was seeded with WikiSQL and filtered. All models were evaluated according to our rubric. We used the following models: Qwen-2.5-3B, Llama-3.1-8B, Qwen-3-32B, and Llama-3.3-70B.

Dataset	SQL Comp.	NL Rich.
Spider Seeded FALCON	**0.43**	**0.39**
WikiSQL Seeded FALCON	0.28	**0.39**
BIRD	0.42	0.31
Spider	0.38	0.29
WikiSQL	0.20	0.28

Table 5: Comparison of the SQL complexity and NL richness of multiple datasets with the most complex scores in bold. Spider seeded is FALCON seeded on spider data. WikiSQL seeded is FALCON seeded on WikiSQL.

6.1 Error Analysis

We conducted a failure mode analysis on 100 randomly sampled incorrect predictions from the Qwen 2.5-3B model finetuned on FALCON's generations, evaluated on the Spider test set, identifying nine error categories ranging from acceptable variations to genuine failures such as spurious predicate hallucination (adding a WHERE cause where none is needed) and result boundary errors (returning too much or too little) (full taxonomy and examples in A.2). Table 6 shows the distribution of error types in the annotated sample.

Error Type	Count	%
Acceptable variation	33	33.0
Spurious predicate hallucination	20	20.0
Result boundary error	18	18.0
Schema entity confusion	10	10.0
Logical negation failure	4	4.0
Degenerate join structure	4	4.0
Cross-row constraint flattening	4	4.0
Hardcoded quantifier substitution	4	4.0
NL misinterpretation	3	3.0
Total	100	100.0

Table 6: Distribution of error types across 100 sampled incorrect predictions from the model finetuned on FALCON-generated data, evaluated on the Spider test set.

Analysis. From this analysis we draw three conclusions. First, approximately one third of flagged errors are acceptable variations that would not mislead a real user, meaning our reported Spider accuracy is meaningfully underestimated. This is a known limitation of both string match and execution match metrics. This underestimation likely affects our model more than Spider-trained baselines, since style-based acceptable variations are more probable when training and test distributions diverge substantially. Second, spurious predicate hallucination and result boundary errors are the most common genuine failure modes, which we attribute to the overrepresentation of complex SQL queries in our dataset. We anticipate that mixing the training distribution with simpler data has the most impact here by reducing these artifacts, which is why we recommend seeding with 5% existing data (ex. from Spider or WikiSQL) to ensure simpler queries are represented. Third, schema entity confusion is more likely a consequence of how the model processes condensed schema information under context pressure than of our dataset composition: when training the model, we compressed schema information for all tables into the prompt, requiring the model to resolve column and table references under significant context pressure, independently of the underlying NL-to-SQL pairs.

Train → Test	EM (%)	FS	Soft-EX
Spider on Spider	**35.95**	**0.89**	**0.42**
Spider on Pipe	0.05	0.69	0.18
FALCON on Spider	0.80†	0.68	0.32
FALCON on Pipe	**13.80**	**0.83**	**0.48**
Mixed on Spider	27.10	0.84	–
Mixed on Pipe	13.60	**0.83**	–

Table 7: Cross-dataset evaluation on Qwen 2.5-3B. In-distribution rows serve as baselines contextualizing the cross-dataset results. Mixed is the intended deployment configuration. EM = Exact Match, FS = Fuzzy Score, Soft-EX = Soft Execution Accuracy. †Raw EM underestimates performance due to acceptable variations undetected by string match (Appendix A.2).

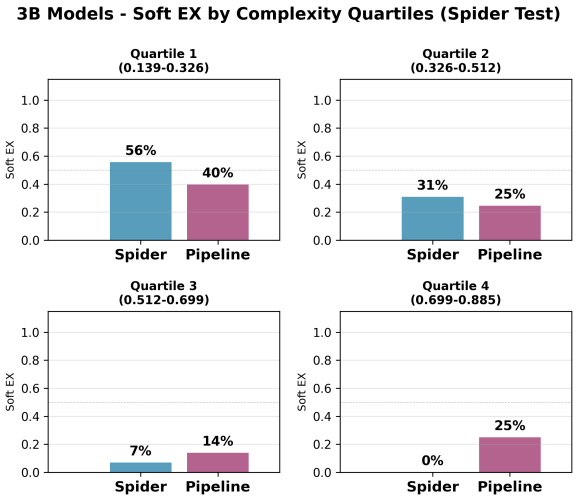


Figure 4: A comparison of soft execution accuracy for a model finetuned on Spider and the FALCON-generated data for different complexities.

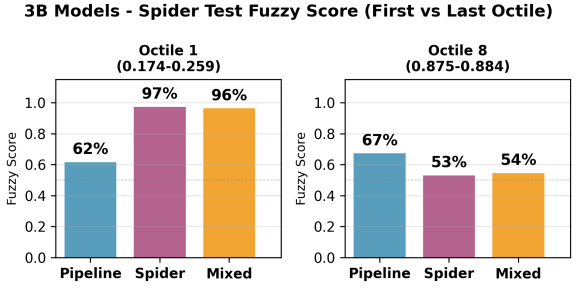


Figure 5: The fuzzy scores for a model finetuned on FALCON, Spider, and a mix of both.

6.2 Cost

FALCON requires no external API access, enabling organizations to generate training data locally without per-query costs or privacy exposure. Using on average 5 LLM calls per example across 5 pipeline stages, generating 5,500 examples requires approximately 34.47 million input tokens and 5.13 million output tokens. At GPT-5.5 pricing, an equivalent run would cost approximately $326, versus $20 for an open model on a commodity GPU (Vast.ai, $0.15/hr), an over 16× reduction, before accounting for the privacy implications of transmitting proprietary schemas to third-party APIs (Appendix A.4).

7 Conclusion

We present FALCON, a model- and database-agnostic framework for generating high-complexity NL-to-SQL data at low cost using compact open models. Through reserved-word seeding, persona-based prompting, and alignment-based filtering, FALCON generates data exceeding existing benchmarks in SQL complexity and natural language richness while maintaining high quality (>0.92 human evaluation across all models). Our ambiguity handling addresses a critical gap in synthetic generation, producing diverse data reflecting real-world linguistic challenges.

Models trained on FALCON-generated data outperform Spider-trained models as query complexity increases. FALCON's ambiguity-aware, schema-grounded approach to NL-to-SQL generation offers a template for producing training data for natural language interfaces to structured knowledge more broadly, and its local, model-agnostic design keeps this accessible to organizations without reliance on frontier APIs.

8 Limitations

FALCON is intentionally designed to target the complex end of the query distribution, where existing datasets and template-based methods already provide adequate coverage of simpler queries. This focus leads to an underrepresentation of simpler queries, as shown in Figure 9, and models trained solely on FALCON data achieve only 0.8% EM on Spider, requiring 5–10% real data supplementation for broad applicability. Ambiguity detection achieves high specificity (98.9%) but low recall (28.6%), missing many genuinely ambiguous queries.

FALCON currently supports SQLite only; extension to other dialects requires dialect-specific reserved word distributions and syntax constraints. Evaluation is limited to public benchmarks (Spider, WikiSQL, BIRD), and production databases may exhibit complexity dimensions not captured academically. Resource constraints limited downstream evaluation to compact models, though these suffice for high-quality generation (>0.92 human evaluation scores). Generated data inherits domain coverage from seed databases.

Ethical Considerations

Domain coverage reflects source databases (Spider and WikiSQL), potentially inheriting domain-specific biases. The database-agnostic design enables organizations to generate data for their specific use cases. AI tools were used to aid in non-essential writing and code generation.

Acknowledgments

The authors would like to thank the anonymous reviewers for their feedback. We would also like to thank Shriya Sravani Yellapragada for her early work on this research. Additional thanks to Beth Ann Hockey and Jeffrey Flanigan for their advice and support.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, John Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Chris Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, E Perez, and 32 others. 2022. Constitutional ai: Harmlessness from ai feedback. *ArXiv*, abs/2212.08073.

Renu Balyan, Kathryn S McCarthy, and Danielle S McNamara. 2020. Applying natural language processing and hierarchical machine learning approaches to text difficulty classification. *International Journal of Artificial Intelligence in Education*, 30(3):337–370.

Matthew Byrd and Shashank Srivastava. 2022. Predicting difficulty and discrimination of natural language questions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 119–130.

Yaxun Dai, Haiqin Yang, Mou Hao, and Pingfu Chao. 2025. PARSQL: Enhancing text-to-SQL through SQL parsing and reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 661–681.

Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, Alex Van Grootel, Brandon Chow, Kai Deng, Katherine Lin, Marcos Campos, K. Venkatesh Emani, Vivek Pandit, Victor Shnayder, Wenjing Wang, and Carlo Curino. 2024. Nl2sql is a solved problem... not! In *Conference on Innovative Data Systems Research*.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2024. Text-to-sql empowered by large language models: A benchmark evaluation. *Proc. VLDB Endow.*, 17(5):1132–1145.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2025. Scaling synthetic data creation with 1,000,000,000 personas. *Preprint*, arXiv:2406.20094.

Yu Guo, Dong Jin, Shenghao Ye, Shuangwu Chen, Jian Yang, and Xiaobin Tan. 2025. SQLForge: Synthesizing reliable and diverse data to enhance text-to-SQL reasoning in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8441–8452, Vienna, Austria. Association for Computational Linguistics.

Cong Duy Vu Hoang, Gioacchino Tangari, Clemence Lanfranchi, Dalu Guo, Paul Cayet, Steve Siu, Don Dharmasiri, Yuan-Fang Li, Long Duong, Damien Hilloulin, Rhicheek Patra, Sungpack Hong, and Hassan Chafi. 2025. Distill-C: Enhanced NL2SQL via distilled customization with LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pages 833–848, Albuquerque, New Mexico. Association for Computational Linguistics.

Yiqun Hu, Yiyun Zhao, Jiarong Jiang, Wuwei Lan, Henghui Zhu, Anuj Chauhan, Alexander Hanbo Li, Lin Pan, Jun Wang, Chung-Wei Hang, Sheng Zhang, Jiang Guo, Mingwen Dong, Joseph Lilien, Patrick Ng, Zhiguo Wang, Vittorio Castelli, and Bing Xiang. 2023. Importance of synthesizing high-quality data for text-to-SQL parsing. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1327–1343, Toronto, Canada. Association for Computational Linguistics.

Pegah Jandaghi, Xianghai Sheng, Xinyi Bai, Jay Pujara, and Hakim Sidahmed. 2024. Faithful persona-based conversational dataset generation with large language models. In *Proceedings of the 6th Workshop on NLP for Conversational AI (NLP4ConvAI 2024)*, pages 114–139, Bangkok, Thailand. Association for Computational Linguistics.

Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. 2024a. The dawn of natural language to sql: Are we fully ready? *Proc. VLDB Endow.*, 17(11):3318–3331.

Fei Li and H. V. Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.*, 8(1):73–84.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025. Omnisql: Synthesizing high-quality text-to-sql data at scale. *Proc. VLDB Endow.*, 18(11):4695–4709.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM already serve as a database interface? A BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, pages 35618–35667.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2024b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Xinyu Liu, Shuyu Shen, Boyan Li, Nan Tang, and Yuyu Luo. 2025. NL2SQL-BUGs: A benchmark for detecting semantic errors in NL2SQL translation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Yuyu Luo, Guoliang Li, Ju Fan, Chengliang Chai, and Nan Tang. 2025. Natural language to sql: State of the art and open problems. *Proc. VLDB Endow.*, 18(12):5466–5471.

Alisia Maria Lupidi, Carlos Gemmell, Nicola Cancedda, Jane Dwivedi-Yu, Jason E. Weston, Jakob Nicolaus Foerster, Roberta Raileanu, and Maria Lomeli. 2025. Source2synth: Synthetic data generation and curation grounded in real data sources. In *Workshop on Synthetic Data × Data Access Problem at the International Conference on Learning Representations (ICLR)*. Best Paper Award.

Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Hailong Li, Yeounoh Chung, Lesly Miculicich, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. 2025. CHASE-SQL: Multi-path reasoning and preference optimized candidate selection in text-to-SQL. In *Proceedings of the International Conference on Learning Representations*.

Irina Saparina and Mirella Lapata. 2024. Ambrosia: a benchmark for parsing ambiguous questions into database queries. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Aditya Vashistha and Shrainik Jain. 2014. Measuring query complexity in SQLShare workload. Technical report, University of Washington.

Bing Wang, Yan Gao, Zhoujun Li, and Jian-Guang Lou. 2023a. Know what I don't know: Handling ambiguous and unknown questions for text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5701–5714, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Niklas Wretblad, Fredrik Riseby, Rahul Biswas, Amin Ahmadi, and Oskar Holmström. 2024. Understanding the effects of noise in text-to-SQL: An examination of the BIRD-Bench benchmark. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 356–369, Bangkok, Thailand. Association for Computational Linguistics.

Kun Wu, Lijie Wang, Zhenghua Li, Ao Zhang, Xinyan Xiao, Hua Wu, Min Zhang, and Haifeng Wang. 2021. Data augmentation with hierarchical SQL-to-question generation for cross-domain text-to-SQL parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8974–8983, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. Synthesizing text-to-SQL data from weak and strong LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7864–7875, Bangkok, Thailand. Association for Computational Linguistics.

Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2018. What it takes to achieve 100% condition accuracy on WikiSQL. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1702–1711, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *Preprint*, arXiv:1709.00103.

A Appendix

A.1 Detailed Pipeline Overview

We demonstrate the pipeline step by step using a running example. Note that all examples are simplified and made up for illustration purposes only; they are not taken directly from the pipeline output. A detailed diagram of the full pipeline is available in Figure 6, and step-by-step walkthroughs showing intermediate outputs for both an ambiguous and a non-ambiguous example are shown in Figures 7 and 8 respectively.

Table: The pipeline begins with an existing dataset of table schemas, such as WikiSQL, and iterates through them sequentially. For this example, we use the table shown in Table 8, which contains information about Australian license plates.

Seeding and SQL Query Generation: For each table, the pipeline selects one of two diversity injection methods with equal probability: reserved word seeding or persona seeding.

Reserved word seeding controls SQL complexity by specifying a target number of SQL reserved words for the generated query. This count is sampled from a probability distribution derived from BIRD's training set, so that the synthetic data mirrors the complexity distribution of human-authored queries. The count is passed directly into the generation prompt (Prompt 15), and chain-of-thought reasoning is used to help the model adhere to the constraint. For example, targeting three reserved words might produce: `SELECT "State/Territory" FROM "table-1" WHERE "Text/background Color" = 'black/white'`.

Persona seeding instead grounds generation in a specific user's goals and domain context. An LLM is prompted to generate a persona for the given table, for example, "License plate designer observing trends by region," along with associated goals (e.g., "To create the most sought-after license plate in Australia") and example queries (e.g., "Background color of Northern Territory vs. New South Wales"). The persona is randomly selected to be either a common professional type or a creative niche type to encourage diversity in register and terminology (Prompts 17 and 18). All three components — persona, goals, and example queries — are included in the SQL generation prompt (Prompt 16).

It is important to note that reserved word seeding only influences SQL generation, while persona seeding influences both SQL and NL generation, ensuring consistency between the query

State/Territory	Text/bg Color	Format	Current Slogan	Current Series	Notes
Australian Capital Territory	blue/white	Yaa·nna	ACT · CELEBRATION OF A CENTURY 2013	YIL·00A	Slogan screenprinted on plate
New South Wales	black/yellow	aa·nn·aa	NEW SOUTH WALES	BX·99·HI	No slogan on current series
New South Wales	black/white	aaa·nna	NSW	CPX·12A	Optional white slimline series
Northern Territory	ochre/white	Ca·nn·aa	NT · OUTBACK AUSTRALIA	CB·06·ZZ	New series began in June 2011

Table 8: Example table used in the pipeline walkthrough, drawn from WikiSQL. Used for demonstration purposes only.

and the question. All generated queries are executed against the database at this stage; any query that fails to execute is discarded.

For the remainder of this example, we assume reserved word seeding produced: `SELECT "State/Territory" FROM "table-1" WHERE "Text/background Color" = 'black/white'`, (ie, selecting states for which the background color of the license plate is black and white).

NL Question Generation: The validated SQL query and table schema are used to generate a natural language question. When persona seeding is used, the persona and goals are also included, and the model is instructed to phrase the question in the voice of that persona, adjusting formality, domain-specific vocabulary, and whether the user would refer to columns by name or by colloquial proxy (Prompt 20). Without persona seeding, a standard chain-of-thought prompt is used (Prompt 19). In both cases, four candidate questions are generated simultaneously, and a compare judge selects the best one based on naturalness, accuracy, and completeness (Prompt 21). For our example, the selected question might be: "Which states use black and white as their license plate colors?"

Ambiguity Detection and Handling: The NL question is scored by a DistilBERT-based ambiguity model trained on Ambrosia data (Saparina and Lapata, 2024), producing a continuous score from 0 to 10. This score determines how the example proceeds through the next stage.

If the score is 7 or above, the question is considered potentially ambiguous and enters the multi-interpretation branch. A prompt-based filter (Prompts 23 and 24) first confirms whether the ambiguity is truly present. Specifically, it tests whether it falls into one of four categories: Scope, Attachment, Entity Vagueness, or Broadness. The first three categories and their accompanying examples in the prompt are adapted from Saparina and Lapata (2024). Broadness is an additional category we introduce, inspired by Floratou et al. (2024), to capture cases where the NL is simply underspecified (e.g., "best player," where it is unclear whether this means most points scored or highest rating). If the filter determines the question is not genuinely ambiguous, the example is discarded. If ambiguity is confirmed, the pipeline attempts to generate a second SQL query representing a distinct valid interpretation of the same question (Prompt 25). The model must also identify the ambiguity type and explain how the new query differs semantically from the original. The new query is tested for executability, and then passed through the same prompt filter again to verify that it is a genuinely different valid reinterpretation rather than a superficial variant. The pipeline makes up to three attempts to produce an acceptable alternative; if none succeed, the example proceeds as a standard unambiguous pair using only the original query.

For our example, "Which states use black and white as their license plate colors?" is ambiguous: "black and white" could mean plates that are both black and white together, or plates that are either black or white. The two valid SQL interpretations would be an `AND` condition and an `OR` condition respectively, both of which are executable and semantically distinct.

If the score is below 7, the example skips the multi-interpretation branch and proceeds directly to the alignment model, which assigns a score between 0 and 1 reflecting how well the NL question and SQL query correspond to each other. Examples scoring 0.6 or above are retained. Note that ambiguous examples do not go through the alignment reward model, as this model was not trained to handle ambiguity. This is why they are filtered separately using prompts.

Final Quality Filter: Regardless of which path an example took through the ambiguity stage, all examples pass through a final prompt-based filter (Prompt 26) before being added to the dataset. This filter checks three things independently: that the SQL is a valid interpretation of the NL, that the SQL is non-redundant and well-formed, and that the NL sounds like something a real user would ask. This step catches any remaining low-quality examples that passed the alignment model but still

contain subtle issues, such as alligned pairs where the question did not make sense to begin with (ex. taking the sum of a categorical column), the SQL is overly convoluted, or any other issues.

A.2 Error Analysis

We conducted a failure mode analysis on 100 randomly sampled incorrect predictions from the Qwen 2.5-3B model finetuned on our pipeline's generations, evaluated on the Spider test set. The following taxonomy and occurrence counts were derived from this sample.

Taxonomy. We identify nine error categories:

Acceptable variation: The predicted SQL is semantically equivalent to the gold query or represents a valid alternative interpretation of the natural language question. This includes cases where an ORDER BY clause is added but not requested, an additional column is returned that is directly implied by the NL question (e.g., the question asks which player scored the most goals and the query returns both player name and goal count), a redundant but non-harmful condition is present, or minor syntactic differences exist that do not affect the result set. These cases would fail string match or execution match but have no meaningful impact on the user's informational need.

Spurious predicate hallucination: The model introduces a WHERE clause condition that has no basis in the natural language question. Example: the question asks for all employees in a department, but the query adds `WHERE status = 'active'` with no such qualifier in the NL.

Result boundary error: The model produces the correct logic but returns too much or too little. For example, limiting to 5 results when the user implies all results, or applying incorrect offsets.

Schema entity confusion: The model maps a natural language entity to the wrong table or column. Example: the question asks for "salary" but the query selects "bonus" because both are numeric compensation columns.

Logical negation failure: The model fails to capture a negation expressed in the NL (e.g., "not", "except", "other than"), producing a query that retrieves the complement of the intended set. Example: "Who did NOT win the award?" returns all winners instead of all non-winners.

Degenerate join structure: The model produces a join that is structurally malformed, redundant, or semantically incorrect. For example, joining a table to itself unnecessarily or using the wrong join key.

Cross-row constraint flattening: Questions like "both X and Y" require set intersection across rows, but the model instead treats it as a condition on a single row. Example: "Flights that took off from Florida and New York" becomes `WHERE takeoff='Florida' AND takeoff='New York'`, which can never be true on a single row.

Hardcoded quantifier substitution: The model sees a superlative such as "highest" or "fewest" and instead of computing the answer from the data, substitutes a hardcoded number. Example: "rated higher than average" becomes `WHERE rating > 4`, which will be wrong on any dataset where the true average differs.

NL misinterpretation: The model misreads the core intent of the NL question and produces a query that answers a related but distinct question. Example: "How many orders were placed last month?" is answered with a list of all orders rather than a count.

A.3 Complexity and Performance

Table 9 demonstrates our pipeline's model agnosticism: generating data with models ranging from Qwen-2.5-3B to Llama-3.3-70B produces similar SQL complexity and NL richness. Filtering leaves SQL complexity largely stable while modestly improving NL richness. Notably, the alignment reward model was trained on BIRD-derived pairs, yet filtering does not systematically shift complexity toward BIRD's distribution — this confirms the filter selects for NL-SQL coherence rather than acting as a proxy for BIRD-like complexity, confirming that using BIRD as both the target distribution and the source of alignment training data does not introduce circularity.

A.4 Cost Analysis

The cost estimates rest on several assumptions. For token counts, we estimate an average of 5 LLM calls per example based on the mandatory pipeline stages: SQL generation, NL generation, compare judge, and final prompt filter. Persona generation adds one additional call for approximately 50% of examples; we treat 5 as a conservative average across both persona-seeded and reserved-word-seeded examples. Input and output token counts were estimated by populating prompt templates with representative examples sampled from the middle of our output log distribution, measured

Model	SQL Comp.	NL Rich.
NF Qwen-3B	0.38	0.40
NF Llama-8B	0.43	0.37
NF Qwen-32B	0.37	0.37
NF Llama-70B	0.41	0.40
F Qwen-3B	0.38	0.41
F Llama-8B	0.39	0.43
F Qwen-32B	0.38	0.40
F Llama-70B	0.45	0.39

Table 9: SQL complexity and NL richness across different models (seeded with WikiSQL) before and after filtering, showing no significant differences between complexity. Before filtering is indicated by NF (No-Filtering) and after filtering is indicated by F (Filtering). All metrics were calculated on a sample size of 100 for each model. We used the following models: Qwen-2.5-3B, Llama-3.1-8B, Qwen-3-32B, and Llama-3.3-70B.

using the GPT-5 tokenizer, yielding 5,265 input tokens and 784 output tokens per example. Based on our reward model statistics and scaled quality measurements across models, we estimate a retention rate of 84%, meaning 6,548 examples must be generated to yield 5,500 kept examples. This gives a total of approximately 34.5 million input tokens and 5.1 million output tokens. At GPT-5.5 pricing ($5.00/1M input tokens, $30.00/1M output tokens), an equivalent API-based run would cost approximately $326. Local runtime was estimated assuming input prefill at approximately 2,000 tokens per second and output generation at approximately 50–80 tokens per second on a single commodity GPU running Qwen-2.5-3B, yielding approximately 136 GPU-hours and a cost of approximately $20 at $0.15/hr spot pricing (e.g., Vast.ai). This represents a 16× cost reduction, before accounting for the privacy implications of transmitting proprietary schemas to third-party APIs. The alignment reward model and DistilBERT ambiguity scorer are local model inference and contribute negligible additional cost. All API pricing reflects publicly listed rates at time of submission, and GPT 5.5 was chosen due to its status as a popular proprietary model at time of submission.

A.5 SQL Complexity Distribution Analysis

Figure 9 shows the SQL complexity distributions across datasets. Spider train and test sets exhibit nearly identical distributions (blue and purple overlapping curves), both centered around 0.2 complexity with minimal representation beyond 0.4. In contrast, our FALCON-generated data (orange) shows a distinct rightward shift with bimodal peaks at 0.35 and 0.45, and substantial density extending beyond 0.5. This distributional difference explains the cross-dataset performance gap in Table 7: Spider-trained models struggle on FALCON data because they lack exposure to queries in the 0.4-0.6 complexity range that dominate our synthetic dataset, while FALCON-trained models do not generalize as well to easier queries.

A.6 Alignment Model

Training Data Creation. We create a supervised dataset from BIRD SQL queries by generating candidate NL and CoT pairs for each query using Qwen-2.5-3B. To prevent overfitting to perfect alignments, we corrupt 50% of examples with 1-5 modifications per query (e.g., altering `WHERE` clauses, etc.) The resulting dataset contains over 9,000 examples (median alignment: 0.72, IQR: 0.54-0.90).

Alignment Score Computation. For each example, we compute a hybrid alignment score $y \in [0, 1]$ by comparing the generated question to BIRD's human-written reference question. Because BIRD questions align with their SQL queries, similarity to the reference indicates how well the generated question aligns with the SQL. Traditional semantic similarity (e.g., Sentence-BERT) often misses critical differences; For instance, "return the average" and "return the highest" may score similarly despite requiring different aggregations (AVG vs. MAX). Our metric combines two weighted components:

1. **Semantic Similarity (40%):** Sentence-BERT embeddings measuring linguistic proximity.
2. **Schema Entity Scoring (60%):** Extracts entities (table/column names, literal values) from the SQL query and detects their mentions in the NL question. Applies multiplicative penalties (0.7^n) for missing ground-truth entities or incorrect entities.

For corrupted examples, we apply an additional 10% penalty per corruption (Figure 10). We validate our computed alignment scores against human annotations on 101 examples, achieving strong correlation (Pearson correlation $r = 0.70$; Appendix 10). We train ModernBERT on this dataset, achieving MSE: 0.0203 and MAE: 0.1129 (Table 11). This alignment model outperforms other

filtering methods when applied to our main synthetic dataset (Table 2).

Human-annotators evaluated 101 examples using our rubric (See A.9 for rubric), mapping scores of 1, 2, and 3 to 0, 0.5, and 1 respectively. Our computed alignment scores using this metric achieved a Pearson correlation of 0.70 with human judgments. Some variance is expected when comparing our continuous scores (0-1) to discrete three-interval annotations.

Metric	Value
Correlation Metrics	
Pearson's r	0.701
Spearman's ρ	0.700

Table 10: Correlation between human evaluation scores and alignment metric on 101 SQL-to-NL pairs ($p < 0.001$). Human evaluators scored examples as 0, 0.5, or 1.0 according to our rubric (Table A.9). Alignment scores were computed using our hybrid similarity metric combining semantic and schema-entity matching.

We trained three model architectures on this dataset: RLVR with Qwen, regression-based CodeBERT, and regression-based ModernBERT. ModernBERT achieved the lowest error rates (MSE: 0.0203, MAE: 0.1129) with compact size (Table 11). We trained this model with learning rate $2e^{-5}$, batch size 64, and early stopping (patience: 5).

Model	MSE	MAE
RLVR w/ Qwen	0.0341	0.1506
FT + CodeBERT	0.0268	0.1262
FT + ModernBERT	**0.0203**	**0.1129**

Table 11: Alignment model performance on held-out test set.

A.7 Human Evaluation Across Seed Databases

We compare human evaluation across seed databases, showing that high human quality scores hold. Four annotators independently scored disjoint subsets of 100 examples each, with disagreements resolved through group discussion.

A.8 Ambiguity Model Training.

The ambiguity model is trained on Ambrosia-style data (Saparina and Lapata, 2024), where each question has (i) a canonical SQL query `gold_query`, (ii) additional valid interpretations `ambig_queries`, and (iii) a boolean `is_ambiguous` flag. We count distinct interpretations as $k_i = |\{\texttt{gold\_query}\} \cup \texttt{ambig\_queries}|$. This count maps to a score $y_i \in [0, 10]$: unambiguous questions receive $y_i = 0$, while ambiguous questions are mapped from k_i to $[5, 10]$ via min-max normalization. This ensures scores reflect interpretation count rather than schema-specific artifacts.

Model	SQL	NL	Alignment
WikiSQL	0.92	0.94	0.95
Spider	0.99	0.95	0.96

Table 12: Human evaluation scores across databases on Llama-3.1-70B.

Pipeline Integration. Predicted scores $\hat{y}$ influence two pipeline stages: (1) **Filtering** removes questions with extreme ambiguity scores (representing very unclear, low quality NL questions), and (2) **Multi-Interpretation Generation** triggers when $\hat{y} > 7$, generating a second plausible SQL query to mirror Ambrosia's structure and provide richer supervision for ambiguous questions.

A.9 Annotation Rubric

Found at Table 13.

A.10 Examples from the Dataset

We provide four representative examples from the FALCON-generated dataset to illustrate the range of query styles and complexity levels produced by the pipeline. (Figure 11, Figure 12, Figure 13, Figure 14)

A.11 Qwen 2.5-3B Finetuning and Results

Models are fine-tuned using LoRA (rank=16, α=32, dropout=0.05) for 3 epochs with learning rate 2×10^{-4}, effective batch size 16 (via gradient accumulation), and completion-only masking that trains solely on SQL outputs while masking instructions and schema. Training uses up to 5,500 examples sampled randomly when datasets exceed this size. Table schemas provided to the model include column names, types, primary keys, and foreign key relationships to connected tables. Models are evaluated on both our held-out synthetic test set (n=2,000) **and the Spider test set (n=2,147), a real human-authored benchmark entirely independent of our generation pipeline**. We do

Metric	Score	Description
SQL-NL Alignment	1.0	**Correct Match:** Covers all NL requirements. Minor relevant additions (e.g., adding a "points" column when asking for "who scored most", setting a limit value, etc.) are permitted.
	0.5	**Minor Issues:** Mostly correct but adds unrelated details (e.g., groups by category when not asked to) or misses 1-2 small parts of a complex request.
	0.0	**Incorrect:** Major omissions or fundamental misunderstanding of the NL intent.
SQL Quality	1.0	**Good:** Efficient structure and standard formatting.
	0.5	**Functional:** Query works but is slightly inefficient or unnecessarily complex.
	0.0	**Poor:** Query is nonsense, invalid, or extremely convoluted.
NL Quality	1.0	**Natural:** The question is sensical and sounds like human speech.
	0.5	**Awkward:** Understandable intent, but phrased in an unnatural or robotic way.
	0.0	**Nonsensical:** Logic errors (e.g., asking for the average of a categorical variable) or incoherent phrasing.

Table 13: The three-point scoring rubric used for human evaluation of synthetic NL-SQL pairs.

not evaluate on BIRD's test set because our alignment reward model was trained on BIRD-derived pairs, which would confound evaluation; Spider serves as our independent real-world generalization benchmark. All three training conditions — synthetic-only, Spider-only, and mixed — are evaluated on both test sets to enable direct cross-dataset comparison.

A.12 Qwen 2.5-1.5B Training and Results

We supplement our downstream task performance with results from a smaller model (Qwen 2.5-1.5B). Models are fine-tuned using LoRA (rank=16, α=32) for 2 epochs with learning rate 2×10^{-4}, effective batch size 16 (via gradient accumulation), and completion-only masking that trains solely on SQL outputs while masking instructions and schema. Training uses 5,500 examples with 2,000 held-out for testing. Due to the more compact model size, only relevant tables appearing in the gold query are included in the table schema. Models are fine-tuned for 2 epochs; loss continued to decrease at the end of training, suggesting additional epochs may further improve performance.

Model	Test Set	Total	EM %	FM %	FS	Exec %
Spider (5.5k)	Spider	2147	31.21	52.77	0.87	49.65
Spider (5.5k)	Pipeline	2000	0.1	3.2	0.66	26.72
Pipeline (5.5k)	Spider	2147	1.07	8.52	0.70	72.71
Pipeline (5.5k)	Pipeline	2000	9.1	42.55	0.83	84.11
Mixed 5% (5.5k)	Spider	2147	22.82	38.94	0.81	–
Mixed 5% (5.5k)	Pipeline	2000	9.2	42.8	0.83	–
Mixed 10% (5.5k)	Spider	2147	27.53	44.67	0.84	–
Mixed 10% (5.5k)	Pipeline	2000	9.65	44.05	0.83	–

Table 14: Performance of Qwen 2.5-1.5B LoRA variants (2 epochs) on Spider and FALCON test sets. Training data sizes shown in parentheses. **Metrics:** EM = Exact Match (string equivalence), FM = Fuzzy Match at 90%, FS = Fuzzy Score (average string similarity), Exec = Execution Accuracy (result set overlap). **Training data:** Spider = real Spider data, Pipeline = FALCON-generated data, Mixed = combined data (5% or 10% real Spider + remaining pipeline). Models trained on FALCON data excel on FALCON tests, while Spider-trained models excel on Spider tests, demonstrating training distribution effects.

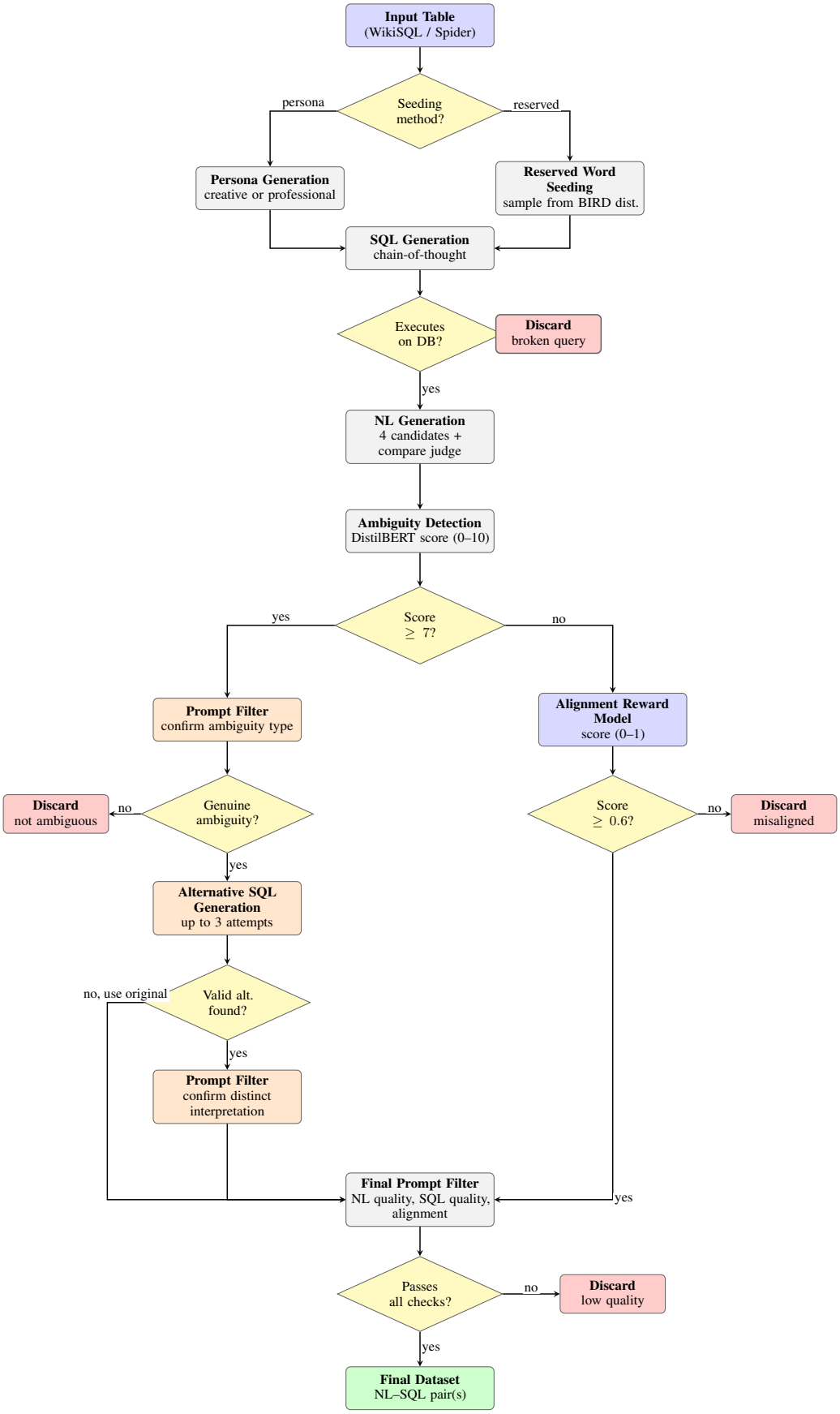


Figure 6: FALCON pipeline flowchart. Yellow diamonds indicate decision points. Orange boxes indicate ambiguity handling steps. Blue/purple boxes indicate alignment filtering. Green indicates the final dataset output. Red indicates discarded examples.

Running Example: Pipeline Walkthrough (Ambiguous)

Step 1 — Input Table (from WikiSQL): Australian license plate data including state/territory, color scheme, format, slogan, and series information.

Step 2 — Persona Generation

```
Persona: Sarah Chen, a graphic designer specializing in government branding, tasked with
modernizing Australia's license plate aesthetic.
Goals: Identify which states use the most visually distinct color schemes to inform a
redesign proposal.
Example queries: Which states use dark backgrounds? How many unique color combinations
exist across territories?
```

Step 3 — SQL Generation (Persona-Seeded)

```
SELECT 'State/Territory', 'Text/background Color'
FROM 'table-1'
WHERE 'Text/background Color' LIKE '%black%'
AND 'Text/background Color' LIKE '%white%';
```

Step 4 — NL Generation (4 candidates, best selected)

```
Reasoning: The SQL filters for plates whose color field contains both "black" and
"white", implying a two-tone scheme. The persona is a designer so the question should
reflect aesthetic interest.

Selected question: "Which states use black and white as their license plate colors?"
```

Step 5 — Ambiguity Detection

Ambiguity score: $7.4 \geq 7.0 \rightarrow$ enters multi-interpretation branch.

Prompt filter confirms: **Scope ambiguity** detected.
Reasoning: "black and white" is ambiguous — it could mean plates whose color scheme includes both black AND white together, or plates that are either black OR white.

Step 6 — Alternative SQL Generation

```
Interpretation 1 (original): plates that are both black and white
SELECT 'State/Territory' FROM 'table-1'
WHERE 'Text/background Color' LIKE '%black%'
AND 'Text/background Color' LIKE '%white%';

Interpretation 2 (generated): plates that are either black or white
SELECT 'State/Territory' FROM 'table-1'
WHERE 'Text/background Color' LIKE '%black%'
OR 'Text/background Color' LIKE '%white%';

Both queries execute successfully. Prompt filter confirms the second query is a
genuinely distinct semantic interpretation.
```

Step 7 — Final Quality Filter

```
Final filter reasoning: The NL question is clear and natural. Both SQL interpretations
are valid and non-redundant. Example passes all checks and is added to the dataset as an
ambiguous pair with two valid SQL interpretations.
```

Figure 7: Step-by-step pipeline walkthrough using a persona-seeded ambiguous example. Intermediate outputs are simplified for illustration purposes.

Running Example: Pipeline Walkthrough (Non-Ambiguous)

Step 1 — Input Table (from WikiSQL): Australian license plate data including state/territory, color scheme, format, slogan, and series information.

Step 2 — Reserved Word Seeding

```
Target reserved word count: 3 (sampled from BIRD distribution).
Reserved words selected: SELECT, FROM, WHERE.
```

Step 3 — SQL Generation (Reserved Word Seeded)

```
SELECT 'State/Territory' FROM 'table-1'
WHERE 'Text/background Color' = 'black/white';

Query executes successfully on the database.
```

Step 4 — NL Generation (4 candidates, best selected)

```
Reasoning: The SQL retrieves state names filtered by a specific color scheme. No persona
is used, so the question should be phrased naturally without domain-specific voice.

Selected question: "Which states have a black and white license plate color scheme?"
```

Step 5 — Ambiguity Detection

```
Ambiguity score: 2.1 < 7.0 → proceeds to alignment reward model.

No ambiguity detected: the question asks for a specific, well-defined color scheme with
no alternative interpretations.
```

Step 6 — Alignment Reward Model

```
Alignment score: 0.84 ≥ 0.6 → retained.

The NL question and SQL query are well-aligned: the question directly corresponds
to the filter condition in the SQL.
```

Step 7 — Final Quality Filter

```
Final filter reasoning: The NL question is clear, natural, and unambiguous. The SQL is
non-redundant and correctly captures the intent. Example passes all checks and is added
to the dataset as a standard NL-SQL pair.
```

Figure 8: Step-by-step pipeline walkthrough using a reserved-word-seeded non-ambiguous example. Intermediate outputs are simplified for illustration purposes.

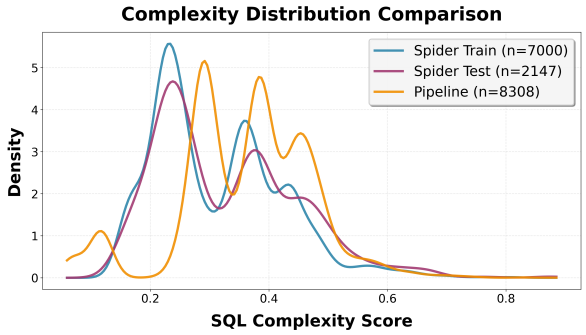


Figure 9: SQL complexity distributions for Spider train (n=7,000), Spider test (n=2,147), and FALCON-generated data (n=8,308). FALCON data exhibits higher complexity with bimodal peaks and extends substantially into harder query ranges (0.4-0.8) underrepresented in Spider.

Predicted NL: What unique user ID rates 'The Matrix' with a score of 1?" **True NL:** List the users who gave the worst rating for movie 'Love Will Tear Us Apart'."

Similarity Computation: 1. Sentence-BERT Similarity: 0.67 **2. Entity-Based Score:** 0.49

Missing true entity: Love Will Tear Us Apart" (penalty: $0.7^1 = 0.70$) Corrupted entity mentioned: The Matrix" (penalty: $0.7^1 = 0.70$) Entity score: $1.0 \times 0.70 \times 0.70 = 0.49$ **3. Weighted Similarity:**

$0.67 \times 0.4 + 0.49 \times 0.6 = 0.268 + 0.294 = 0.562$

4. Corruption Penalty: $0.562 \times 0.9 = \mathbf{0.506}$

Figure 10: Similarity scoring example. Despite high sentence similarity (0.67), entity penalties and corruption penalty reduce the final score to 0.506.

Example 1: Casual query.

Persona (shortened for brevity): Amanda Thompson is a travel blogger who uses storm record data to create safety guides for her readers.

NL: "What are the 5 regions that have experienced the most storms in the past 10 years, and what are their region names and codes, so I can warn my readers about these storm-prone areas?"

```
SELECT `region`.`region name`, `region`.`region code`,
       COUNT(`storm`.`storm id`) AS `storm frequency`
FROM `region`
JOIN `affected region`
     ON `region`.`region id` = `affected region`.`region id`
JOIN `storm`
     ON `affected region`.`storm id` = `storm`.`storm id`
WHERE `storm`.`storm date` > DATE('now', '-10 years')
GROUP BY `region`.`region name`, `region`.`region code`
ORDER BY `storm frequency` DESC LIMIT 5;
```

This example illustrates how persona seeding produces natural, conversational queries grounded in a believable real-world motivation. The trailing context ("so I can warn my readers") reflects how actual users frame database questions around their goals rather than as abstract data retrieval tasks, and the SQL faithfully captures the intent including the temporal filter and ranking.

Figure 11: Example 1: Casual query generated via persona seeding.

Example 2: Formal query.

Persona (shortened for brevity): Emilia Rodriguez is a senior real estate analyst at a property investment firm, responsible for identifying profitable investment opportunities.

NL: "Can I get a detailed report showing the property type descriptions, their average agreed selling prices, and the number of properties for each type, all filtered for properties with more than 5 rooms and selling prices over $1 million, and sorted by the average selling price in descending order?"

```
SELECT `rpt`.`property type description`,
       AVG(`p`.`agreed selling price`) AS `average_agreed_selling_price`,
       COUNT(`p`.`property id`) AS `number_of_properties`
FROM `properties` `p`
JOIN `reference property types` `rpt`
     ON `p`.`property type code` = `rpt`.`property type code`
WHERE `p`.`room count` > 5
  AND `p`.`agreed selling price` > 1000000
GROUP BY `rpt`.`property type description`
ORDER BY `average_agreed_selling_price` DESC;
```

This example demonstrates that the pipeline also produces formal, technically precise queries with complex filtering and aggregation. The persona provides a credible motivation for the specificity of the query: an analyst seeking to identify the highest-value property types within the premium market segment, explaining why the query scopes to large, high-price properties rather than the full dataset.

Figure 12: Example 2: Formal query generated via persona seeding.

Example 3: Ambiguous query with multiple valid interpretations.

NL: "How can I obtain a list of patients who had procedures costing more than $5,000, including their names, addresses, and hospital stay details, as well as the names of the physicians and nurses who cared for them?"

Interpretation 1 (per-procedure filter):

```
SELECT ‘patient‘.‘name‘, ‘patient‘.‘address‘,
       ‘stay‘.‘stay start‘, ‘stay‘.‘stay end‘,
       ‘procedures‘.‘name‘ AS ‘procedure_name‘,
       ‘physician‘.‘name‘ AS ‘physician_name‘,
       ‘nurse‘.‘name‘ AS ‘nurse_name‘
FROM ‘patient‘
JOIN ‘undergoes‘ ON ‘patient‘.‘ssn‘ = ‘undergoes‘.‘patient‘
JOIN ‘stay‘ ON ‘undergoes‘.‘stay‘ = ‘stay‘.‘stay id‘
JOIN ‘procedures‘ ON ‘undergoes‘.‘procedures‘ = ‘procedures‘.‘code‘
JOIN ‘physician‘ ON ‘undergoes‘.‘physician‘ = ‘physician‘.‘employee id‘
JOIN ‘nurse‘ ON ‘undergoes‘.‘assisting nurse‘ = ‘nurse‘.‘employee id‘
WHERE ‘procedures‘.‘cost‘ > 5000.0
ORDER BY ‘stay‘.‘stay end‘ - ‘stay‘.‘stay start‘ DESC;
```

Interpretation 2 (total cost filter):

```
SELECT ‘patient‘.‘name‘, ‘patient‘.‘address‘,
       ‘stay‘.‘stay start‘, ‘stay‘.‘stay end‘,
       ‘physician‘.‘name‘ AS ‘physician_name‘,
       ‘nurse‘.‘name‘ AS ‘nurse_name‘
FROM ‘patient‘
JOIN ‘undergoes‘ ON ‘patient‘.‘ssn‘ = ‘undergoes‘.‘patient‘
JOIN ‘stay‘ ON ‘undergoes‘.‘stay‘ = ‘stay‘.‘stay id‘
JOIN ‘procedures‘ ON ‘undergoes‘.‘procedures‘ = ‘procedures‘.‘code‘
JOIN ‘physician‘ ON ‘undergoes‘.‘physician‘ = ‘physician‘.‘employee id‘
JOIN ‘nurse‘ ON ‘undergoes‘.‘assisting nurse‘ = ‘nurse‘.‘employee id‘
WHERE ‘undergoes‘.‘patient‘ IN (
    SELECT ‘undergoes‘.‘patient‘ FROM ‘undergoes‘
    JOIN ‘procedures‘
         ON ‘undergoes‘.‘procedures‘ = ‘procedures‘.‘code‘
    GROUP BY ‘undergoes‘.‘patient‘
    HAVING SUM(‘procedures‘.‘cost‘) > 5000.0
)
ORDER BY ‘stay‘.‘stay end‘ - ‘stay‘.‘stay start‘ DESC;
```

The phrase "procedures costing more than $5,000" admits two valid interpretations: a per-procedure filter (any single procedure exceeds $5,000) or a total cost filter (the sum of all procedures for a patient exceeds $5,000). The pipeline detects this as a Broadness ambiguity and generates both SQL interpretations as valid supervision signal.

Figure 13: Example 3: Ambiguous query with two valid SQL interpretations.

Example 4: Reserved word seeding.

NL: "In 2016, which events were held in stadiums that can accommodate more than 60,000 people, listed in alphabetical order by stadium name?"

```
SELECT ‘stadium‘.‘name‘, ‘event‘.‘name‘
FROM ‘stadium‘
JOIN ‘event‘ ON ‘stadium‘.‘id‘ = ‘event‘.‘stadium id‘
WHERE ‘event‘.‘year‘ = ’2016’
  AND ‘stadium‘.‘capacity‘ > 60000
ORDER BY ‘stadium‘.‘name‘ ASC;
```

This example illustrates how reserved word seeding controls query complexity by grounding it in the distribution of reserved word counts observed in BIRD. The seeding value of 9 is sampled from this distribution, and the model is prompted to generate a query containing that many reserved words. The resulting SQL faithfully reflects the target complexity, joining two tables on a shared identifier and applying both a temporal filter and a capacity threshold before ordering results alphabetically by stadium name.

Figure 14: Example 4: Query generated via reserved word seeding.

SQL Generation with Reserved Word Seeding

```
You are an expert SQL developer tasked with generating a creative
and non-trivial SQLite query.

<database_schema>
{table}
</database_schema>

<task_requirements>
1. Source Material: Construct the query using ONLY information
from the <database_schema> provided above.
2. Logic: The query must be executable, logically sound, and
produce actionable insights.
3. Identifier Formatting: You must encase ALL column names in
backticks (‘) and all string literals in single quotes.
4. Formatting: Output a single SQLite query inside “‘sql
code blocks.
5. Syntax: End with a semicolon (;).
</task_requirements>

<constraint_checklist>
- The query must contain EXACTLY {length_seed} reserved words.
- Reserved words to choose from: {reserved_word_patterns}
- Do not use fewer or more than {length_seed}.
</constraint_checklist>

Step-by-step instructions:
1. Analyze the table schema.
2. Use exactly {length_seed} reserved words from the list.
3. Construct a query that uses exactly those reserved words
and answers a non-trivial question.
4. Verify backtick formatting.

Output ONLY the SQL query inside code blocks. No explanations.
```

Figure 15: SQL generation prompt with reserved-word seeding.

SQL Generation with Persona Seeding

```
You are an expert SQL developer. Generate a SQLite query that
matches the specific goals of the user persona described below.

<database_schema>
{table}
</database_schema>

<persona_profile>
*Persona:** {persona}
*Goals:** {goals}
*Inspiration Queries:** {example_queries}
</persona_profile>

<instructions>
1. Relevance: The query must answer a question this specific
persona would genuinely ask. Use the inspiration queries
as a guide for complexity, but do not copy them exactly.
2. Strict Formatting:
- Use valid SQLite syntax.
- Encase ALL column names in backticks (e.g., `Column Name`).
- All string literals in single quotes (e.g. 'New South Wales').
3. Output: A single executable SQL query inside "`sql tags,
ending with a semicolon.
4. Efficiency: Minimal steps to achieve the insight.
</instructions>

Output ONLY the SQL query. No natural language.
```

Figure 16: SQL generation prompt with persona seeding.

Out-of-the-Box Persona Generation

```
You are tasked with inventing a creative yet plausible persona
who interprets the provided database. This should be a person
who does not have a background in SQL or data analysis, but
wants to use the data for a relevant hobby, question, or other
job related use case.

<database_schema>
{table}
</database_schema>

<instructions>
Create a single persona. Their motivation must be grounded in
the data provided.

Output your response as a JSON object following this exact schema:
{
"persona_description": "Who they are and what makes them
unique (Max 1 sentence).",
"goals": "What they are trying to achieve (Imaginative
but plausible).",
"example_queries": [
"Query idea 1 (Natural language or SQL description)",
"Query idea 2",
"Query idea 3"
]
}
</instructions>
```

Figure 17: Creative persona generation prompt.

Common Persona Generation

```
You are tasked with generating a realistic and likely persona
who would query this database in a professional capacity.

<database_schema>
{table}
</database_schema>

<instructions>
Create a single, common persona. Avoid generic or previously
used personas.

Output your response as a JSON object following this exact schema:
{
"persona_description": "Role and background (Max 2 sentences).",
"goals": "What they are trying to analyze.",
"example_queries": [
"Practical query idea 1",
"Practical query idea 2",
"Practical query idea 3"
]
}
</instructions>
```

Figure 18: Common persona generation prompt.

Natural Language Question Generation

```
You are a natural language expert. Translate the SQLite query
below into 4 distinct natural language questions.

<context>
<database_schema>
{table}
</database_schema>

<sql_query>
{sql_query}
</sql_query>
</context>

<instructions>
Analyze the SQL intent and generate 4 questions that logically
lead to this query.
1. No SQL Leakage: Do not use technical terms (SELECT, GROUP BY)
or column names with underscores. Use natural English.
2. Accuracy: Do not infer details not present in the query.
Any implications should be clear and specified.
3. Variety: Phrase the questions differently (e.g., different
levels of formality or sentence structure).

Structure your thinking for EACH question:
1. Parse SQL Intent -> 2. Map to Schema -> 3. Draft Question
-> 4. Validate.

Output must be a single JSON object with this structure:
{
"reasoning1": "Analysis for Q1 (Max 4 sentences)",
"question1": "Natural language question 1",
...
}
</instructions>
```

Figure 19: NL question generation prompt with chain-of-thought.

Natural Language Question Generation with Persona Voice

```
You are a natural language expert. Translate the SQLite query
into 4 questions using the voice and style of a specific persona.

<context>
<database_schema>
{table}
</database_schema>

<sql_query>
{sql_query}
</sql_query>

<persona_profile>
*Persona:** {persona}
*Goals:** {goals}
</persona_profile>
</context>

<instructions>
Generate 4 natural language questions that this specific persona
would ask to get this data.
1. **Voice & Tone:** Mimic their likely jargon, casualness,
or strictness.
- Does this persona know SQL terms? Or do they use vague
business terms?
- Are they polite or demanding?
2. **Accuracy:** The question must be answerable by the provided
SQL. Any implications should be clear and specified. For
example, if the SQL filters on over 100 people, the question
should say something like "big rooms (>100 people)" or
something similar to make the implication clear.

Output must be a single JSON object:
{
"reasoning1": "How the persona views this data (Max 2 sentences)",
"question1": "The question in the persona's voice",
"reasoning2": "...",
"question2": "...",
"reasoning3": "...",
"question3": "...",
"reasoning4": "...",
"question4": "..."
}
</instructions>
```

Figure 20: NL question generation with persona-specific voice and tone.

Compare Judge for NL Candidate Selection

```
You are an expert evaluator of Natural Language to SQL datasets.

<task>
Select the best Natural Language Question from the options provided.
</task>

<inputs>
<database_schema>
{table}
</database_schema>

<sql_query>
{sql_query}
</sql_query>

<candidate_questions>
{nl_question}
</candidate_questions>
</inputs>

<evaluation_criteria>
1. Completeness: The question implies all filters/logic
present in the SQL.
2. No Hallucination: It does not ask for data absent from
the SQL.
3. Naturalness: It sounds like a human wrote it (no underscores,
no "SELECT *").
4. Independence: The question must make sense without seeing
the table schema.
</evaluation_criteria>

Output as JSON:
{
"index": <integer_index_of_best_question>,
"question": "<text_of_best_question>"
}
```

Figure 21: LLM-as-a-judge prompt for NL candidate selection. (Compare)

Score Judge for NL Candidate Selection

```
You are an expert natural language expert tasked with correcting a natural language question.
Check whether or not the natural language question is valid based on the SQLite query and table
information and provide a score between 0 and 100.

Your score should be based on the following criteria:
1. The natural language question should produce a question. It does not include the SQLite
query or table information.
2. The natural language question should be specific enough to be answered by the SQLite query.
3. The natural language question should not include details that the SQLite query does not
contain.
4. The natural language question should not include any information besides the question.
5. The natural language question should only be one singular natural language question.

Natural Language Question: {nl_question}

SQLite Query: {sql_query}

Table Information: {table}

Format your response as a JSON object with the following keys:
- "score": the score between 0 and 100
- "reasoning": the reasoning for the score

Example response:
“‘json
{{
"score": 80,
"reasoning": "The natural language question is valid based on the SQLite query and table
information but isn't specific enough to be answered by the SQLite query."
}}
“‘

Do not include any other text in your response besides the JSON object.
```

Figure 22: LLM-as-a-judge prompt for NL candidate selection. (Score)

Ambiguity Prompt-Based Filter (Part 1)

```
You are an expert in SQL interpretation and ambiguity detection.
<task>
Determine if the provided SQL query is a valid interpretation
of the potentially ambiguous Natural Language (NL) question.
</task>

<inputs>
<nl_question>
{nl_question}
</nl_question>
<sql_query>
{sql_query}
</sql_query>
<database_schema>
{table}
</database_schema>
<ambiguity_reasoning>
{reasoning}
</ambiguity_reasoning>
</inputs>

<ambiguity_types>
Only the following ambiguity types are considered valid:
1. **Scope**: Where a modifier or constraint applies
(e.g., global vs filtered scope)
- EXAMPLE: What activities does each gym offer?
Interpretations:
- **1. Show classes that are common to all gyms.**
- SELECT Classes.Name FROM ...
- GROUP BY HAVING
- COUNT (DISTINCT Gyms_Classes.GymID) =
(SELECT COUNT(*) FROM Gyms)
- **2. For each gym, show me the classes there.**
- SELECT Gyms.Name, Classes.Name FROM ...
2. **Attachment**: Which part of the sentence a modifier attaches to
(e.g., prepositional phrase attachment)
- EXAMPLE: Show the writers and editors on a work-for-hire.
Interpretations:
- **1. Which writers are contracted for a work-for-hire
and which editors have the same contract?**
- SELECT Name FROM Screenwriters
- WHERE Contract = 'Work-for-Hire'
- UNION SELECT Name FROM Editors
- WHERE Contract = 'Work-for-Hire'
- **2. Provide a list of all writers and a list of
work-for-hire editors.**
- SELECT Name FROM Screenwriters
- UNION SELECT Name FROM Editors
- WHERE Contract = 'Work-for-Hire'
3. **Entity VAGUENESS**: Which entity or concept is being referred to
(e.g., entity resolution, referential ambiguity)
- EXAMPLE: Who issued CD Special?
Interpretations:
- **1. Which bank issued CD Special?**
- SELECT Bank.Name FROM ...
- WHERE Products Name = 'CD Special'
- **2. Which branch issued CD Special?**
- SELECT Branch.Name FROM ...
- WHERE Products.Name = 'CD Special'
- **3. Find the bank and branch that issued CD Special.**
- SELECT Bank.Name, Branch.Name FROM ...
- WHERE Products.Name = 'CD Special'
```

Figure 23: Ambiguity prompt-based filter (Part 1).

Ambiguity Prompt-Based Filter (Part 2)

```
4. **Broadness**: Sentence is vague and could be interpreted
in multiple ways.
- EXAMPLE: Show me some stats on the best player
Interpretations:
1. Best player = most runs. stats = batting average, etc.
2. Best player = highest draft pick. stats = position, etc.
3. Best player = most wins. stats = wins, losses, etc.
many more
</ambiguity_types>

Here are a few more examples of ambiguity:
NL: Can you give me a breakdown of Terry Labonte's teams,
average start and finish positions, and the most top 5
finishes he had with each team between 1990 and 1999?
Ambiguity explanation: not clear if "between 1990-1999" applies
to all conditions or just the last one
NL: What are the titles and directors of episodes that premiered
in February and were written by writers who have worked on
multiple episodes?
Ambiguity explanation: could be ones that BOTH premiered in
February and were written by multi-episode writers, or EITHER
condition
NL: Which NHL teams have the most Canadian players drafted from
the OHL or WHL in the 1994 Entry Draft, and what positions
do they play?
Ambiguity explanation: could mean both OHL 1994 or WHL 1994,
or OHL (in general) vs WHL 1994

Here are a few more examples of NOT valid ambiguity types:
NL: What is the total number of goals scored by George Utley
during his time at Barnsley Football Club?
Ambiguity explanation: No valid alternative way to interpret.
NL: Which players have the best singles win-loss ratio in the
Lithuania Davis Cup team?
Ambiguity explanation: No valid alternative way to interpret.
NL: Which year between 2005 and 2011 had the highest BBC Two
viewership for Comic Relief?
Ambiguity explanation: No valid alternative way to interpret.
The phrase "that year" makes the reference unambiguous.

*REJECT** the following as NOT valid ambiguity types:
- Pattern matching variations (LIKE '%X%' vs LIKE 'X%')
- Conditional priority differences
- Implementation style differences (subquery vs JOIN)
- Grouping semantics differences (unless true Scope/Entity)
- Table/schema confusion
- Minor formatting or syntax variations

<scoring_criteria>
- **1.0 (Valid):** SQL represents a valid semantic interpretation
due to Scope, Attachment, Entity, or Broadness ambiguity.
- **0.0 (Invalid):** SQL contradicts the NL, differs only in
implementation style, or is not a valid ambiguity type.

*Key requirement**: Ambiguity must be genuine and semantic.
Implementation differences should be scored 0.0.
</scoring_criteria>

Output as JSON:
{
"score": <float>
}
```

Figure 24: Ambiguity prompt-based filter (Part 2).

Multi-Interpretation SQL Generation

```
You are an expert SQL developer.

<task>
The provided NL question is ambiguous. The first SQL query
below provides ONE interpretation. Your job is to identify the
type of ambiguity and then generate a DIFFERENT SQL query that
provides a SECOND valid interpretation.

<ambiguity_types>
Valid ambiguity types:
1. Scope: Where a modifier or constraint applies
2. Attachment: Which part of the sentence a modifier attaches to
3. Entity Vagueness: Which entity or concept is being referred to
4. Broadness: Sentence is vague and could be interpreted in
multiple ways
</ambiguity_types>
</task>

<inputs>
<nl_question>
{nl_question}
</nl_question>

<first_interpretation_sql>
{first_sql_query}
</first_interpretation_sql>

<database_schema>
{table}
</database_schema>
</inputs>

<instructions>
1. Analyze the ambiguity (Scope, Attachment, Entity Vagueness,
or Broadness) and state which type is present or none
2. Construct a new SQL query that interprets the question
differently than the first query (or "No ambiguity found"
if none of the types are present).
3. Adhere to strict formatting: Use SQLite syntax, enclose
columns in backticks, end with semicolon.
</instructions>

Output as JSON:
{
"reasoning": "Identify the ambiguity and how this new query
interprets it differently (Max 1 sentence).",
"sql_query": "The new SQLite query"
}
```

Figure 25: Multi-interpretation SQL generation for ambiguous queries.

Final Prompt-Based Quality Filter

```
You are an expert in SQL interpretation.
Your job is to determine if:
1. The SQL query is a valid interpretation of the NL question
2. The SQL query makes sense and is not unnecessarily redundant
3. The NL makes sense and matches something an actual human would ask

EXAMPLES:
Fail example:
NL: Can you list the spacious classrooms in rooms 143, 547,
or 700, starting with the ones that can hold the most students?
SQL: SELECT ‘building‘, ‘room number‘ FROM ‘classroom‘
WHERE ‘capacity‘ > 40 AND ‘room number‘ IN (’143’, ’547’, ’700’)
ORDER BY ‘capacity‘ DESC LIMIT 5;
This should be REJECTED (score 0.0) because the NL is weird.
The "spacious" meaning capacity > 40 is not easily inferable
and overall the NL just sounds confusing and wordy.
NOTE: Sometimes implications are ok. For example, if it said
"selecting only rooms with large capacities", that would
reasonably connect to a filter on capacity > X. But just
adding one adjective to a question is not enough to be
considered valid.

Pass example:
NL: Can you give me a detailed report on the average height,
average weight, and number of perpetrators for each hometown,
sorted in descending order by the number of perpetrators, so
I can identify key trends and patterns?
SQL: SELECT ‘home town‘, AVG(‘height‘) AS ‘average_height‘,
AVG(‘weight‘) AS ‘average_weight‘, COUNT(*) AS
‘number_of_perpetrators‘ FROM ‘people‘ GROUP BY ‘home town‘
ORDER BY ‘number_of_perpetrators‘ DESC;
This should be ACCEPTED (score 1.0) because the NL is clear
and the SQL makes sense. The NL implies that we are looking
for the average height, average weight, and number of
perpetrators for each hometown, sorted in descending order
by the number of perpetrators. The SQL does this and runs
successfully on the database schema.

If all of the above are true, then the score should be 1.0,
otherwise it should be 0.0.

NL: {nl}
SQL: {sql}
Database schema: {database_schema}

Output as JSON:
{
"score": <float>,
"reasoning": "Why the score is what it is (max 2 sentences)"
}
```

Figure 26: Final prompt-based quality filter verifying NL-SQL alignment, SQL quality, and NL naturalness.

When Relevance Is Not Evidence: A Controlled Diagnostic of Claim-Conditioned Support Ranking

Ke Lu
Independent Researcher
look970726@gmail.com

Abstract

Retrieval-augmented generation ranks passages by topical relevance or semantic similarity, but generation needs passages that actually *support* a target claim. We study when these targets come apart, framing the task as **claim-conditioned support ranking over a constructed candidate shortlist**: given a query, an explicit target claim, and a small fixed set of candidates, rank them by evidential support. We build a controlled, typed development set of **175 controlled contrastive examples** spanning seven spurious-evidence cues, each pairing one supporting passage with one targeted non-support passage expressing the designated cue, plus neutral distractors. We evaluate with a **corrected, tie-aware** protocol that uses role-free evaluator-facing opaque identifiers and neutral tie handling. Tested query-only relevance rankers (lexical, dense, and a cross-encoder) yield tie-aware focal-pair error rates of roughly 18–34%; in the BM25 cue decomposition, several strata have high error, including negation and relation directionality; a gold-claim NLI reference—which receives information the rankers do not—reduces but does not eliminate the error. We argue that evidence-ranking evaluation should explicitly *condition on the claim* whose support is assessed, and sketch a typed, claim-conditioned evaluation direction that separates relevance, support, and source attributes. This is a controlled diagnostic on development data: it does not estimate real-world prevalence or provide a deployable method.

1 Introduction

Retrieval-augmented generation (RAG) grounds language models in retrieved passages (Lewis et al., 2020), but retrieval is optimized for topical *relevance* or semantic *similarity*, whereas faithful generation requires passages that provide evidential *support* for the specific claim being made. These targets are related but not identical: a passage can be highly similar to a query—sharing its entities, numbers, or phrasing—yet fail to support the intended answer, or even contradict it, because of negation, a reversed relation, a different entity, or a stale value. For instance, a passage asserting a cipher *is* permitted shares nearly all terms with the claim that it is *not*, yet must rank below genuine support (§2).

We isolate this distinction with a controlled diagnostic. The relevance/support distinction and entailment-aware retrieval are not new (§4); our aim is to isolate *when* relevance-oriented rankers prefer high-overlap non-evidence to genuine support. We frame **claim-conditioned support ranking over a constructed candidate shortlist**: given a query, an explicit target claim, and a small fixed candidate set, a system ranks candidates by evidential support for the claim. This is deliberately *not* corpus-scale first-stage retrieval; it is a controlled reranking probe that lets us build controlled contrastive pairs and read off pairwise errors directly. We use this diagnostic to ground a broader, forward-looking argument: evidence-ranking evaluation should condition on the claim whose support is assessed and should separate relevance from support and source attributes (§5).

Because different systems may see different information, we label four **information-access conditions** and keep them separate throughout: **(A)** query-only relevance ranking; **(B)** generated-claim-conditioned support ranking (a deployable setting we do *not* implement here); **(C)** gold-claim-conditioned diagnostic ranking; and **(D)** a gold-label oracle. Comparing Condition A with Condition C is an *unequal-information* comparison: it diagnoses the value of explicit claim access plus a support signal, not model architecture alone.

Our contributions are:

1. a precise formulation of claim-conditioned support ranking over a constructed shortlist,

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 88–93
October 28, 2026 ©2026 Association for Computational Linguistics

with an explicit Condition A/C/D information-access distinction (§2);

2. a **corrected, tie-aware evaluation** for this setting—role-free evaluator-facing opaque identifiers, neutral tie handling, tie-aware pairwise error, and expected rank metrics—that removes a role-correlated tie-break present in the prior implementation (§2);
3. a **cue-decomposed diagnosis** on 175 controlled development examples: all tested query-only relevance rankers exhibit nonzero focal-pair error, while several BM25 strata show high error (negation, directionality, and redundancy/majority), with a gold-claim NLI model included as a diagnostic reference (§3).

Scope. All results are on *constructed development data* whose labels were produced and audited by the same project (§2). We make no claim about real-world prevalence, all-retriever generality, or a deployable mitigation, and we present no downstream generation results here (§6).

2 Controlled Diagnostic Design

Shortlist structure. Each example is a query with an explicit target claim (the query's intended answer expressed as a proposition) and a fixed shortlist of five candidate passages: one *supporting* passage, one *targeted non-support* passage designed to preserve cue-relevant query/entity/attribute signals without supporting the claim, and three neutral distractors. The supporting and non-support passages form a **controlled contrastive pair**. Negation and directionality often instantiate near-minimal surface edits; other cues preserve targeted query/entity/attribute overlap while varying referent, time, source style, or redundancy. Formally, given a query q, target claim c, and candidates $P = \{p_1, \ldots, p_5\}$ with a supporting p^+ and targeted non-support p^-, a query-only ranker scores $f(q, p)$ and a claim-conditioned ranker $g(q, c, p)$; the diagnostic asks whether p^- outranks p^+.

Cue types. The 175 examples are balanced across seven cue types (25 each) describing *why* the non-support passage is superficially attractive: entity overlap, number overlap, temporal mismatch, negation/polarity, relation directionality, authority style, and redundancy/majority. The examples are *self-contained*—each shortlist supplies all passages needed to judge support—and use either public facts or explicitly fictional mini-worlds.

Worked minimal pair (negation, fictional). To illustrate the self-contained design without external knowledge, we use an invented mini-world. *Claim:* "Under the Meridian Data Standard, the CX-2 cipher is not permitted for storing customer records." *Supporting:* "The Meridian Data Standard states that CX-2 must not be used to store customer records; systems are required to use CX-4 or higher." *Non-support (negation):* "Under the Meridian Data Standard, CX-2 is a permitted cipher for storing customer records." The non-support passage shares nearly all terms with the claim but asserts the opposite polarity, so lexical and similarity signals rank it alongside—or above—the supporting passage.

Label provenance (important caveat). The examples were constructed with *intended* support and non-support roles and subsequently subjected to a project-internal audit under a two-axis relation/attribute protocol. This is a **development quality-control process, not independent human reliability validation**: because the same project created and audited the examples, construction bias remains possible. The audit found no obvious malformed records but flagged temporal-scope ambiguity and construction confounds (e.g., the authority-style non-support passages are also contradictions) that would require independent annotation to resolve. We therefore treat labels as development annotations and report descriptive findings, not out-of-sample generalization.

Corrected, tie-aware metrics. Our unit of analysis is the support/non-support pair. **Spurious Win Rate (SWR)** is the mean over examples of 1 if the non-support passage outscores the supporting one, 0.5 on an exact score tie, and 0 otherwise; it is score-based and therefore invariant to how ties are ordered. **Expected Support@1** is the probability that a supporting passage is ranked first under uniform random tie-breaking within equal-score groups (computed exactly by enumeration for these five-candidate shortlists). We also report the support/non-support **tie rate**.

These corrections matter: the prior implementation broke score ties using role-bearing candidate identifiers, which alphabetically favored the non-support passage and inflated the headline error. Using role-free opaque identifiers and tie-aware scoring, BM25's pairwise error falls from a strict role-resolved $57/175{=}0.326$ to a tie-aware 0.280 (16 exact ties), and the same artifact depressed Sup-

System	Info	SWR↓	E[S@1]↑
Query-only relevance (Condition A)			
Lexical overlap	query	0.246	0.744
BM25	query	0.280	0.697
Dense E5-small	query	0.274	0.726
Dense MiniLM	query	0.337	0.663
CE relevance	query	**0.177**	**0.823**
Gold-claim diagnostic reference (Condition C)			
BM25 + NLI	query+claim	0.086	0.909
Gold-label upper bound (Condition D)			
Oracle	gold labels	0.000	1.000

Table 1: Corrected tie-aware Spurious Win Rate (SWR; lower is better) and expected Support@1 (E[S@1]; higher is better) on the 175-example controlled development set. "CE relevance" is a relevance cross-encoder; "query+claim" denotes the *gold* target claim. **Conditions C and D receive information unavailable to Condition-A systems and are diagnostic references, not input-equivalent baselines.** Only the two lexical systems have exact support–non-support focal-pair score ties (rates 0.091 BM25, 0.297 lexical overlap), so their SWR requires the tie-aware treatment. Because each cue contributes 25 examples, aggregate metrics are balanced diagnostic averages, not estimates under a natural cue distribution.

port@1. All SWR numbers below use the corrected metric; the continuous-score systems have no exact support–non-support focal-pair ties, so their SWR values are unchanged.

3 Results

We evaluate five query-only rankers—lexical overlap, BM25, two dense retrievers (`all-MiniLM-L6-v2`, `e5-small-v2`), and a relevance cross-encoder (`ms-marco-MiniLM-L-6-v2`)—and, as diagnostic references, a gold-claim NLI reranker (`nli-deberta-v3-small`; He et al., 2023) and a gold-label oracle.

Query-only relevance rankings exhibit systematic support-ordering errors. When evaluated under the support objective, every tested Condition-A system exhibits nonzero focal-pair error under the tie-aware metric (Table 1): SWR ranges from 0.177 (a supervised relevance cross-encoder) to 0.337 (dense MiniLM). The lexical and dense baseline point estimates are broadly similar on this development set, and we do not interpret their ordering. The strongest query-only system—the cross-encoder relevance reranker—still errs on roughly one pair in six. SWR isolates the focal support–non-support ordering, whereas expected Support@1 can additionally reflect how the supporting passage ranks against the three neutral candidates.

Cue type	BM25 (A)	BM25+NLI (C)
number overlap	0.00	0.00
entity overlap	0.08	0.00
temporal mismatch	0.08	0.04
authority style	0.24	0.00
redundancy/majority	0.48	0.16
relation directionality	0.50	0.28
negation/polarity	0.58	0.12

Table 2: Per-cue tie-aware SWR (↓), 25 examples per cue. For the query-only ranker (BM25) errors are highest for **negation**, **relation directionality**, and **redundancy/majority**. The gold-claim NLI reference (Condition C) reduces every cue with nonzero BM25 error, drives entity and authority to zero, and leaves number at zero, with the largest residual in **directionality**. Cue rows are descriptive construction strata, not causal-effect estimates.

BM25 errors are highest for negation, directionality, and redundancy. Table 2 decomposes BM25 by cue. BM25 errors are highest for negation (0.58) and directionality (0.50), with the passage-level redundancy/majority stratum close behind (0.48), and near zero for number and entity overlap. Because that redundancy/majority stratum is not a controlled set-level majority manipulation, we do not interpret it as a majority-effect estimate (§6). Negation and directionality provide the clearest semantic contrast mechanisms, flipping the claim's polarity or reversing its relation while sharing almost all of its terms.

A gold claim plus NLI helps, under unequal information. The Condition-C reference reranks the fixed five candidates by fusing the min–max-normalized BM25 relevance score rel with cross-encoder NLI probabilities (premise=passage; the hypothesis is a question–answer serialization of the gold claim, `Question: {q}\nAnswer: {a}`, where `a` is the gold answer associated with the target claim): $s = \mathrm{rel} + 2\,p_{\mathrm{ent}} - 2\,p_{\mathrm{con}} - p_{\mathrm{neu}}$. It lowers aggregate tie-aware SWR to 0.086, reduces all six cues with nonzero BM25 error, and leaves three cues at zero (Table 2). This reference receives information the Condition-A rankers do not, so the gap measures claim access plus a support signal, *not* a better retriever. Even so, a residual error remains: the largest residual under this NLI reference is relation directionality (0.28). The oracle (Condition D) reaches 0 by construction and only bounds

the setting.

4 Discussion

What this study does and does not show. On this controlled, cue-typed development set, tested query-only rankers exhibit support-ordering errors. In the BM25 cue decomposition, these errors are concentrated in negation and directionality, where the contrastive pairs are closely matched, and in the passage-level redundancy stratum (Table 2). These systems optimize query relevance, not claim support, so this is an *objective mismatch*, not a failure of relevance ranking on its own terms; a gold-claim NLI signal reduces but does not eliminate the observed errors.

Related work. The relevance/support distinction is long-standing, from the PASCAL RTE challenge (Dagan et al., 2006) to natural-logic accounts of surface versus semantic inference (MacCartney and Manning, 2007). It is also central to modern work: fact verification retrieves and labels evidence for a claim (Thorne et al., 2018), natural-language inference models entailment and contradiction (Bowman et al., 2015; Williams et al., 2018), attribution work asks whether generated statements are supported by cited sources (Rashkin et al., 2023; Gao et al., 2023), and knowledge-conflict studies examine models facing contradictory evidence (Longpre et al., 2021; Xie et al., 2024; Xu et al., 2024). Most closely, prior work uses claim–evidence entailment to improve evidence retrieval (Yang et al., 2021), learns support retrievers conditioned on question–answer pairs (Zhang et al., 2025), and systems such as SIDE rank sources by how well they support a claim (Petroni et al., 2023); Alt et al. (2026) rank evidence for sufficiency and non-redundancy, and utility-oriented retrieval such as SCARLet (Xu et al., 2025) optimizes downstream usefulness. Embedding retrieval also has *representational* limits—for a fixed dimension, some top-k sets are unreturnable by any query (Weller et al., 2026)—whereas our failures are task-level, arising even when support is present. Methodologically, our contrastive pairs follow controlled diagnostics that expose hidden failures—lexical-substitution NLI sets (Glockner et al., 2018) and overlap heuristics (McCoy et al., 2019)—and are closest to VitaminC (Schuster et al., 2021), whose contrastive pairs differ in support; unlike its verification *labeling*, we score claim-conditioned *ranking* with a tie-aware metric. Support-aware, answer-conditioned, and entailment-based ranking are therefore *not* new, and we claim neither them nor the relevance/support distinction itself. Our contribution is a *controlled, cue-typed ranking diagnostic* that localizes support-ordering failures across controlled cue types, under a corrected tie-aware evaluation separating query-only relevance (Condition A) from gold-claim support (Condition C).

5 Toward Claim-Conditioned Evidence-Ranking Evaluation

The diagnostic grounds an evaluation direction for knowledge-grounded generation, bearing on evidence selection, provenance, and verification. On this controlled set, relevance-oriented rankers systematically exhibit cases in which high-overlap non-evidence outranks genuine support; adding gold-claim access plus an NLI signal reduces but does not eliminate these errors. **(1) Relevance is an inadequate sole objective for evidence selection**: our controlled contrastive pairs show that relevance-oriented ranking does not reliably separate support from high-overlap non-support. **(2) Evaluate against explicit claims**: evidence ranking should be scored relative to the specific claim, and each system's information access (query-only vs. claim-conditioned) stated, since these settings are not comparable. **(3) Typed controlled diagnostics can complement end-to-end RAG evaluation**: cue-decomposed controlled contrastive pairs localize failure modes (here, in BM25's cue decomposition; Table 2) that aggregate scores obscure. **(4) Separate the axes**: benchmarks and systems should distinguish query relevance, claim support, source attributes, and provenance rather than collapsing them into a single relevance score. Contradictory passages can be valuable counterevidence for verification; SWR measures whether a passage supports the designated target claim, not its overall usefulness for verifying or challenging that claim, so a claim-aware system should expose support and contradiction as distinct relations. We do *not* claim that structured knowledge bases or entailment models resolve this. A claim-aware, axis-separated view of evidence ranking provides a clearer basis for evaluating evidence selection in knowledge-grounded generation.

6 Limitations

This is a controlled diagnostic, and its scope is correspondingly narrow. **(i)** Systems rank *constructed five-candidate shortlists*, not corpus-scale retrieval; SWR is a focal-pair quantity, whereas Support@1 is a five-candidate shortlist quantity; neither measures corpus-scale retrieval recall. **(ii)** The 175 examples are *development/legacy data*: the same project developed the cue taxonomy and metrics on this family (seed $\subset$ 70 $\subset$ 175), so results are descriptive, not out-of-sample estimates. The benchmark is intentionally small and controlled rather than representative of open-domain retrieval traffic: its purpose is to test whether specific confounding cues can induce ranking failures under tightly matched conditions, not to estimate their prevalence in deployed RAG systems. **(iii)** Labels were constructed and audited within the same project, rather than independently human-validated; construction bias remains possible, and temporal-scope and authority/contradiction confounds are unresolved. **(iv)** The legacy construction places *exactly one* supporting and *one* targeted non-support passage per shortlist, which cannot represent multiple or partial support; relatedly, the redundancy/majority cue is passage-level and does not constitute a controlled set-level majority manipulation. **(v)** Cue type is partially confounded with relation label by construction—for example, the negation, directionality, authority, and legacy redundancy strata use contradictory targeted passages—so per-cue differences should not be interpreted as isolated cue effects. **(vi)** We test a *limited set of local models* and one generic NLI model. **(vii)** The NLI reference uses a *gold* claim (Condition C); a deployable generated-claim setting (Condition B)—draft an answer, extract its claims, retrieve and rerank support and counterevidence, then revise or abstain—is proposed future work, not implemented or evaluated here (cf. SIDE (Petroni et al., 2023) for an editor-facing retrieval-and-verification system). Condition C also uses fixed, hand-chosen fusion weights set on an earlier 70-example seed subset and later examined, but not re-tuned, on the 175-example development family; it should therefore be read as an *in-development diagnostic reference*, not a held-out estimate of deployable performance. **(viii)** We report *no* downstream generation results: an earlier Flan-T5 probe used contexts that changed under the corrected tie handling and has not been rerun, so downstream propagation remains an open validation step. Whether these ranking errors propagate into generation errors depends on downstream model behavior and remains an important direction for future evaluation. We do not argue these limitations are harmless; rather, the justified conclusion is limited to the controlled, development-set diagnostic above.

Acknowledgments

We used generative AI assistance (Anthropic Claude) for literature search, research framing, construction of controlled diagnostic examples, LLM-assisted label quality control, design and implementation of evaluation metrics, code generation and debugging, experiment planning, analysis, and manuscript drafting and revision. Reported results and references were cross-checked against the released artifacts and cited sources. The author reviewed the study design and final manuscript and takes full responsibility for the work.

Code, data, and result artifacts are available at `https://github.com/kelu-look/when-relevance-is-not-evidence` (release tag `v1.0-akbc2026`).

References

Guy Alt, Eran Hirsch, Serwar Basch, Ido Dagan, and Oren Glickman. 2026. User-centric evidence ranking for attribution and fact verification. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7215–7237, Rabat, Morocco. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment (MLCW 2005)*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6465–6488.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that

require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations (ICLR)*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7052–7063.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200, Prague. Association for Computational Linguistics.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Fabio Petroni, Samuel Broscheit, Aleksandra Piktus, Patrick Lewis, Gautier Izacard, Lucas Hosseini, Jane Dwivedi-Yu, Maria Lomeli, Timo Schick, Michele Bevilacqua, Pierre-Emmanuel Mazaré, Armand Joulin, Edouard Grave, and Sebastian Riedel. 2023. Improving Wikipedia verifiability with AI. *Nature Machine Intelligence*, 5:1142–1148.

Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. 2023. Measuring attribution in natural language generation models. *Computational Linguistics*, 49(4):777–840.

Tal Schuster, Adam Fisch, and Regina Barzilay. 2021. Get your vitamin C! robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 624–643, Online. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 809–819.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2026. On the theoretical limitations of embedding-based retrieval. In *The Fourteenth International Conference on Learning Representations (ICLR)*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1112–1122.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations (ICLR)*.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8541–8565.

Yilong Xu, Jinhua Gao, Xiaoming Yu, Yuanhai Xue, Baolong Bi, Huawei Shen, and Xueqi Cheng. 2025. Training a utility-based retriever through shared context attribution for retrieval-augmented language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 629–648, Suzhou, China. Association for Computational Linguistics.

Fan Yang, Eduard Dragut, and Arjun Mukherjee. 2021. Improving evidence retrieval with claim-evidence entailment. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1553–1558, Held Online. INCOMA Ltd.

Zeyu Zhang, Alessandro Moschitti, and Thuy Vu. 2025. Retrieving support to rank answers in open-domain question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 35098–35105, Suzhou, China. Association for Computational Linguistics.

Fine-Grained Table Retrieval Through the Lens of Complex Queries

Wojciech Kosiuk[1], **Xingyu Ji**[2], **Yeounoh Chung**[3], **Fatma Özcan**[3], **Madelon Hulsebos**[4*]
[1]University of Amsterdam, [2]UC Berkeley, [3]Google, [4]CWI

Abstract

Enabling question answering over tables and databases in natural language has become a key capability in the democratization of insights from tabular data sources. These systems first require retrieval of data that is relevant to a given natural language query, for which several methods have been introduced. In this work we present and study Decomposition-based Connectivity-aware Table Retrieval (DCTR), a table retrieval mechanism combining fine-grained typed query decomposition with schema connectivity signals for open-domain question answering over relational databases. We evaluate the effectiveness of the two mechanisms through the lens of *retrieval complexity* which we measure along the axes of query- and data complexity. Our analyses over industry-aligned benchmarks show that typed decomposition improves robustness for highly compositional queries, while connectivity-aware retrieval is most useful when schema structure provides informative join signals.

1 Introduction

Open-domain question answering and text-to-SQL over large heterogeneous table collections face a fundamental retrieval challenge: users rarely know which tables are relevant, and required join paths cannot be discerned from the query alone (Chen et al., 2025c). Ambiguous attribute names and inconsistent terminology create mismatches between natural-language queries and schema organization (Lei et al., 2020). Multi-constraint queries allow for multiple plausible interpretations (Gomm et al., 2025), making it difficult to identify correct tables before SQL generation. Retrieval errors propagate to downstream SQL generation, and are compounded when varying data model designs admit multiple valid SQL structures for the same query (Fürst et al., 2024), where even reliably evaluating query correctness remains challenging (Kim et al., 2024). Moreover, while recent methods devise query decomposition for table retrieval, single-hop retrieval using full-query embeddings is a common default (Chen et al., 2025b; Zou et al., 2025), motivating a close examination of the robustness of single-hop retrieval for complex compositional queries.

Despite recent progress in table retrieval mechanisms, single-vector table embeddings cannot represent multi-constraint relevance signals (Weller et al., 2025) and degrade when query semantics diverges from schema conventions (Chen et al., 2025b; Lei et al., 2020). Industry workloads intensify these challenges: queries are verbose and compositional, databases span hundreds of tables with unaligned semantics, and relevant data is distributed across many join-connected tables. While existing studies surface patterns in retrieval performance across methods and datasets (Ji et al., 2025), deeper insights are lacking regarding such complex retrieval characteristics.

Focusing on complex retrieval settings, we study Decomposition-based Connectivity-aware Table Retrieval (DCTR), which combines 1) typed query decomposition for fine-grained query-schema alignment using multi-vector embeddings, and 2) schema-connectivity grouping for surfacing join-related tables beyond semantic similarity. We evaluate DCTR on industry-aligned benchmarks for open-domain analytical tabular question answering and show that decomposition is most useful for long, compositional queries and smaller embedding models. Finally, we formalize retrieval complexity along the query- and data dimensions, and analyze the influence of query signal-to-noise ratios, and schema heterogeneity and connectivity. Our findings surface the usefulness and limitations of fine-grained query decomposition and global connectivity-awareness in complex retrieval settings involving highly compositional queries and

*Correspondence to: madelon@cwi.nl

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 94–107
October 28, 2026 ©2026 Association for Computational Linguistics

large relational databases.

2 Related Work

2.1 Text-to-SQL and Schema Linking

Schema linking aligns natural-language queries with schema elements at varying granularities. RASL (Eben et al., 2025) introduces multi-stage retrieval for databases by decomposing schemas into semantic units. Nahid et al. (Nahid et al., 2025) propose a context-aware bidirectional schema retrieval approach that links queries to tables and columns using LLM-guided selection. Recent work questions explicit linking: Maamari et al. (Maamari et al., 2024) argue that high-capacity LLMs identify relevant schema despite distracting context, while RESDSQL (Li et al., 2023a) frames linking as ranking-enhanced encoding decoupled from SQL generation. These works generally assume a known data context and limited retrieval scope, providing a narrow view of the open-domain setting in complex data context.

2.2 Retrieval in Open-Domain Text-to-SQL

Open-domain QA faces ambiguous intent and heterogeneous corpora. MURRE (Zhang et al., 2024) performs iterative multi-hop retrieval with query rewriting which is complemented by data augmentation in Abacus-SQL (Xu et al., 2025), while LinkAlign (Wang et al., 2025) scales schema linking to large databases via multi-round query rewriting and response filtering. EnrichIndex (Chen et al., 2025b) and Pneuma (Balaka et al., 2025) show the value of enriching retrieval indices with LLM-generated metadata, while GenEdit (Maamari et al., 2025) integrates domain knowledge in SQL generation through specialized operators and user feedback. Complementary techniques include graph-based hierarchical retrieval (Zou et al., 2025) and weak supervision (Liang et al., 2025). To address the large scale of databases, R2D2 (Bodensohn and Binnig, 2024) retrieves table segments instead of full tables. JAR (Chen et al., 2025c) introduces join-aware reranking to balance table-query relevance with table-table compatibility, which was adapted for iterative retrieval by Boutaleb et al.. We build on these retrieval principles, but introduce *global* connectivity-aware retrieval to surface relevant tables beyond semantic similarity, and typed query decomposition for fine-grained schema alignment. Our use of FK-based schema graphs shares structural motivation with Relational Graph Transformers (Dwivedi et al., 2026), though that work targets node-level prediction rather than table retrieval. We analyze the effectiveness through the lens of complex queries and databases providing a novel view of performance in complex settings.

3 Problem Setting

Open-Domain Table Retrieval Let $T = \{T_1, \ldots, T_n\}$ denote the set of tables in a relational database, where each table T_i consists of a set of columns $C(T_i) = \{c_1, \ldots, c_m\}$, possibly connected through foreign-key relationships. In an open-domain setting, a natural-language query q expresses an information need from which the data scope needs to be inferred for retrieval of relevant tables from a large topic-diverse set of tables. The ground-truth set of relevant tables for q is denoted as $T^*(q)$. The table retrieval task aims to return a ranked list $R_k(q)$ of top-k candidate tables such that capped recall (CR) (Thakur et al., 2021) is maximized: $\mathrm{CR@k}(q) = \frac{|R_k(q) \cap T^*(q)|}{\min(k, |T^*(q)|)}$.

Retrieval Complexity The composition of queries and data that is typically faced in real-world retrieval tasks for end-to-end question answering stretches the complexity levels of what existing evaluations surface. We characterize retrieval complexity along two dimensions: *query complexity* and *data complexity*. We term query complexity as *the complexity induced by how the query is formulated in terms of semantic density and functional composition* (Gomm et al., 2025). We term data complexity as *the complexity resulting from how the data is structured within a database due to database normalization and schema size* (Chen et al., 2025c). These complexity dimensions guide our analysis of fine-grained retrieval for complex question answering tasks.

4 Fine-Grained Table Retrieval

We describe the Decomposition and Connectivity-aware Table Retrieval method (DCTR) that is tailored for complex retrieval tasks over two dimensions: 1) functional query decomposition for fine-grained schema alignment, and 2) global connectivity aware retrieval for recovering relevant tables beyond the semantic similarity scope.

Before queries are executed, an index of the data is built first. To support scalable retrieval over large schemas with thousands of tables and a proportional number of columns, we construct

two dense indices: one encoding table names and one encoding column names. Retrieval is performed by comparing embeddings of query elements against embeddings of these different schema elements.

4.1 DCTR Retrieval Pipeline

DCTR comprises a retrieval pipeline instrumented with fine-grained typed component extraction from queries, component-wise retrieval, join-graph construction, and FK-based expansion to account for global connectivity beyond semantic similarity (Fig. 1).

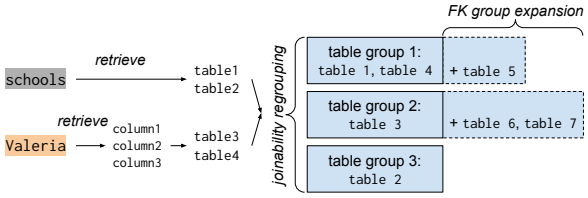


Figure 1: Overview of DCTR with component-wise retrieval, group formation, and FK-group expansion.

Typed Query Decomposition in Semantic Units In an open-domain setting, natural language queries reflect constraints over different entities, attributes, and operations (Gomm et al., 2025). To align these with concrete schema elements and operators, we first perform a *typed query decomposition* step that segments an incoming query q into atomic *semantic units*, forming the basis for both retrieval and downstream task execution (e.g. SQL generation). The decomposition is done using a language model with a fixed few-shot instruction prompt and decoding configuration to ensure consistent behavior across datasets (Appendix A). Since the prompt receives only the query text and no schema information, decomposition cost is independent of database size. DCTR classifies each *semantic unit* as one of three component types:

- Schema components, denoting candidate tables or columns names and are matched to database schemas during retrieval.
- Value components, corresponding to entities or literals that act as query filters.
- Aggregator components, which specify aggregation or comparison operations (e.g., `average`, `min`, `max`).

For example, the query "*What was the average sale of Luka Dončić jerseys in 2025?*" is decomposed into schema components `jersey` and `sale`, value components `Luka Dončić` and `2025`, and aggregator component `average`. In the retrieval phase, only schema and value components are used while the aggregator components benefits the downstream tasks, e.g. text-to-SQL.

First Pass Retrieval In the first stage of retrieval, each query component is processed independently for each database. For each component, we perform dense similarity search against both the table and column indices, retrieving the candidates per component. This initial retrieval produces unranked lists of tables and columns. The results from all components and all databases are then merged into a single candidate set. Columns retrieved from the column index are mapped back to their parent tables, ensuring that all relevant tables are included in the set that is forwarded to the graph-based grouping stage.

Schema Graph and Group Construction As discussed in (Chen et al., 2025c), joinable tables are relevant to consider but may not surface in a first-pass retrieval stage, which is conditioned on semantic similarity alone, requiring multi-hop reasoning over join paths. Given the set of candidate tables retrieved in the first pass, we represent each database schema as an undirected graph, where nodes correspond to tables and edges represent foreign-key (FK) relationships.

We restrict this graph to the first-pass candidate tables and the FK relationships between them. Each connected component defines a *table group*. By global FK-expansion, we then include all FK-connected tables with any table in the group to recover multi-hop join context.

Group Scoring Each table group is scored based on its coverage of the extracted query components. For each component, DCTR first computes the cosine similarity between its embedding and the embeddings of all tables in the group.

We introduce a parameter *vote_k* to cap the number of tables included in the scoring of a table group. The resulting *group coverage score* is obtained by summing the similarity scores of the top *vote_k* tables per component across all components: $\text{group\_cov\_score(T, C)} = \sum_{j=1}^{|C|} \text{top}_{\text{vote}_k}(\{\text{sim}(t_1, C_j), ..., \text{sim}(t_n, C_j)\})$. Higher group coverage score indicates that the tables in the group collectively cover more of the query's components, and therefore are more likely to contain the information needed to answer the query accurately. In the current formulation, all

components contribute equally to the group score. Because generic components (e.g., "id") produce uniformly high similarity across groups, they contribute little discriminative signal. Disambiguation is driven by specific components (e.g., "customer", "transaction date") that differentiate groups in the scoring.

Final Retrieval The final retrieval stage selects the most relevant tables by retaining the top $n_$groups scoring table groups per database. Within each selected group, the vote_k tables with the highest individual coverage scores are selected as the final candidate set. These two hyperparameters control the breadth and granularity of the retrieval, allowing the pipeline to balance recall and precision depending on query complexity and schema size.

5 Experimental Setup

5.1 Method and Baselines Implementation

Retrieval Implementation We implement the retrieval backend using Qdrant with two collections. Schema representations are indexed separately: table identifiers are stored in `table_embeddings`, while column identifiers are stored in `column_embeddings`. For query decomposition we use `gpt-4.1-mini`; switching to `gemini-2.5-pro` changes capped recall by only 0.007 on average. For text-to-SQL evaluation we use `gemini-2.5-pro`.

Hyperparameters DCTR introduces three parameters: *vote_k*, the number of in-group candidates selected per component during group scoring; *n_groups*, the number of table subgroups retained per database for downstream ranking; and *expand_groups*, whether to expand high-scoring subgroups via join paths in the schema graph.

Dense Retrieval Baseline We compare our method against the common dense retrieval baseline (Karpukhin et al., 2020). Given a natural-language query q, the baseline embeds the full query as a single vector and tables are represented by an embedding of their name, column names, and a sample of data values, in line with the strongest method from the TARGET benchmark (Ji et al., 2025). Retrieval is performed based on cosine similarity across table embeddings and the given query embedding.

Embedding Models All retrieval components use dense text embeddings for query units and schema elements. We evaluate the following three models covering different capacity–efficiency trade-offs. **Stella-large** (Zhang et al., 2025): a high-capacity model (400M parameters, 1024-dim) designed for fine-grained semantic representation. **BGE-small** (Xiao et al., 2024): a lightweight general-purpose model (33M parameters, 384-dim) serving as an efficiency-oriented baseline. **E5-small** (Wang et al., 2024): A lightweight model (33M parameters, 384-dim) trained with weakly supervised contrastive objectives for scalable retrieval.

5.2 Datasets and Metrics

Datasets We evaluate on three text-to-SQL benchmarks accessed via TARGET (Ji et al., 2025), selected to mirror the friction points of industry retrieval: large schemas, ambiguous column naming, compositional queries, and multi-level nested joins. Together they span enterprise settings, stretching both query and data complexity (Sec. 5.2). **BEAVER** (Chen et al., 2025a): an enterprise benchmark from real data warehouses, with large schemas (77.2 tables per database on average) and verbose queries (31.9 words on average) involving multi-table joins and aggregations. **FIBEN** (Sen et al., 2020): a finance-domain benchmark with 300 natural-language questions over a single dense schema of 152 tables, requiring reasoning over domain-specific jargon and complex nested queries. **BIRD** (Li et al., 2023b): a cross-domain benchmark spanning 11 databases with smaller but well-connected schemas (6.8 tables, 1.4 FKs per table). We report results on the test sets of BEAVER and FIBEN, and the validation set of BIRD. Full dataset characteristics are presented in Table 7 (Appendix D).

Retrieval Metrics We evaluate performance using Capped Recall RC@k for $k \in \{5, 10, 25\}$ across all embedding models and benchmarks (Sec. 3). First-stage candidate breadth is fixed to 30 and the remaining hyperparameters values are reported per experiment. Results are reported over multiple runs to account for stochasticity from LLM query decomposition. The dense baseline is deterministic and shows no variance.

Complexity Measures Following our notion of retrieval complexity as defined in Sec. 3, we investigate retrieval performance under varying degrees

of *query complexity* and *data complexity*, as typical industry workloads stretch these attributes. For each query, we capture attributes outlined below and analyze capped recall@25, aggregated across datasets and embedding models.

For *query complexity*, we measure 1) the length of the query in number of tokens, and 2) the number of extracted functional query components. For *data complexity*, we measure the total number of tables connected through FKs with the gold tables involved in a query reflecting the degree of schema normalization and connectivity. Distributional statistics of the datasets along these dimensions are presented in Appendix D.

6 Results

6.1 Overall Retrieval Performance

Tables 1-3 report capped Recall@k for $k \in \{5, 10, 25\}$ across datasets, embedding models, and retrieval settings. For small scope ($k = 5$), low settings for vote_k and n_groups perform best. As k increases, moderate vote_k and n_groups improve recall in several settings, with the largest gains on BEAVER and for smaller embedding models. Notably, smaller embedding models benefit most from DCTR, narrowing the gap with a high-capacity baseline model. Connectivity-aware retrieval is particularly beneficial for larger schemas, as seen on BEAVER. Good priors for DCTR settings are proportional to the retrieval scope: for small k (e.g. 5) increasing vote$_k$ or n_{groups} is ineffective, while for larger k, i.e. 10 or 25, setting vote$_k$ and n_{groups} to 2 and 5, respectively, yields consistent gain in recall (App. B).

Group expansion shows dataset-dependent behavior. It improves recall on BIRD (+3-5%), particularly at larger k, but degrades performance on FIBEN and BEAVER (up to 30%). This aligns with structural differences across benchmarks (App. D). That is, BIRD databases are small yet well-connected (6.8 tables, 1.4 FKs per table), making expansion effective for recovering non-semantically similar but relevant tables. In contrast, FIBEN and BEAVER contain substantially larger schemas (152 and 77.5 tables/DB), yielding more candidate tables that might get truncated by the k-capped evaluation. A dynamic value for k would be viable in practice to fully utilize global FK-expansion. Specifically, on BEAVER the same three hub tables dominate the top-3 in 7 of 10 failure cases (Appendix E), indicating that FK-expansion should be conditioned on schema density. We therefore treat expansion as an optional connectivity signal whose utility depends on schema topology, not as a default improvement over non-expanded retrieval.

Table 1: Capped Recall@5 (mean ± std) across datasets and models. Bold values indicate the best result per dataset and embedding model.

Setting	Model	BEAVER	FIBEN	BIRD
Baseline	stella	0.213	0.302	**0.901**
	bge-small	0.181	0.168	0.809
	e5-small	0.154	0.025	0.739
DCTR vote$_k=1$, $n_{\text{groups}}=1$	stella	**0.231**$_{\pm.004}$	**0.317**$_{\pm.001}$	0.834$_{\pm.001}$
	bge-small	**0.223**$_{\pm.005}$	**0.294**$_{\pm.003}$	**0.826**$_{\pm.002}$
	e5-small	**0.199**$_{\pm.009}$	**0.257**$_{\pm.004}$	**0.817**$_{\pm.007}$

Table 2: Capped Recall@10 (mean ± std) across datasets and models. Bold values indicate the best result per dataset and embedding model.

Setting	Model	BEAVER	FIBEN	BIRD
Baseline	stella	0.277	**0.413**	**0.961**
	bge-small	0.231	0.262	0.909
	e5-small	0.175	0.140	0.850
DCTR vote$_k=2$, $n_{\text{groups}}=2$	stella	**0.313**$_{\pm.007}$	0.396$_{\pm.006}$	0.899$_{\pm.004}$
	bge-small	**0.326**$_{\pm.002}$	**0.331**$_{\pm.006}$	0.893$_{\pm.003}$
	e5-small	**0.273**$_{\pm.007}$	**0.346**$_{\pm.004}$	0.882$_{\pm.006}$
DCTR vote$_k=2$, $n_{\text{groups}}=2$, expand_groups	stella	0.073$_{\pm.014}$	0.202$_{\pm.002}$	0.936$_{\pm.003}$
	bge-small	0.095$_{\pm.001}$	0.091$_{\pm.014}$	**0.942**$_{\pm.001}$
	e5-small	0.109$_{\pm.001}$	0.078$_{\pm.008}$	**0.904**$_{\pm.005}$

Table 3: Capped Recall@25 (mean ± std) across datasets and models. Bold values indicate the best result per dataset and embedding model.

Setting	Model	BEAVER	FIBEN	BIRD
Baseline	stella	0.413	**0.584**	0.986
	bge-small	0.350	**0.461**	0.976
	e5-small	0.281	0.283	0.949
DCTR vote$_k=5$, $n_{\text{groups}}=5$	stella	**0.435**$_{\pm.002}$	0.513$_{\pm.007}$	0.971$_{\pm.002}$
	bge-small	**0.429**$_{\pm.003}$	0.448$_{\pm.001}$	0.972$_{\pm.002}$
	e5-small	**0.408**$_{\pm.008}$	**0.516**$_{\pm.003}$	0.965$_{\pm.001}$
DCTR vote$_k=5$, $n_{\text{groups}}=5$, expand_groups	stella	0.228$_{\pm.001}$	0.259$_{\pm.001}$	**0.991**$_{\pm.000}$
	bge-small	0.232$_{\pm.002}$	0.127$_{\pm.014}$	**0.988**$_{\pm.002}$
	e5-small	0.228$_{\pm.001}$	0.011$_{\pm.003}$	**0.975**$_{\pm.001}$

6.2 Retrieval Complexity Analysis

Query complexity. Figure 2 shows the retrieval performance against *query length* in number of tokens. We find that DCTR consistently outperforms the dense baseline for longer queries, with the gap widening beyond 40 tokens. For smaller

embedding models, in particular, the baseline is more sensitive to noise in the query than DCTR illustrating the effectiveness of fine-grained query decomposition for complex queries.

Figure 3 reinforces this finding along the axis of query compositionality: while both methods perform comparably for queries with few components, the baseline's recall degrades steadily as compositionality increases, regardless of model capacity, whereas DCTR remains stable. High-capacity models are more robust to irrelevant noise (Fig. 2) but cannot resolve highly composite queries, confirming the need for component-level retrieval.

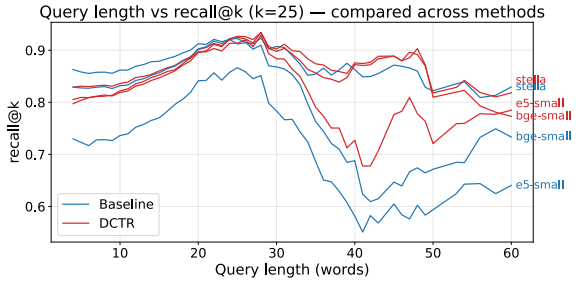


Figure 2: Query length vs recall@25, aggregated across datasets. Comparison through three embedding models, shows that DCTR outperforms the baseline, especially for longer queries ($\geq$40 tokens).

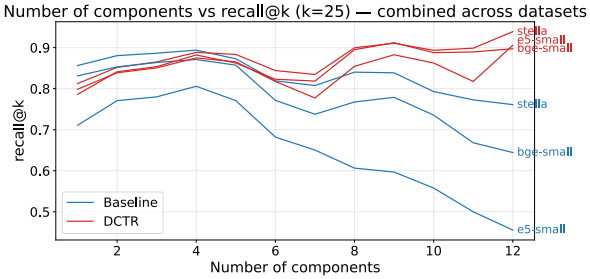


Figure 3: Number of components vs Recall@25 across datasets. Comparison through three embedding models shows that DCTR is robust for compositional queries while a single-vector baseline does not resolve compositional queries regardless of model capacity.

Data complexity. We assess data complexity via the connectivity of gold tables, measured by their number of join relationships within the database. Figure 4 illustrates that DCTR achieves stronger performance on queries involving densely connected tables, indicating that connectivity-aware grouping improves retrieval of joinable table sets.

6.3 Downstream Text-to-SQL Performance

We assess downstream utility by comparing DCTR (*stella* model, $k \in \{5, 10, 25\}$) against a long-context baseline that passes the full schema

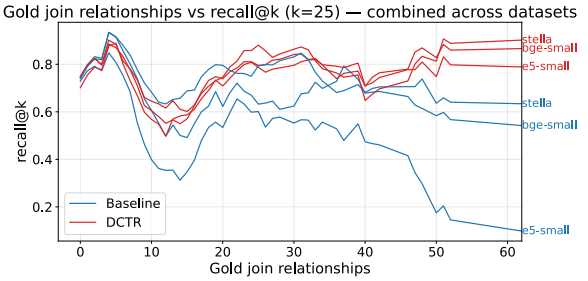


Figure 4: Number of tables connected to gold tables vs recall@25, combined across datasets, for different embedding models. DCTR improves more on queries where gold tables are densely connected.

to Gemini (Chung et al., 2025), using execution accuracy (Table 4). DCTR improves execution accuracy by 3% on BEAVER and 5% on FIBEN while matching the baseline on BIRD. The performance over k varies per dataset and correlates with retrieval difficulty: BIRD achieves peak accuracy at small k where recall is high, FIBEN requires moderate k, and BEAVER, where retrieval is most challenging, benefits from larger k.

Table 4: Downstream SQL execution accuracy of DCTR against the long-context baseline.

Setting		BEAVER	FIBEN	BIRD
Baseline		0.2730	0.1370	0.6630
DCTR	k=5	0.2150	0.0970	**0.6656**
	k=10	0.2400	**0.1870**	0.6630
	k=25	**0.2980**	0.1270	0.6623

7 Conclusion

The emergence of natural language interfaces to large-scale databases amplifies the importance of retrieval mechanisms that are robust to complex query- and data characteristics. In this context, we introduce Decomposition-based Connectivity-aware Table Retrieval (DCTR), which yields recall improvements over common single-vector dense retrieval baselines in several complex retrieval settings. Analysis of retrieval complexity along the axes of query- and data complexity surfaces that typed query decomposition is key in addressing highly compositional queries that require fine-grained multi-table schema alignment, while global connectivity-aware retrieval uncovers relevant tables beyond semantically similar tables in densely connected relational databases. The modular group scoring in DCTR naturally accommodates extensions such as component weighting, schema-density aware expansion, user-specific priors, and multi-turn refinement of decomposed components, which we leave for future work.

Limitations

Despite its effectiveness, DCTR has several limitations. Its performance depends on accurate query decomposition as errors may propagate to retrieval, and gains diminish for simple, single-table queries where added reasoning incurs unnecessary overhead.

Connectivity-aware retrieval further assumes an informative relational structure. In weakly connected, de-normalized, or poorly specified schemas, global connectivity signals may offer limited advantage over strong dense baselines.

The method also introduces additional cost due to query decomposition and multi-stage retrieval, which may affect latency-sensitive deployments. Finally, evaluation on industry-representative benchmarks may not fully reflect the diversity of real-world schema designs and query distributions.

References

Muhammad Imam Luthfi Balaka, David Alexander, Qiming Wang, Yue Gong, Adila Krisnadhi, and Raul Castro Fernandez. 2025. Pneuma: Leveraging llms for tabular data representation and retrieval in an end-to-end system. *arXiv preprint arXiv:2504.09207*.

Jan-Micha Bodensohn and Carsten Binnig. 2024. Rethinking table retrieval from data lakes. In *Proceedings of the Seventh International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM 2024, Santiago, Chile, 14 June 2024*, pages 2:1–2:5. ACM.

Allaa Boutaleb, Bernd Amann, Rafael Angarita, and Hubert Naacke. Exploring multi-table retrieval through iterative search. In *EurIPS 2025 Workshop: AI for Tabular Data*.

Peter Baile Chen, Fabian Wenz, Yi Zhang, Devin Yang, Justin Choi, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. 2025a. Beaver: An enterprise benchmark for text-to-sql. *Preprint*, arXiv:2409.02038.

Peter Baile Chen, Tomer Wolfson, Michael Cafarella, and Dan Roth. 2025b. Enrichindex: Using llms to enrich retrieval indices offline. *Preprint*, arXiv:2504.03598.

Peter Baile Chen, Yi Zhang, and Dan Roth. 2025c. Is table retrieval a solved problem? exploring join-aware multi-table retrieval. *Preprint*, arXiv:2404.09889.

Yeounoh Chung, Gaurav T. Kakkar, Yu Gan, Brenton Milne, and Fatma Özcan. 2025. Is long context all you need? leveraging llm's extended context for nl2sql. *Proceedings of the VLDB Endowment*, 18(8):2735–2747.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. 2026. Relational graph transformer. *Preprint*, arXiv:2505.10960.

Jeffrey Eben, Aitzaz Ahmad, and Stephen Lau. 2025. Rasl: Retrieval augmented schema linking for massive database text-to-sql. *Preprint*, arXiv:2507.23104.

Jonathan Fürst, Catherine Kosten, Farhad Nooralahzadeh, Yi Zhang, and Kurt Stockinger. 2024. Evaluating the data model robustness of text-to-sql systems based on real user queries. *Preprint*, arXiv:2402.08349.

Daniel Gomm, Cornelius Wolff, and Madelon Hulsebos. 2025. Are we asking the right questions? on ambiguity in natural language queries for tabular data analysis. In *AI for Tabular Data workshop at EurIPS 2025*.

Xingyu Ji, Parker Glenn, Aditya G. Parameswaran, and Madelon Hulsebos. 2025. Target: Benchmarking table retrieval for generative tasks. *Preprint*, arXiv:2505.11545.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Heegyu Kim, Taeyang Jeon, Seunghwan Choi, Seungtaek Choi, and Hyunsouk Cho. 2024. Flex: Expert-level false-less execution metric for reliable text-to-sql benchmark. *Preprint*, arXiv:2409.19014.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Online. Association for Computational Linguistics.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023a. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. *Preprint*, arXiv:2302.05965.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023b. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Preprint*, arXiv:2305.03111.

Hsing-Ping Liang, Che-Wei Chang, and Yao-Chung Fan. 2025. Improving table retrieval with question generation from partial tables. *Preprint*, arXiv:2508.06168.

Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. The death of schema linking? text-to-sql in the age of well-reasoned language models. *Preprint*, arXiv:2408.07702.

Karime Maamari, Connor Landy, and Amine Mhedhbi. 2025. Genedit: Compounding operators and continuous improvement to tackle text-to-sql in the enterprise. In *Proceedings of the 11th Conference on Innovative Data Systems Research (CIDR)*, Amsterdam, The Netherlands.

Md Mahadi Hasan Nahid, Davood Rafiei, Weiwei Zhang, and Yong Zhang. 2025. Rethinking schema linking: A context-aware bidirectional retrieval approach for text-to-sql. *Preprint*, arXiv:2510.14296.

Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. Athena++: natural language querying for complex nested sql queries. *Proc. VLDB Endow.*, 13(12):2747–2759.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Yihan Wang, Peiyu Liu, and Xin Yang. 2025. Linkalign: Scalable schema linking for real-world large-scale multi-database text-to-sql. *arXiv preprint arXiv:2503.18596*.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2025. On the theoretical limitations of embedding-based retrieval. *Preprint*, arXiv:2508.21038.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. *Preprint*, arXiv:2309.07597.

Keyan Xu, Dingzirui Wang, Xuanliang Zhang, Qingfu Zhu, and Wanxiang Che. 2025. Abacus-sql: A text-to-sql system empowering cross-domain and open-domain database retrieval. *arXiv preprint arXiv:2504.09824*.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. Jasper and stella: distillation of sota embedding models. *Preprint*, arXiv:2412.19048.

Xuanliang Zhang, Dingzirui Wang, Longxu Dou, Qingfu Zhu, and Wanxiang Che. 2024. Murre: Multi-hop table retrieval with removal for open-domain text-to-sql. *arXiv preprint arXiv:2402.10666*.

Jiaru Zou, Dongqi Fu, Sirui Chen, Xinrui He, Zihao Li, Yada Zhu, Jiawei Han, and Jingrui He. 2025. Gtr: Graph-table-rag for cross-table question answering. *arXiv preprint arXiv:2504.01346*.

Appendix

A Query Decomposition Prompt

We use a fixed instruction prompt to extract schema-related query components. The prompt is shown below:

```
---Goal---
Given a natural language query and a
↪  list of component types, identify
↪  all schema
element components from the text.

---Output---
For each identified component, extract
↪  the following information:
- component_name: Name of the component,
↪  use exact phrase from the input
↪  query.
  Keep each as short and concise as
  ↪  possible.
- component_description: Short
↪  description of the component's
↪  attributes and any
  potential context that might be
  ↪  helpful for understanding the role
  ↪  of the component
  within the query.

---Schema Element Component
↪  Description---
The specific data schema element (e.g.
↪  table or column names) directly
↪  referenced
in the query. Think about which parts of
↪  the query are most likely to have
↪  counterparts
in the available data, and explain why.
↪  Be as concise as possible, and shave
↪  any
constraints or filtering phrases
↪  surrounding the schema element
↪  component.

Note: Exclude phrases that are more
↪  likely to refer to specific cell
↪  values rather than
schema elements. Only focus on elements
↪  most likely to have matching schema
↪  entries.

---Examples---
{few_shot_examples}

---Real Data---
Input Query:
{query}

Output:
```

B Ablation Study

We study the effect of key hyperparameters (*n_groups*, *vote_k*, and *expand_groups*) on retrieval performance. Since datasets differ in database count, schema size, and inter-table connectivity, hyperparameter selection is essential for balancing coverage and precision. We focus on $k = 25$, where the effects of grouping and voting are most pronounced and easier to observe.

Results. Table 5 summarizes the ablation results. Across datasets, increasing *n_groups* and *vote_k* consistently improves recall up to a saturation point, beyond which gains diminish or disappear. For $k = 25$, optimal performance is typically achieved with *n_groups* and *vote_k* in the range of 5–6, while larger values yield no further improvement.

On BEAVER and FIBEN, enabling group expansion substantially degrades performance across all configurations. In contrast, BIRD benefits from group expansion in several settings, often reaching near-perfect recall. These trends are consistent across embedding models, indicating that hyperparameter behavior is largely independent of the embedding choice and instead driven by dataset characteristics.

Ablation results for k=5,10 For $k \in 5, 10$, we evaluate larger hyperparameter configurations $(n_{\text{groups}}, \text{vote}_k)$, with settings 1–3 considered for $k = 5$ and 1–4 for $k = 10$, each evaluated both with and without *expand_groups*.

Results in Table 6 show the effect of hyperparameter choices for $k = 5$ and $k = 10$, using only `stella-large`, as no significant differences were observed across models.

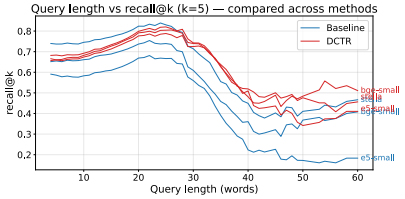
The optimal $(n_{\text{groups}}, , vote_k)$ configuration varies by dataset. For BEAVER, the best performance is achieved with $(1, 1)$ at $k = 5$ and $(2, 2)$ at $k = 10$. In FIBEN, $(1, 1)$ yields the highest recall for both $k = 5$ and $k = 10$. For BIRD, $(3, 3)$ provides the best results at $k = 5$. Note that BIRD is evaluated on a 200-query subsample, so these optimal values differ when considering the full dataset.

C Results for Detailed Per-Query Analysis for k=5,10

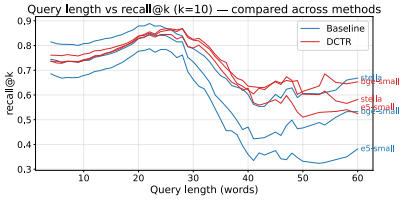
Figures [5, 6, 7, 8] report per-query analyses for both query and data complexity at $k = 5$ and $k = 10$.

In addition to the results discussed in the main text, Figure 7 presents recall as a function of the number of gold tables. As expected, retrieval difficulty increases with the number of required tables, leading to consistent recall degradation. Our method follows a similar trend to the dense baseline, indicating that large multi-table requirements remain challenging.

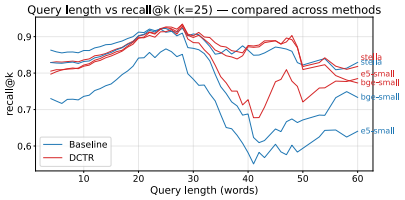
The remaining analyses replicate the trends observed in the main section. On the query complexity side, performance improves with increasing query length and number of extracted components. On the data complexity side, the method is more robust for queries involving densely connected tables, yet performance deteriorates when a large number of gold tables must be jointly retrieved.



(a) CappedRecall@5



(b) CappedRecall@10



(c) CappedRecall@25

Figure 5: Query length vs recall@k, combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. Across all values of k, the proposed method consistently outperforms the baseline, especially for longer queries ($\geq$40 tokens).

D Datasets and Query distribution

Figure 9 shows the distribution of query and data complexity attributes across the evaluated queries. Most queries are shorter than 40 tokens, contain

Dataset	(n_{groups}, $vote_k$)	Embedding Model					
		stella	bge-small	e5-small	stella	bge-small	e5-small
		expand_groups=False			expand_groups=True		
BEAVER	(3,3)	0.3901	0.3869	0.3782	0.2231	0.2338	0.2155
	(4,4)	0.4106	0.4073	0.3933	0.2306	0.2392	0.2392
	(5,5)	0.4332	0.4321	0.4106	0.2284	0.2306	0.2284
	(6,6)	0.4504	0.4332	0.4138	0.2285	0.2349	0.2371
	(7,7)	0.4569	0.4353	0.3966	0.2381	0.2403	0.2371
FIBEN	(3,3)	0.4526	0.3890	0.4091	0.2324	0.1149	0.1227
	(4,4)	0.4769	0.4051	0.4508	0.2524	0.1271	0.1044
	(5,5)	0.5152	0.4491	0.5178	0.2594	0.1366	0.1131
	(6,6)	0.5466	0.4700	0.5535	0.2672	0.1271	0.1262
	(7,7)	0.5648	0.4795	0.5614	0.2715	0.1393	0.1184
BIRD (200 queries)	(3,3)	0.9214	0.9337	0.9509	1.0000	1.0000	0.9951
	(4,4)	0.9582	0.9754	0.9730	1.0000	1.0000	0.9975
	(5,5)	0.9754	0.9705	0.9902	1.0000	1.0000	0.9951
	(6,6)	0.9754	0.9803	0.9926	1.0000	1.0000	0.9877
	(7,7)	0.9754	0.9779	0.9926	1.0000	0.9926	0.9975

Table 5: Hyperparameter ablation results. Capped Recall@25 for different (n_{groups}, $vote_k$) settings.

Dataset	(n_{groups}, vote_k)	Recall@5		Recall@10	
		False	True	False	True
BEAVER	(1,1)	0.2333	0.0317	0.2575	0.0830
	(2,2)	0.1993	0.0294	0.3308	0.0873
	(3,3)	0.1880	0.0294	0.3114	0.0862
	(4,4)			0.2931	0.0916
FIBEN	(1,1)	0.4626	0.1737	0.4743	0.1766
	(2,2)	0.3763	0.1694	0.4034	0.2066
	(3,3)	0.3477	0.1604	0.4368	0.1988
	(4,4)			0.4359	0.1918
BIRD (200 queries)	(1,1)	0.8354	0.7641	0.8452	0.9165
	(2,2)	0.8697	0.8010	0.9042	0.9754
	(3,3)	0.8747	0.7740	0.9140	0.9754
	(4,4)			0.9435	0.9779

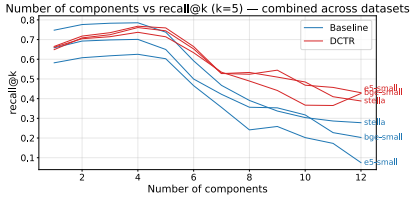
Table 6: Hyperparameter ablation results for `stella-large` embedding model. Capped Recall@5 and @10 are reported for different (n_{groups}, $vote_k$) settings, with *expand_groups* enabled (True) or disabled (False).

Metric	BEAVER	BIRD	FIBEN
Queries	209	1534	300
Unique DBs	6	11	1
Avg query (w)	31.9	14.6	12.3
Avg SQL (tok)	152.3	25.4	61.9
Tables	463	75	152
Columns	4238	798	373
Avg Tab./DB	77.2	6.8	152.0
Avg Col./Tab.	9.2	10.6	2.5
Avg FK/Tab.	1.1	1.4	1.0

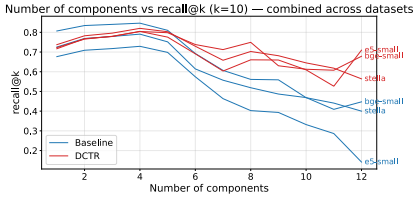
Table 7: Benchmark datasets characteristics.

1-6 extracted components, require 1-4 gold tables, and involve 1-10 join relationships among those tables. These distributions contextualize the per-query analyses in the main paper and clarify the typical structural characteristics of the evaluated workloads.

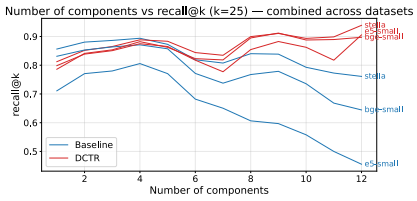
Additionally, Table 7 summarizes benchmark characteristics. BEAVER and FIBEN contain substantially more tables and higher average tables per database than BIRD, making retrieval considerably more challenging in these enterprise-scale settings. In contrast, BIRD exhibits the highest number of foreign keys per database, a structural property that better aligns with our connectivity-aware design and increases the likelihood of benefiting from group expansion.



(a) CappedRecall@5

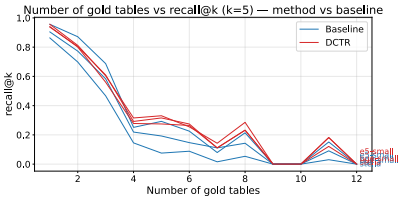


(b) CappedRecall@10

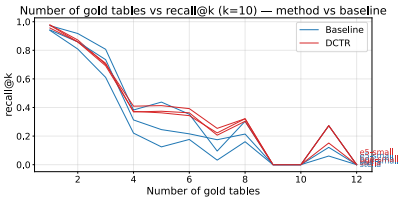


(c) CappedRecall@25

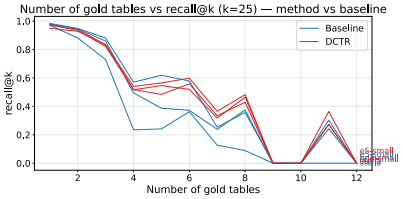
Figure 6: Number of components vs recall@k, combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. Our method consistently performs better than the baseline for queries with more components. The proposed method consistently outperforms the baseline as the number of query components increases. At k=25 (Figure 6c), our method maintains stable performance for highly compositional queries, whereas the baseline exhibits a performance drop.



(a) CappedRecall@5

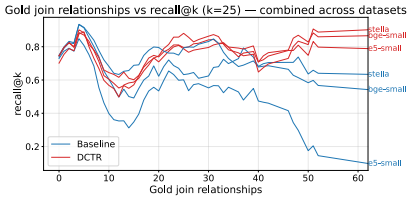


(b) CappedRecall@10

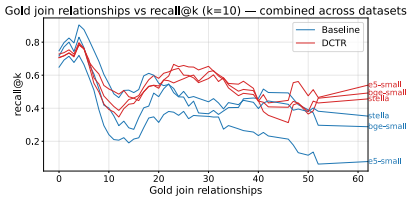


(c) CappedRecall@25

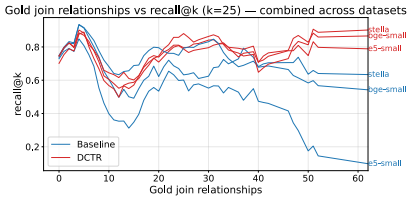
Figure 7: Number of tables gold tables vs recall@k, combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. Both methods exhibit a degradation in capped recall as there the number of gold tables increases. The proposed method behaves similarly to the baseline, indicating limited advantage when there is a lot of relevant tables to retrieve.



(a) CappedRecall@5

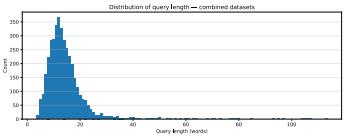


(b) CappedRecall@10

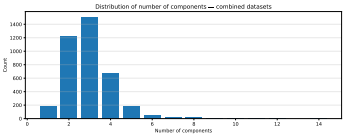


(c) CappedRecall@25

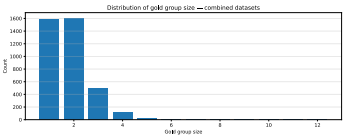
Figure 8: Number of gold tables join relationships vs recall@k, combined across datasets. Comparison of three embedding models for the proposed method versus the baseline. DCTR improves performance on queries for which gold tables are densely connected.



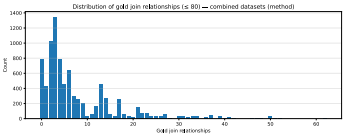
(a) Distribution of query length



(b) Distribution of number of extracted schema/value components



(c) Distribution of average group size of gold tables



(d) Distribution of number of join relationships in gold tables

Figure 9: Distribution of analyzed properties. Results combined for all datasets.

E FK-Expansion Failure Cases

Table 8 presents examples where FK-expansion degrades retrieval on BEAVER. Hub tables such as `SPACE_UNIT` and `LIBRARY_SUBJECT_OFFERED` are connected to many tables via foreign keys. When expansion is enabled, these hub nodes pull in large numbers of unrelated tables, displacing gold tables under the fixed k budget.

#	Query (abridged)	Gold tables	Top-3 after expansion
1	For each course, provide room number, building name, area, organization name, room usage, term code, course level, total subjects and units.	COURSE_CATALOG_ SUBJECT_OFFERED, FAC_ROOMS, FAC_BUILDING, FAC_MAJOR_USE, FAC_BUILDING_ADDRESS, SIS_COURSE_DESCRIPTION (6 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
2	Show unique activity titles, locations, term start date, and supervisor name for all independent activities.	IAP_SUBJECT_DETAIL, IAP_SUBJECT_PERSON, ACADEMIC_TERMS_ALL, IAP_SUBJECT_SESSION (4 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
3	For each building key, what is the building name and the number of courses offered by the Center for International Studies?	CIS_COURSE_CATALOG, COURSE_CATALOG_ SUBJECT_OFFERED, FCLT_ROOMS, FCLT_BUILDING (4 tables)	FAC_BUILDING, FAC_ROOMS, FCLT_ORGANIZATION
4	For each course, list its title, building name, duration, and a running average of duration partitioned by academic year.	ACADEMIC_TERMS, COURSE_CATALOG_ SUBJECT_OFFERED, FCLT_ROOMS, FCLT_BUILDING (4 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
5	For each course, list its name, building name, start date, previous and next course names, access level, and assignable area.	COURSE_CATALOG_ SUBJECT_OFFERED, ACADEMIC_TERMS, FCLT_ROOMS, FCLT_BUILDING (4 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
6	Provide information about all physical hosts.	RESOURCE_PROVIDERS, ALLOCATIONS, INVENTORIES, COMPUTE_NODES (4 tables)	PORTS, SUBNETROUTES, SUBNETPOOLPREFIXES
7	For each sponsor of independent activity periods, list sponsor name, number of sessions, total enrollment, min/max fee.	IAP_SUBJECT_SESSION, IAP_SUBJECT_DETAIL, IAP_SUBJECT_SPONSOR (3 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
8	What is the term code, term description, total IAP sessions, total fee collected, and min/max enrollment per term?	IAP_SUBJECT_SESSION, IAP_SUBJECT_DETAIL, ACADEMIC_TERMS_ALL (3 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
9	For each term, list term code, description, whether current, and total number of CIS course types.	ACADEMIC_TERMS_ALL, ACADEMIC_TERM_PARAMETER, CIS_COURSE_CATALOG (3 tables)	SPACE_UNIT, LIBRARY_SUBJECT_ OFFERED, SPACE_FLOOR
10	Group biology and chemistry courses by current status and department; list max/avg duration and units with subtotals.	SUBJECT_SUMMARY, ACADEMIC_TERMS_ALL, SUBJECT_OFFERED (3 tables)	SIS_COURSE_ DESCRIPTION, ACADEMIC_TERMS, FCLT_ORGANIZATION

Table 8: FK-expansion failure cases on BEAVER. In 7 of 10 cases, the same three hub tables (SPACE_UNIT, LIBRARY_SUBJECT_OFFERED, SPACE_FLOOR) dominate the top-3, displacing all gold tables under the fixed k.

EventSocraticKG: Event-Centric Knowledge Graph Construction for Narrative Retrieval-Augmented Generation

William Rosengren
KTH Royal Institute of Technology
wrose@kth.se

Abstract

Graph-based retrieval over book-length narrative text faces a choice at construction time. Merging every mention of a character into one node erases the temporal context that separates an early portrayal from a later one; keeping mentions apart scatters a character across disconnected per-chunk nodes and leaves the consolidation to query time. We present **EventSocraticKG** (eSoKG), which (i) makes events first-class nodes in a bipartite event–entity graph, (ii) extracts them from 5W1H question–answer pairs written for each chunk instead of from the chunk text, and (iii) merges mentions across chunks by their full description, with embeddings proposing candidates and an LLM confirming them. At query time eSoKG serves graph structure only, with no source chunks and no LLM call. On 332 long-form questions over five books (four pre-2000 novels and one post-cutoff novel), scored by an LLM judge and re-scored by two independent judges, eSoKG lies within half a point of an unmerged entity–event comparator and of a full-budget entity-centric method that spends roughly twice the per-query tokens, under all three judges. The post-cutoff book separates retrieval from recall: an LLM without retrieval collapses there while every retrieval method holds, but eSoKG trails both graph comparators, so part of its pooled standing reflects books the models already know. Its category profile, strongest on the consistency categories and weakest on Narrative and Thematic, is largely unchanged on that book. The price is a one-time build two to three times the comparators'. What it buys is a chunk-free query path in the low per-query cost regime and a merged graph on which character-chain, shared-event, and causal-chain queries are short graph walks that need no language model.

1 Introduction

A novel is a sequence of events whose meaning depends on everything that came before. Answering questions about the first Harry Potter book, such as "which characters appear in scenes with both Harry and Snape during his first year?" or "what earlier events lead the trio to the trapdoor?", requires tracing chains of events across the whole book rather than looking up a single passage. A knowledge graph (KG) that records events, their participants, and the causal–temporal links between them does this structural work once at construction time and exposes it cheaply at every query. Graph-based retrieval-augmented generation (RAG) builds on this, and so do agents that navigate the graph directly (Ghandi et al., 2026; Ni et al., 2025). In both cases the answer can only be as good as the graph, which puts the weight on how the graph is built.

Published methods for building a KG over book-length text share the same recipe: split the source into chunks, extract a subgraph per chunk, and combine the subgraphs. The combination step is where things get hard. The same character is named differently across chunks, and reconciling those mentions raises a dilemma. *Merging* every mention into one node, as entity-centric methods do by name, can average away a character whose state drifts across the story. *Not merging*, the choice made by the closest prior work, E^{2}RAG (Zhang et al., 2026), to preserve that temporal context, leaves a character as many disconnected nodes, so a query about "every event this character took part in" must first gather all the fragments back together, doing at query time the consolidation that construction skipped.

We take the opposite position from E^{2}RAG and ask whether the dilemma is false: whether a character's *per-event participations* can carry the temporal context that keeping mentions separate protects, so that a merged graph answers narrative questions as well as an unmerged one, but more cheaply per query. Our method, **EventSocraticKG** (eSoKG), makes three design choices that no prior system combines:

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 108–127
October 28, 2026 ©2026 Association for Computational Linguistics

- an **event-centric bipartite schema** with no entity–entity edges;
- **5W1H question-mediated extraction**, adapted from SocraticKG (Choi et al., 2026), which turns each chunk into question–answer pairs before any structure is extracted, to surface participant roles and causal links that direct extraction misses;
- **cross-chunk merging by full description**, using embeddings to propose merge candidates and an LLM to confirm them, rather than string-matching names.

At query time eSoKG retrieves a *chunk-free* context: distilled entity, event, and relation descriptions only, with no raw source text and no LLM call. We evaluate the resulting graph inside a RAG pipeline and ask two questions:

RQ1 (quality) Does a chunk-free structural context answer narrative questions as accurately as graph-based methods whose context includes the raw source chunks?

RQ2 (cost) How does the approach compare in cost, both the per-query cost of serving a question and the one-time cost of building the graph?

Separately from this QA evaluation, we argue for a benefit it does not measure: because events are first-class nodes and merging consolidates each entity into one, the graph answers structural queries (a character's event chain, the events two characters share, the causal chain into an event) as short graph walks needing no language model, which the entity-centric and unmerged comparators cannot serve directly.

Against four comparators on the ChronoQA benchmark plus one post-cutoff novel, eSoKG lies within half a point of the best method on the pooled question set under all three judges, at a low per-query cost, and it is the only method that spends no LLM tokens at retrieval time. On the single contamination-free book it trails both graph comparators, and its one-time build cost is two to three times the comparators'.

2 Background and Related Work

From triples to events. The triple (s, p, o) fits binary facts but forces a narrative event, with several participants in distinct roles, anchored in time and linked to causes and effects, into either compound nodes specific to one passage (coverage, at the cost of reuse) or a set of pairwise edges that dissolves the event's unity (coherence, at the cost of the event) (Choi et al., 2026). Hyperedges resolve this by connecting any number of entities, and their bipartite reformulation, in which each hyperedge becomes a node joined by ordinary edges to its participants, keeps n-ary relations while staying compatible with standard graph tooling (Luo et al., 2025; Wen et al., 2016). Event-centric KGs (Bai et al., 2025) take this form: events are nodes, and edges record participation and event–event causality.

LLM-driven construction. Prompting an instruction-tuned LLM with a passage and a schema yields entities, relations, and events directly (Wei et al., 2024; Edge et al., 2025; Guo et al., 2025; Bai et al., 2025). The open question is whether the graph faithfully captures the source. No gold reference graph exists for a book-length narrative, so the question has to be answered *extrinsically*, by placing the graph in a RAG pipeline and scoring downstream answers.

QA-mediated extraction and deduplication. A separate strand writes content out as natural-language question–answer pairs before extraction; SocraticKG (Choi et al., 2026) argues the *5W1H* structure of that step surfaces motivations and roles that single-pass extraction skips. Cross-chunk deduplication (Christen, 2012) must answer two questions, *when* two mentions co-refer and *whether* they should be merged, because merging a referent with distinguishable states erases what separates them.

Positioning. Table 1 places eSoKG against its three closest comparators. It inherits E^{2}RAG's bipartite event-centric schema and target use case but takes the opposite stance on merging; it adapts SocraticKG's 5W1H step but places it inside an event schema rather than a flat triple graph; and unlike LightRAG it drops entity–entity edges and name-based merging entirely. EventRAG (Yang et al., 2025b) also consolidates events by similarity but targets multi-hop QA over knowledge-heavy collections with dated edges, not the unlabelled state-drift of a single narrative.

Axis	LightRAG	E²RAG	eSoKG (ours)
Focus	general	narrative	narrative
Schema	entity–rel.	bipartite ev.	bipartite ev.
Extract	1–2 calls/chunk	2 calls/chunk	5W1H QA → ent. → ev.
Dedup	by name	none	by description (emb.+LLM)
Context	graph+chunks	graph+chunks	**chunk-free**

Table 1: eSoKG against its closest comparators. SocraticKG (not shown) contributes the 5W1H QA step but outputs a flat triple graph evaluated intrinsically.

3 EventSocraticKG

3.1 Graph

For a book b, the eSoKG graph is

$$G^{(b)}_{\text{eSoKG}} = \big(V_{\text{ent}} \cup V_{\text{evt}},\ E_{\text{evt}} \cup B\big),$$

with entity nodes V_{ent}, event nodes V_{evt}, typed directed event–event relations $E_{\text{evt}} \subseteq V_{\text{evt}} \times V_{\text{evt}}$ (causal and temporal), and a bipartite participation relation $B \subseteq V_{\text{ent}} \times V_{\text{evt}}$. There are *no* entity–entity edges: entities relate only through shared events, which makes the event chains in E_{evt} the load-bearing structure for narrative reasoning. Nodes and edges carry types from closed vocabularies (10 entity types; 12 event types such as `Movement`, `Decision`, `Conflict`; 9 relation types such as `causes`, `enables`, `follows`), passed to the LLM as guidance; node types have an `Other` fallback.

3.2 Indexing

Indexing runs in two phases: local per-chunk extraction, then global reconciliation. The source is split with a token-based sliding window (2000-token chunks, 400 overlap), and each chunk is keyed by its content hash and processed independently.

Phase 1: QA-mediated per-chunk extraction. Each chunk goes through three sequential LLM calls. *(1) QA generation*: the chunk is turned into self-contained, pronoun-resolved question–answer pairs; the prompt uses the 5W1H questions to drive coverage of the chunk, not as a quota. *(2) Entity extraction*: a second call reads *only the QA pairs* (never the raw chunk) and returns entity records (name, type, aliases, a 1–3 sentence description) tagged with ephemeral identifiers. *(3) Event extraction*: a third call receives the QA pairs and a slim view of the entities and returns events (each with participants referenced by the ephemeral tags) together with intra-chunk event–event relations. Splitting entity and event extraction costs one extra round-trip but lets the event extractor reference a stable entity list, producing cleaner participant lists. Bipartite edges are then added from each event's explicit participant list, with a case-insensitive substring fallback scanning the event description for known entity names (the same rule E²RAG uses as its only mechanism). Mentions receive chunk-scoped stable identifiers, so two mentions of one name in two chunks stay distinct until Phase 2.

Phase 2: description-based deduplication. After Phase 1 a character is typically spread over many nodes. Phase 2 merges duplicates using two signals. Embeddings are cheap over all pairs but capture only surface form; an LLM can judge true co-reference but is too costly per pair. eSoKG therefore lets *embeddings propose and the LLM confirm*: a merge needs both a candidate pair from the embeddings and a yes from the LLM. Each node is embedded from its name, type, description, and aliases (entities) or participants (events). At a similarity threshold τ, connected components of the thresholded pairwise-similarity graph form candidate clusters; multi-member clusters go to an LLM told the candidates came from vector similarity, that none is privileged, and that returning zero merges is valid. The threshold is swept high-to-low so safe merges commit and enrich descriptions before looser candidates are examined, and the whole sweep is run *twice*, re-embedding merged clusters from their consolidated descriptions to catch duplicates whose original vectors were too narrow. Event thresholds are kept higher than entity thresholds, and the event prompt is told that recurring events with overlapping participants are still distinct, because a false event merge permanently destroys a distinction whereas a miss only leaves a duplicate. Commits are also *asymmetric*: an absorbed entity's chunk-scoped vector is kept, with a `merged_with` pointer, so it still works as a retrieval entry point, while an absorbed event's vector is deleted as noise. All stored identifiers (event participants, E_{evt}, and B) are finally rewritten to canonical ids.

3.3 Chunk-free retrieval

Retrieval makes *no* LLM call. The question is embedded once and reused for two parallel seeding paths. *Path A (event-first)* queries the event store

Algorithm 1 eSoKG indexing

Require: Book b
Phase 1: per-chunk extraction
for all $c \in \textsc{Chunk}(b)$ **do**
  $qa \leftarrow \textsc{QALLM}(c)$ ▷ 5W1H QA pairs
  $V^c_{\text{ent}} \leftarrow \textsc{EntityLLM}(qa)$
  $V^c_{\text{evt}}, E^c_{\text{evt}} \leftarrow \textsc{EventLLM}(qa, V^c_{\text{ent}})$
  $B \leftarrow B \cup \textsc{Link}(V^c_{\text{ent}}, V^c_{\text{evt}})$
  accumulate $V_{\text{ent}}, V_{\text{evt}}, E_{\text{evt}}$
end for
Phase 2: global deduplication
$V^\star_{\text{ent}} \leftarrow \textsc{Dedup}(V_{\text{ent}})$; $V^\star_{\text{evt}} \leftarrow \textsc{Dedup}(V_{\text{evt}})$
rewrite participants, E_{evt}, B to canonical ids
return $(V^\star_{\text{ent}} \cup V^\star_{\text{evt}},\ E_{\text{evt}} \cup B)$

Algorithm 2 $\textsc{Dedup}(V)$: embeddings propose, LLM confirms

Require: nodes V; thresholds $\tau_1 > \ldots > \tau_k$
$M \leftarrow$ row-normalised embeddings of V; $S \leftarrow MM^\top$
for pass $\in \{1, 2\}$ **do**
  for $\tau \in \{\tau_1, \ldots, \tau_k\}$ **do**
    $G_\tau \leftarrow \{(i, j) : S_{ij} \geq \tau\}$ over current canonicals
    for all multi-member component K of G_τ **do**
      for all $K' \in \textsc{LLMVerify}(K)$ **do**
        $\textsc{Merge}(K')$ ▷ survivor absorbs the rest
      end for
    end for
  end for
  if pass $= 1$ **then** re-embed merged clusters; recompute S
  end if
end for
return canonical $V^\star$

directly for the top-k events (k=10). *Path B (entity-first)* queries the entity store, canonicalises each hit via its `merged_with` pointer, walks B to gather every event the entity participates in, and reranks that (often large) set by cosine to the query, keeping the top 10. The two seed sets are unioned A-first. The seeds are then expanded by an undirected breadth-first walk over E_{evt} to depth two, since a cause of a retrieved effect is as relevant as an effect of it, though edge direction is preserved in the rendered context. The expanded set is truncated to a 15k-token budget by keeping the largest breadth-first prefix that fits, preserving narrative locality. The context is assembled as three sections: an entities table, an events table, and an event-relationships list emitted in connected-component order so chains stay together. No raw source text is included.

3.4 Algorithmic summary

Algorithms 1–3 state indexing, deduplication, and retrieval; LINK is the bipartite construction of §3.2. Default thresholds and batch sizes are in Appendix A.

Algorithm 3 eSoKG retrieval (no LLM call)

Require: question q, graph G, budget β
$\mathbf{v}_q \leftarrow \textsc{Embed}(q)$
$S^A_{\text{evt}} \leftarrow \textsc{TopK}(V^\star_{\text{evt}}, \mathbf{v}_q)$ ▷ Path A
$S_{\text{ent}} \leftarrow \textsc{Canon}(\textsc{TopK}(V^\star_{\text{ent}}, \mathbf{v}_q))$
$S^B_{\text{evt}} \leftarrow \textsc{TopK}(\textsc{EventsFor}(S_{\text{ent}}, B), \mathbf{v}_q)$ ▷ Path B
$E \leftarrow \textsc{Expand}(S^A_{\text{evt}} \cup S^B_{\text{evt}},\ E_{\text{evt}})$ ▷ 2-hop, undirected
$E^\star \leftarrow \textsc{TruncateToBudget}(E, \beta)$
return $\textsc{Assemble}(E^\star, B, E_{\text{evt}})$

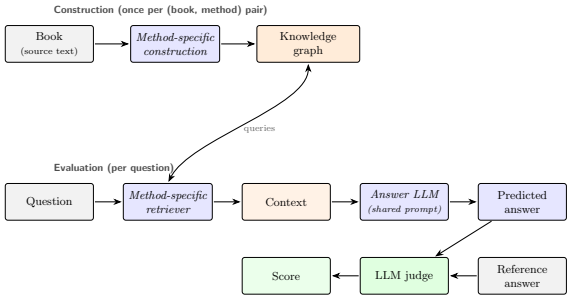


Figure 1: The evaluation. A graph is built once per (book, method) pair and queried per question. The answer LLM, its prompt, and the judge are shared across methods, so the only variable is the graph and its retriever.

4 Experimental Setup

Corpus. We evaluate on 332 long-form questions over five books (Table 2). Four are novels from ChronoQA (Zhang et al., 2026), the narrative-QA benchmark released with E^{2}RAG, chosen to span a range of lengths and question counts; each question is tagged with one of eight reasoning categories (e.g. Causal Consistency, Character Consistency). The fifth, *The Tortured Kingdom* (TTK), is a novel first published serially in March 2026 and included as a contamination-resistant test set: because the four ChronoQA novels are old and famous, an answer LLM can produce correct answers from parametric memory alone, whereas a post-cutoff novel lets retrieval contribution be separated from memorisation. Its 101 questions were generated with Claude Opus 4.7 from ChronoQA's published template (extended to all eight categories) and read for coherence, but not fact-checked against the source.

Methods. Five methods are compared under identical chunking (2000/400): `pureLLM` (no retrieval, raw question, clean parametric baseline); `LightRAG` (Guo et al., 2025) in `hybrid` mode at its default 30k-token budget; `LightRAG-15K`, the same capped at 15k; vanilla E^{2}RAG (Zhang et al., 2026); and eSoKG. LightRAG and E^{2}RAG return graph content *plus* the source chunks backing re-

Book	Src. tok.	#Q	Q words
Pride & Prejudice (P&P)	164k	54	36.5
The Phantom of the Opera	116k	70	26.0
Harry Potter (HP)	112k	25	30.0
The Tortured Kingdom (TTK)	90k	101	63.2
The Wonderful Wizard of Oz	50k	82	23.2
Total	—	**332**	—

Table 2: Evaluation corpus. Source length in `cl100k` tokens; mean question length in words. HP is *Harry Potter and the Philosopher's Stone*; TTK is the post-cutoff novel.

Score	Anchor
10	Matches ground truth exactly or faithful paraphrase; all key points present.
7	Mostly correct; minor omissions or wording that do not change meaning.
5	Partially correct; some key points but major gaps or inaccuracies.
3	Mostly incorrect; only superficial overlap.
1	Off-topic, wrong, or hallucinated.

Table 3: Judge rubric anchors (1–10).

trieved nodes; `eSoKG` is the only method returning graph content with no chunks. `E`^{2}`RAG`'s combined (HyDE) mode is excluded because, for these famous books, the ungrounded hypothetical answer carries ground-truth vocabulary into retrieval, testing memorisation rather than the graph. All three graph methods and their retrievers live in one forked LightRAG codebase.

Models and protocol. Figure 1 summarises the evaluation. One LLM, Claude Sonnet 4.6, serves as indexer, answerer, judge, and the comparators' retrieval-time keyword/cue extractor; one embedding model, Amazon Titan v2 (1024-d), is used everywhere. Sharing models removes model identity as a confound. The graph is built once per (book, method) and reused. Each (book, method, question) is answered five times, since the decoder runs at non-zero temperature. All methods share one answer prompt, adapted from LightRAG's with the output-format constraints removed so that uncited methods are not penalised, and with external knowledge forbidden. An LLM judge scores each answer on a 1–10 rubric from the question, prediction, and reference only. Per-question scores are the mean over generations. Token cost is tracked per stage, attributing in-retriever LLM calls to the retrieval phase; `eSoKG`'s retrieval phase costs zero tokens. We report both the *all-questions* set (332) and the *common clean* set (260 questions every method answered with no refusal on any run). The judge scores on the rubric anchors of Table 3, returning an integer and a short rationale. The full prompts (QA generation, entity/event extraction, answer, and judge) are reproduced in Appendix E. Because the judge shares a model with the indexer and answerer, every answer was also re-scored by two judges from other model families, Qwen3-32B (Yang et al., 2025a) and DeepSeek-V3.2 (DeepSeek-AI, 2024), under the same prompt and rubric at temperature 0. Nothing was regenerated, so only the judge changes.

5 Results

Overall (Table 4). Under the primary judge, Sonnet 4.6, the two averaging schemes agree on the ranking. The four retrieval methods lie within a 0.6-point band, and the no-retrieval `pureLLM` is over two points below the lowest of them by-Q. `eSoKG` is second on the pooled set (7.21 by-Q; 7.79 clean), above `E`^{2}`RAG` (7.12; 7.69) and the budget-matched `LightRAG-15K` (6.73; 7.50), and behind only the full-budget `LightRAG` (7.33; 7.92), which spends roughly twice the per-query tokens (Table 7). It also refuses on the fewest questions of any retrieval method (8.1%). Paired tests over the 332 per-question means (Table 5) make the margins over `LightRAG-15K` and `pureLLM` significant and leave the gap to `LightRAG` and the edge over `E`^{2}`RAG` within noise. How this ordering holds up under two other judges is taken up below (*Independent judges*).

Refusals and the clean set. The shared prompt lets a method emit a fixed refusal string when its context is insufficient; `pureLLM`, which never receives that instruction, never refuses. Removing the union of all methods' refusals leaves the 260-question common clean set, on which the ranking is unchanged (Table 4, *Clean*). Every method gains once refusal-prone questions are removed, but unevenly: `LightRAG-15K` gains most ($+0.77$, consistent with its 14.8% refusal rate), `pureLLM` least ($+0.35$, no refusals). `eSoKG` refuses least among retrieval methods and gains no more than `LightRAG` and `E`^{2}`RAG` do from removing refusals ($+0.58$ against $+0.59$ and $+0.57$), so its standing against those two is not an artefact of abstention. Against `LightRAG-15K`, which refuses on 21 of 54 P&P and 19 of 101 TTK questions (Table A6), it partly is: `eSoKG`'s margin shrinks from $+0.48$ to

Method	Per-book mean (all-questions) HP	P&P	Phantom	Oz	TTK	Average by-Q	by-book	Clean mean	Refuse (%)
LightRAG	**8.35**	6.07	7.94	**7.13**	**7.48**	**7.33**	**7.39**	**7.92**	9.0
eSoKG	8.20	6.81	**8.01**	7.06	6.74	7.21	7.36	7.79	**8.1**
E^{2}RAG	7.74	**7.00**	7.81	6.40	7.16	7.12	7.22	7.69	9.6
LightRAG-15K	8.26	5.05	7.67	6.80	6.53	6.73	6.86	7.50	14.8
pureLLM	7.34	5.47	6.13	4.94	2.47	4.70	5.27	5.05	0.0

Table 4: Mean score (1–10), Sonnet 4.6 judge (other judges: Table 5). *by-Q* averages over all 332 questions; *by-book* weights each book equally; *Clean mean* is over the 260-question common clean set; *Refuse* is the fraction of questions the method refused at least once. The four retrieval methods lie within a 0.6-point band; pureLLM is >2 points below the lowest by-Q. Best per column in bold.

	Sonnet 4.6	Qwen3-32B	DeepSeek-V3.2
Mean of per-question means (rank)			
LightRAG	7.33 (1)	7.79 (1)	8.18 (1)
eSoKG	7.21 (2)	7.54 (3)	8.15 (2)
E^{2}RAG	7.12 (3)	7.63 (2)	7.90 (3)
LightRAG-15K	6.73 (4)	7.27 (4)	7.35 (4)
pureLLM	4.70 (5)	5.01 (5)	4.80 (5)
eSoKG − method, mean [95% CI]			
LightRAG	−0.12 [−0.32, +0.08]	−0.25 [−0.46, −0.03]*	−0.04 [−0.30, +0.22]
E^{2}RAG	+0.08 [−0.13, +0.30]	−0.09 [−0.32, +0.14]	+0.25 [−0.03, +0.53]
LightRAG-15K	+0.48 [+0.26, +0.71]*	+0.27 [+0.02, +0.52]	+0.80 [+0.50, +1.11]*
pureLLM	+2.50 [+2.20, +2.81]*	+2.53 [+2.16, +2.90]*	+3.35 [+2.95, +3.74]*

Table 5: Judge robustness: the original and two independent judges scoring the *same* 8,300 answers. Top: mean over the 332 per-question means (rank). Bottom: paired difference eSoKG minus method, 95% bootstrap CI; *Holm-adjusted Wilcoxon $p < 0.05$. Agreement and per-book values in Appendix D.

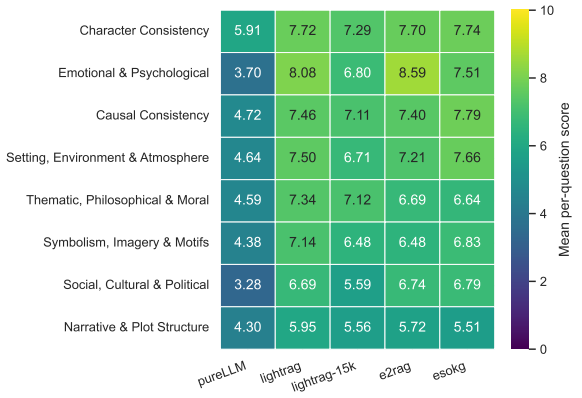


Figure 2: Mean score by category (rows, sorted by cross-method mean) and method, all books pooled, Sonnet 4.6 judge. eSoKG leads the retrieval methods on the consistency-style categories; every retrieval method is weakest on Narrative & Plot Structure.

+0.29 on the clean set.

Independent judges (Table 5). All 8,300 answers were re-scored under the same prompt by Qwen3-32B and DeepSeek-V3.2. Over the 332 per-question means both correlate closely with Sonnet 4.6 (Pearson $r = 0.92$ and 0.94, Spearman $\rho = 0.85$ and 0.91; Table A4) and reproduce the large effects: pureLLM stays more than two points below every retrieval method, and LightRAG, eSoKG, and E^{2}RAG lie within 0.3 points of one another under every judge. Within that band the judges differ. DeepSeek gives Sonnet 4.6's ordering and the same four verdicts. Qwen places E^{2}RAG above eSoKG (−0.09, not significant), makes eSoKG's deficit to LightRAG significant (−0.25), and shrinks its margin over LightRAG-15K from +0.48 to +0.27. That margin's bootstrap CI still excludes zero, but the Holm-adjusted Wilcoxon test, our criterion for significance, does not reject ($p = 0.63$ after Holm adjustment), so under Qwen we count it as not significant. The eSoKG–E^{2}RAG swap under Qwen is a weighting effect. Under every judge eSoKG beats E^{2}RAG on HP and Oz and loses on TTK, and TTK holds 101 of the 332 questions, so the pooled by-Q mean leans on it; averaged by book, eSoKG leads E^{2}RAG under Sonnet 4.6 and DeepSeek and ties under Qwen (Table A4). The ordering among the top three therefore depends on the judge and on the weighting. What holds under every judge is that eSoKG lies within 0.3 points of the best method.

By question type (Fig. 2). The per-category pattern fits the event-centric design. On the pooled set eSoKG leads the retrieval methods on Causal Consistency (7.79) and Character Consistency (7.74), the categories that ask how events and character state cohere across the book; the Causal lead holds under all three judges, the Character lead under two (Table A5). E^{2}RAG leads on Emotional & Psychological, which rewards local event detail, and LightRAG on Thematic and Symbolism, which reward broad topical neighbourhoods. Every retrieval method scores lowest on Narrative & Plot Structure, where whole-book plot reasoning is hard whatever the graph.

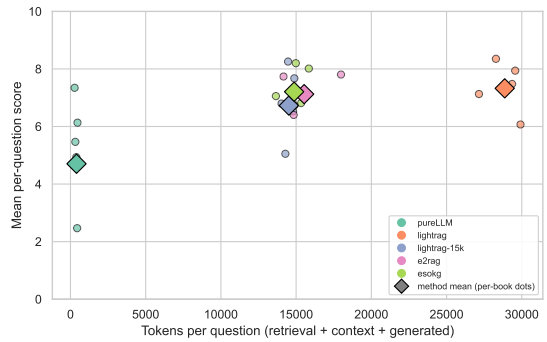


Figure 3: Quality vs. operational cost: mean judge score against mean total tokens per query. Small markers are (book, method) pairs; large diamonds are method means. eSoKG, E^{2}RAG, and LightRAG-15K form a low-cost cluster at similar quality; LightRAG buys a small edge at almost twice the tokens.

Method	Calls	Prompt tok	Total tok
LightRAG	206	1.09M	1.34M
E^{2}RAG	142	0.79M	1.12M
eSoKG	474	2.18M	**3.27M**
Ph.1 (extract)	220	1.23M	2.12M
Ph.2 (dedup)	254	0.94M	1.15M

Table 6: One-time construction cost for *Harry Potter* (2000/400 chunking). eSoKG is the most expensive to build; the two indented rows decompose it.

Method	Retr.	Ctx.	Gen.	Total
pureLLM	0	0	402	402
LightRAG	808	27,470	590	28,867
LightRAG-15K	808	13,186	522	14,516
E^{2}RAG	516	14,476	542	15,534
eSoKG	**0**	14,169	698	14,867

Table 7: Mean per-query tokens by stage (over all books). **Retr.**: pre-retrieval keyword/cue LLM call. **Ctx.**: retrieved context. **Gen.**: answer tokens. eSoKG pays nothing at retrieval. Ctx. counted with cl100k; Retr. and Gen. as reported by the API. LightRAG-15K shares LightRAG's retrieval call.

Cost (Table 7, Fig. 3). Per query, eSoKG sits in the low-cost cluster (14.9k tokens), close to LightRAG-15K (14.5k) and E^{2}RAG (15.5k) and about half of full-budget LightRAG (28.9k). What is distinctive is the *shape* of the cost: eSoKG is the only method with zero retrieval-stage tokens, because it retrieves via embeddings and a graph walk with no pre-retrieval LLM call, and spends its budget on the answer instead (it writes the longest answers on every book, ∼20% longer than LightRAG). Within the low-cost cluster it has the highest pooled mean under Sonnet 4.6 and DeepSeek, and the second highest, 0.09 behind E^{2}RAG, under Qwen (Table 5). The drawback is the build cost. Building the HP graph cost eSoKG 3.27M tokens against 1.34M (LightRAG) and 1.12M (E^{2}RAG), split between QA-mediated extraction (2.12M) and deduplication (1.15M). Against full-budget LightRAG the per-query saving overtakes this build premium only after ∼140 queries on this book; against the cheaper LightRAG-15K it never does. Table 6 decomposes the build cost; per-book graph sizes are in Appendix B, where eSoKG's merging is visible as far fewer entity nodes than the unmerged E^{2}RAG (438 vs. 1,592 on HP).

Post-cutoff novel. TTK is the one book the models are very unlikely to have seen in pre-training. pureLLM collapses from 7.34 on HP to 2.47 here while every retrieval method holds between 6.53 and 7.48 (Table 4), so retrieval contributes beyond memory, and the other books' scores partly reflect recall. TTK shows no eSoKG advantage in overall score. eSoKG (6.74) sits behind LightRAG (7.48) and E^{2}RAG (7.16) and above LightRAG-15K (6.53) under all three judges, and it leads no category (Table A7). The shape of its scores across categories is nevertheless close to the pooled one. Setting and Character remain its two strongest categories, and Narrative, Thematic, and Symbolism its three weakest. The one change is Causal Consistency, its best category on the pooled set and fifth of eight here. Since this profile appears on a book the model has not memorised, it more likely comes from the method than from recall.

Qualitative example. Figure 4 shows what each graph retrieves for one question: *"who gave Hagrid the dragon egg in the Hog's Head, and why was it significant?"* Answering it means linking the pub exchange to Hagrid's later disclosure of how to get past Fluffy. eSoKG retrieves both events as named, connected nodes, with Quirrell, Harry, and Ron each a single merged node. E^{2}RAG, which does not merge, leaves Hagrid, the egg, and Fluffy spread over several nodes and surfaces the dragon's aftermath instead of the pub exchange. LightRAG returns a broad topical neighbourhood with no event nodes and with mentions left apart (*Quirrell* and *Professor Quirrell* are separate nodes). eSoKG's merging is imperfect too: *Hagrid* and *Dumbledore* each appear as two nodes here because a late description drifted past the embedding threshold, the

case a name check would catch (Appendices C and F).

6 Discussion

RQ1 (quality): yes on the pooled set, not on the post-cutoff book. On the 332 pooled questions a context with *no source text* scores within 0.3 points of the two methods that hand the model the passages, under every judge, and neither gap is significant under Sonnet 4.6 (Qwen makes the gap to LightRAG significant, -0.25). For these books the distilled structure carries enough to answer. On TTK it does not match: eSoKG trails both graph comparators there, so part of the pooled parity reflects books the model already knows. The result is also bounded by the setup. The same model wrote the structure and answered from it, so what is shown is that the structure preserves enough *for that model*, not that the graph is complete on its own.

Why the gap to LightRAG is small. LightRAG's edge, 0.12 points under Sonnet 4.6 and 0.04 to 0.25 under the independent judges, tracks its budget: it feeds ∼27.5k context tokens to eSoKG's 14.2k, and at a matched budget LightRAG-15K falls *below* eSoKG under all three judges, significantly under two. What the budget buys is mostly coverage. The two books where LightRAG-15K refuses most, P&P and TTK, are the two where LightRAG beats it by about a point under every judge. On HP and Phantom, where refusals are rare, the gap closes, and under Qwen LightRAG-15K scores above every other method on both books (Table A4). eSoKG's margin over LightRAG-15K is likewise concentrated on P&P, and the Qwen result shows what kind of margin it is: the mean difference stays positive because refusals score near the floor, while the Wilcoxon signed-rank test, which is insensitive to a tail of large differences, finds the median per-question difference indistinguishable from zero. The matched budget also shows a coverage effect in eSoKG's favour. The same 15k tokens are enough to answer more often when they hold events than when they hold entity summaries (8.1% refusals against 14.8%), and where both methods answer, eSoKG still leads under Sonnet 4.6 ($+0.29$ on the clean set) and DeepSeek, and ties under Qwen (Appendix D). The event-centric context therefore covers more of what the questions ask for per token; whether it also answers better where both have enough, two of three judges say it does and one says it does not. eSoKG's longer answers remain an untested confound: length sensitivity is not measured here, and the one published result for the judge's model family finds a conciseness rather than a verbosity preference (Soumik, 2026). That a model relies more on its context when the context gives it more evidence is consistent with (Sun et al., 2026).

RQ2 (cost): competitive per query, costly to build. Per query eSoKG is parity-or-better against the cheap methods and a clear saving only against the full-budget one; its real premium is at build time, where the QA-mediated extraction and the cross-chunk merging together cost 2–3× the comparators' tokens. The per-query saving offsets this only slowly and only against the full-budget comparator (§5), so as a drop-in retrieval index for one-shot QA the build cost is hard to justify. Whether it is worth paying depends less on query volume than on what the graph is *used* for, which the next point takes up.

What the structure supports beyond QA. The evaluation measures one use of the graph, but the merged event-centric structure supports operations the comparators cannot serve directly. Because events are first-class nodes and merging consolidates each entity into one, and because the chunk-free design makes the structure *the graph* rather than an index into text, several useful queries are short graph walks needing no language model: the events two characters share (the overlap of their event neighbourhoods), what led to and followed an event (its incoming and outgoing edges), and a character's arc (the ordered chain of events they take part in). LightRAG has no event nodes to walk between; E^{2}RAG has them but leaves the same character and event split across unmerged copies, so any such query must first regather the fragments, which is the work eSoKG's merging already did. These are properties of the representation, not measured results. They are, however, the operations a graph-navigating agent (Ghandi et al., 2026; Ni et al., 2025) would compose, and the one-shot RAG setting of our evaluation exercises none of them, which is probably the setting that least favours a traversable graph.

7 Conclusion

We asked whether an event-centric KG built with 5W1H QA-mediated extraction and cross-chunk

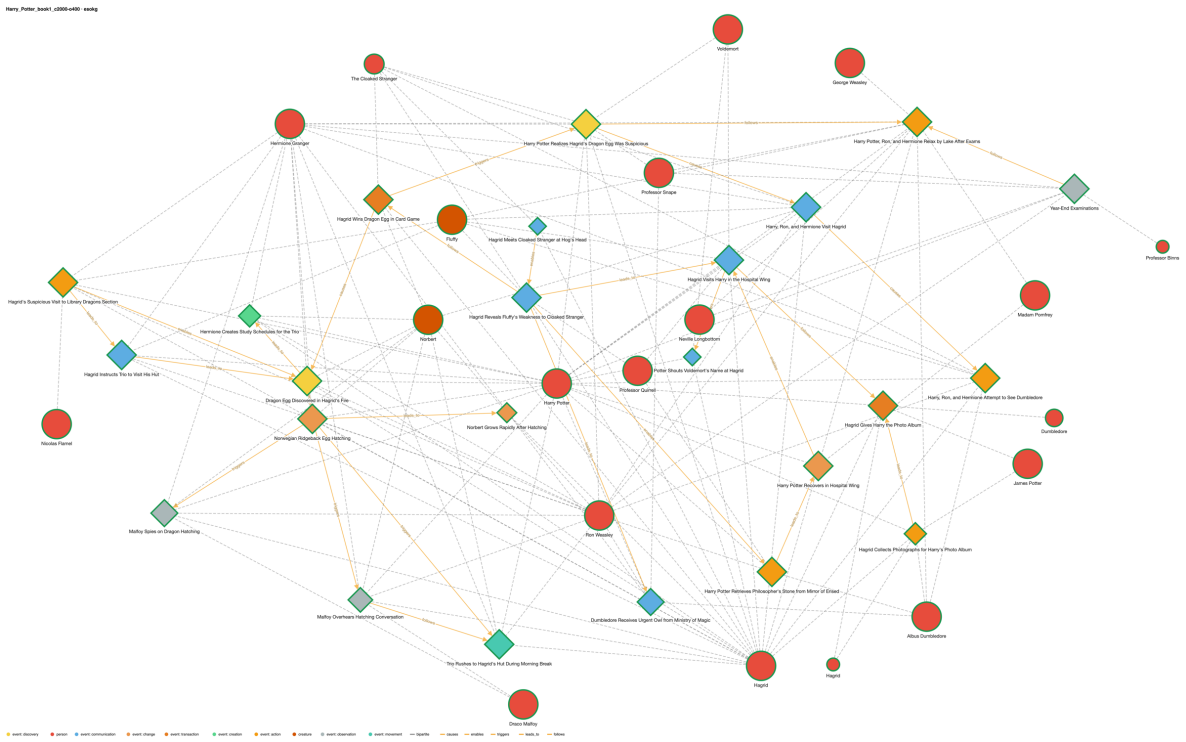


Figure 4: Subgraph eSoKG retrieves for the query *"who gave Hagrid the dragon egg in the Hog's Head, and why was it significant?"* Circles are entities, diamonds are events (coloured by subtype), and edges are typed participation and causal/temporal links (*causes*, *enables*, *triggers*, *leads_to*, *follows*). The two events the question turns on, *Hagrid Meets Cloaked Stranger at Hog's Head* and *Hagrid Reveals Fluffy's Weakness to Cloaked Stranger*, are both retrieved and connected, and major characters (Harry, Quirrell, Ron) appear as single merged nodes. Merging is imperfect: *Hagrid* and *Dumbledore* each appear here as two nodes (Appendix C gives whole-graph counts).

merging by description, queried chunk-free, can answer narrative questions as well as existing graph-based retrieval (RQ1), and at what cost (RQ2). On quality, the pooled evaluation places eSoKG within half a point of an unmerged entity–event comparator and of a full-budget entity-centric one that spends twice the per-query tokens, under the original and two independent judges, with no source text shown to the model. What is statistically robust is narrower than that. Under every judge eSoKG beats the no-retrieval baseline by more than two points, and under two of three it beats LightRAG-15K, the entity-centric method at a matched budget. Its gap to LightRAG and its edge over E^{2}RAG are within noise under Sonnet 4.6 and DeepSeek; under Qwen the gap to LightRAG becomes significant and E^{2}RAG moves ahead by a non-significant margin. The ordering among eSoKG, E^{2}RAG, and LightRAG therefore depends on the judge. Merging mentions by description leaves the graph able to answer causal-consistency questions, where eSoKG leads under every judge, and character-consistency questions, where it leads under two of three, which is the opposite of what the closest prior work assumes. The parity does not extend to the one post-cutoff book, where eSoKG trails both graph comparators, so part of it reflects books the model already knows. On cost, the saving is per query and the price is a one-time build two to three times the comparators'. The merged event-centric graph also answers character-arc, shared-event, and causal-chain queries as direct graph walks, which the comparators cannot serve directly and which the one-shot evaluation does not exercise. For event-centric narrative KGs, description-based merging is a workable alternative to keeping mentions separate on the books tested here. Confirming it needs several post-cutoff novels and a judge closer to a human reader.

Limitations

The evaluation is a whole-system comparison. eSoKG differs from the comparators in four coupled ways (event schema, 5W1H extraction, description merging, chunk-free context) and is not internally ablated, so no result can be credited to one choice

alone. Separating them is the natural next step.

The primary judge is the same model that builds the graphs and writes the answers, and it has read most of the books, so absolute scores partly reflect recall and small gaps are unreliable. The pooled ranking is stable across the all-questions and clean views, and `pureLLM`'s collapse on the post-cutoff book shows the judge punishes ungrounded answers. Re-scoring every answer with two independent judges (Table 5) reproduces the large effects, but the two disagree on the fine ordering. One reproduces Sonnet 4.6's ranking and all four verdicts. The other swaps `eSoKG` and `E²RAG`, makes the deficit to `LightRAG` significant, and leaves the margin over `LightRAG-15K` positive in mean but not under the signed-rank test. The ordering among retrieval methods is therefore judge-dependent, and no LLM judge here can show which is closer to a human reader. `eSoKG` also writes the longest answers, and the judges' sensitivity to answer length is not measured.

Pooled means fold refusals, scored near the floor, into answer quality, and weight TTK's 101 questions four times HP's 25. The clean set and the by-book average (Table 4) separate these effects, but the paired tests inherit the pooled weighting.

On the one contamination-free book `eSoKG` trails both graph comparators, so the pooled parity partly reflects books the model already knows. A firmer claim needs several post-cutoff novels. TTK's questions are also longer than the ChronoQA books' (63.2 vs. 23–37 words) and were generated by a different model from an extended template, so the HP–TTK contrast varies question style as well as contamination.

All questions and reference answers are LLM-generated and not human-verified, so the evaluation measures agreement with an LLM-written reference. Construction-cost figures come from one book. The cross-chunk merging is not guaranteed to succeed: a character whose mentions fail to merge is split over several nodes, and the evaluation does not measure how often this happens or what it costs.

Acknowledgements

The author thanks Ricky Molén for insightful discussions on the design and evaluation, and the anonymous reviewers for their detailed and constructive feedback. The author used Claude Code for language polishing, code assistance, and analysis scripting; all technical content, experimental results, and conclusions were verified by the author.

References

Jiaxin Bai, Wei Fan, Qi Hu, Qing Zong, Chunyang Li, Hong Ting Tsang, Hongyu Luo, Yauwai Yim, Haoyu Huang, Xiao Zhou, Feng Qin, Tianshi Zheng, Xi Peng, Xin Yao, Huiwen Yang, Leijie Wu, Yi Ji, Gong Zhang, Renhai Chen, and Yangqiu Song. 2025. AutoSchemaKG: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora.

Sanghyeok Choi, Woosang Jeon, Kyuseok Yang, and Taehyeong Kim. 2026. SocraticKG: Knowledge graph construction via QA-driven fact extraction.

Peter. Christen. 2012. *Data matching : concepts and techniques for record linkage, entity resolution, and duplicate detection*, 1st ed. 2012. edition. Data-centric systems and applications. Springer-Verlag.

DeepSeek-AI. 2024. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph RAG approach to query-focused summarization.

Taraneh Ghandi, Hamidreza Mahyar, and Shachar Klaiman. 2026. GraphWalk: Enabling reasoning in large language models through tool-based graph navigation. *Preprint*, arxiv:2604.01610 [cs.AI].

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China. Association for Computational Linguistics.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Luu Anh Tuan. 2025. HyperGraphRAG: Retrieval-augmented generation via hypergraph-structured knowledge representation.

Tengjun Ni, Xin Yuan, Shenghong Li, Kai Wu, Ren Ping Liu, Wei Ni, and Wenjie Zhang. 2025. StepChain GraphRAG: Reasoning over knowledge graphs for multi-hop question answering. *Preprint*, arxiv:2510.02827 [cs.CL].

Sadman Kabir Soumik. 2026. Judging the judges: A systematic evaluation of bias mitigation strategies in LLM-as-a-judge pipelines.

Kaiser Sun, Fan Bai, and Mark Dredze. 2026. Task matters: Knowledge requirements shape LLM responses to context-memory conflict.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2024. ChatIE: Zero-shot information extraction via chatting with ChatGPT. *Preprint*, arxiv:2302.10205 [cs.CL].

Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1300–1307.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zairun Yang, Yilin Wang, Zhengyan Shi, Yuan Yao, Lei Liang, Keyan Ding, Emine Yilmaz, Huajun Chen, and Qiang Zhang. 2025b. EventRAG: Enhancing LLM generation with event knowledge graphs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16967–16979. Association for Computational Linguistics.

Ze Yu Zhang, Zitao Li, Yaliang Li, Bolin Ding, and Bryan Kian Hsiang Low. 2026. Respecting temporal-causal consistency: Entity-event knowledge graph for retrieval-augmented generation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2017–2054, Rabat, Morocco. Association for Computational Linguistics.

A Implementation Defaults

Table A1 lists the configuration used by the eSoKG pipeline (§3.2). These are configuration, not tuned hyperparameters: they were fixed by light iteration during development on a held-out short story (Liam O'Flaherty's *The Sniper*, used nowhere in the evaluation) and were not re-tuned on the evaluation corpus.

B Graph Statistics

Table A2 reports the size and shape of each method's graph per book, after deduplication. Three structural differences are explicit in the counts: eSoKG has *no* entity–entity edges (entities relate only through shared events); LightRAG has no event nodes or bipartite edges (it is entity-centric); and eSoKG has far fewer entity nodes than the unmerged E^{2}RAG on every book, a direct consequence of its cross-chunk merging.

Parameter	**Default**
Entity dedup thresholds (pass 1 & 2)	0.8, 0.7, 0.6, 0.5
Event dedup thresholds (pass 1 & 2)	0.85, 0.75, 0.65
Max / min candidates per LLM dedup call	20 / 15
Concurrent dedup calls	4
Max extraction output tokens	16384
Per-chunk retries	3
Chunk size / overlap (eval)	2000 / 400
Retrieval seeds per path (k)	10
Expansion depth (hops)	2
Retrieval token budget	15k

Table A1: eSoKG configuration defaults. Event thresholds are kept above entity thresholds because a false event merge is unrecoverable.

Book	**Method**	**Ent.**	**Evt.**	**E–E**	**Ev–Ev**	**Bip.**
HP	eSoKG	438	770	0	808	4,629
	LightRAG	706	—	1,167	—	—
	E^{2}RAG	1,592	1,310	1,197	860	3,819
TTK	eSoKG	378	716	0	773	3,648
	LightRAG	544	—	843	—	—
	E^{2}RAG	972	1,219	818	845	2,922
P&P	eSoKG	264	882	0	934	5,050
	LightRAG	524	—	1,040	—	—
	E^{2}RAG	1,787	1,650	1,401	1,072	4,696
Oz	eSoKG	144	335	0	355	1,891
	LightRAG	229	—	390	—	—
	E^{2}RAG	548	649	496	487	1,885
Phantom	eSoKG	421	686	0	757	3,472
	LightRAG	649	—	1,009	—	—
	E^{2}RAG	1,147	1,156	969	791	2,956

Table A2: Node and edge counts per book and method after deduplication. **Ent./Evt.**: entity/event nodes. **E–E**, **Ev–Ev**, **Bip.**: entity–entity, event–event, bipartite edges. A dash means the schema has no such element.

C Deduplication on *Harry Potter*

The *Harry Potter* graph has 438 entity nodes after Phase 2. Table A3 reports, per principal character, how many person nodes survive, the degree of the dominant node, and the degrees of any stragglers. Five of ten characters consolidate to a single node; the rest keep a dominant node plus one to three stragglers whose degrees are in the single digits against a dominant node of 63 or more. Fragmentation, where it happens, is minor in weight, and the missed merges are exactly the cases a name check would catch (the stragglers share the dominant node's surface name). LightRAG, which merges by name, fails the opposite way: on the same book its graph keeps *Quirrell* and *Professor Quirrell* as separate nodes where eSoKG consolidates both. The two failure modes are mirror images, motivating

the hybrid name+description criterion suggested in §6. These are illustrative examples, not a measured merge-accuracy comparison.

Character	Nodes	Dom. deg.	Straggler deg.
Harry Potter	1	611	—
Ron Weasley	4	294	6, 4, 2
Hermione Granger	4	202	3, 2, 1
Hagrid	3	189	3, 0
Albus Dumbledore	2	102	5
Professor Snape	1	88	—
Neville Longbottom	1	87	—
Voldemort	1	80	—
Draco Malfoy	2	77	2
Professor Quirrell	1	63	—

Table A3: Surviving person nodes per principal character after Phase 2, ordered by dominant-node degree. A dash means the character consolidated to a single node.

D Supplementary Results

How scores are computed. Every score in this paper is a mean of per-question means. Each (book, method, question) triple is answered five times; a judge scores each of the five answers once on the 1–10 rubric of Table 3; the five scores are averaged into a per-question mean; and a table cell is the mean of those per-question means over the questions in its scope (a book, a category, the 332-question pooled set, or the 260-question clean set). A table that does not name a judge reports the primary judge, Sonnet 4.6. The two independent judges, Qwen3-32B and DeepSeek-V3.2, scored the same 8,300 answers and appear only where a caption names them. Refusals are detected from the answer text and do not depend on any judge.

D.1 Independent judges

Table A4 gives the per-book means under the two independent judges behind Table 5, with per-book agreement with Sonnet 4.6. On the 260-question clean set the orderings are: Qwen3-32B `LightRAG` 8.39, `E²RAG` 8.19, `LightRAG-15K` 8.18, `eSoKG` 8.12, `pureLLM` 5.45; DeepSeek-V3.2 `eSoKG` 8.96, `LightRAG` 8.95, `E²RAG` 8.65, `LightRAG-15K` 8.44, `pureLLM` 5.32. Table A5 gives the per-category means under all three judges.

D.2 Refusals

Table A6 gives, per book, the questions on which each retrieval method refused at least once; removing their union leaves the 260-question common clean set used in §5.

	HP	P&P	Phant.	Oz	TTK	by-book
Qwen3-32B						
`LightRAG`	8.82	6.59	8.11	**7.54**	**8.15**	**7.84**
`eSoKG`	8.66	6.91	7.95	7.41	7.42	7.67
`E²RAG`	8.14	**7.26**	8.08	7.00	7.89	7.67
`LightRAG-15K`	**8.84**	5.55	**8.17**	7.30	7.15	7.40
`pureLLM`	7.72	5.66	6.78	5.76	2.16	5.62
r / ρ vs. Sonnet 4.6	.76 / .68	.93 / .89	.82 / .68	.93 / .90	.94 / .89	.92 / .85
Offset vs. Sonnet 4.6	+0.46	+0.31	+0.30	+0.54	+0.48	+0.43
DeepSeek-V3.2						
`LightRAG`	**9.41**	6.54	9.07	7.84	**8.42**	8.26
`eSoKG`	**9.41**	**7.54**	**9.29**	**7.87**	7.59	**8.34**
`E²RAG`	8.66	7.37	9.00	7.03	7.92	8.00
`LightRAG-15K`	9.34	5.00	8.79	7.51	6.97	7.52
`pureLLM`	8.10	4.90	6.74	5.46	2.05	5.45
r / ρ vs. Sonnet 4.6	.85 / .72	.94 / .93	.91 / .84	.95 / .95	.93 / .91	.94 / .91
Offset vs. Sonnet 4.6	+1.01	+0.19	+1.07	+0.68	+0.52	+0.66

Table A4: Independent judges, all-questions set: per-book mean of per-question means (best per column in bold within each judge), Pearson r / Spearman ρ with Sonnet 4.6 over per-question means, and mean offset; the last column pools all 332 questions. By category, all three judges give `eSoKG` the lead on Causal and Setting; Character and Social hold under DeepSeek and pass to `LightRAG` under Qwen. Neither judge ever returned 2, 4, or 6, and both returned valid JSON on all 8,300 calls.

D.3 Post-cutoff book

Table A7 gives the per-category breakdown on the post-cutoff *The Tortured Kingdom* under Sonnet 4.6, where `pureLLM` collapses across the board while the retrieval methods hold.

E Prompts

This appendix reproduces the prompts as run; the worked example in the event-extraction prompt has been lightly edited for the paper. Braced tokens (e.g. `{chunk_text}`, `{context}`) are placeholders filled at runtime.

E.1 Question generation (post-cutoff set)

The question set for *The Tortured Kingdom* was generated with Claude Opus 4.7 from the template below, extended from ChronoQA's published template (which ships only a Causal-Consistency worked example) to cover all eight categories, one generation pass per category. `{{DOCUMENT}}` is replaced with the source text and `{{CATEGORY_BLOCK}}` with one block per pass.

Question-generation master template

```
I want to test a few rag systems on their reasoning
      capabilities and the capability to pick up nuanced
      details. Use the document shown below, design
      queries for it, also provide the ground truth for
      each query. In addition, it should focus on testing
      the RAG system's causal consistency. for example, 1.
       it can ask the RAG system how would a certain
      character behavior in a given context/even from the
      document, and see if the answer is consistent with
      the character's personality/traits at that specific
```

Category	Sonnet 4.6				Qwen3-32B				DeepSeek-V3.2			
	LR	eSoKG	E^2	LR15	LR	eSoKG	E^2	LR15	LR	eSoKG	E^2	LR15
Causal Consistency	7.46	**7.79**	7.40	7.11	8.06	**8.26**	7.98	7.99	8.53	**9.05**	8.19	7.86
Character Consistency	7.72	**7.74**	7.70	7.29	**8.27**	8.05	8.12	7.77	8.63	**8.73**	8.42	7.84
Emotional & Psychological	8.08	7.51	**8.59**	6.80	8.45	7.95	**9.18**	7.28	8.58	8.18	**9.53**	7.28
Narrative & Plot Structure	**5.95**	5.51	5.72	5.56	**6.33**	5.67	5.98	6.02	**6.92**	6.59	6.41	6.21
Setting, Env. & Atmosphere	7.50	**7.66**	7.21	6.71	8.02	**8.28**	7.96	7.37	8.29	**8.51**	8.11	7.04
Social, Cultural & Political	6.69	**6.79**	6.74	5.59	**7.36**	7.17	7.25	6.21	7.32	**7.61**	7.44	6.05
Symbolism, Imagery & Motifs	**7.14**	6.83	6.48	6.48	**7.43**	7.08	7.13	6.99	**7.99**	7.76	7.24	7.30
Thematic, Phil. & Moral	**7.34**	6.64	6.69	7.12	**7.57**	6.64	6.81	7.28	**8.23**	7.32	7.42	7.95

Table A5: Per-category mean of per-question means for the four retrieval methods under each judge, all books pooled (LR=LightRAG, E^2=E^2RAG, LR15=LightRAG-15K); best retrieval method per category and judge in bold. eSoKG leads Causal and Setting under every judge, Character and Social under Sonnet 4.6 and DeepSeek; E^2RAG leads Emotional and LightRAG leads Narrative, Symbolism, and Thematic under every judge. pureLLM (not shown) is lowest in every cell.

Method	HP	P&P	Phantom	Oz	TTK	Total
LightRAG	0	13	0	9	8	30
eSoKG	1	8	0	4	14	27
E^2RAG	2	9	0	7	14	32
LightRAG-15K	0	21	0	9	19	49
Questions	25	54	70	82	101	332
In clean set	23	31	70	69	67	260

Table A6: Questions on which each retrieval method refused at least once, by book, with questions per book and questions remaining in the common clean set. pureLLM never refuses. Refusals are detected from the answer text and do not depend on the judge.

```
      point in time (character can experience development
      so their response to the same thing can very). 2. it
       can test if the RAG system confuse the event that
      has not happened but in the document with the query
      (the future events that has not happen should not be
       account for in the answer). Give the queries and
      ground truth in json format. The document is here:

{{DOCUMENT}}

Focus on Causal Consistency: Apart from character behavior
       over time and future event confusion, other causal
      consistency tests (e.g., testing how well it
      understands cause-and-effect relationships in the
      plot) can also be included. Query Difficulty: query
      should involve complex, nuanced reasoning/
      understanding of the document provided.

In this run, generate questions ONLY for the following
      category:

{{CATEGORY_BLOCK}}

Each query object must have EXACTLY three fields: `query`,
       `ground_truth`, and `type`.

Format example:
{
  "queries": [
    {
      "query": "During Harry's first night at Hogwarts,
      the Gryffindor students climb the moving staircases.
       How does the text describe the corridors and
      stairways' magical behavior, and what is the
      immediate impact on Harry's sense of direction?",
      "ground_truth": "The staircases sometimes change
      direction, doors can vanish or move, and some
```

Category	pLLM	LR	LR15	E^2RAG	eSoKG
Char.	1.84	7.06	5.98	**7.60**	7.48
Causal	2.32	**7.46**	6.91	6.98	6.96
Narr.	3.02	**6.50**	5.52	5.84	4.90
Set.	1.89	**7.80**	6.68	7.60	7.67
Social	1.35	**7.75**	6.03	7.34	7.03
Emot.	1.35	8.06	7.20	**8.18**	7.17
Sym.	3.10	**6.78**	5.44	6.04	5.90
Theme	5.96	**8.00**	7.96	7.26	5.90
Overall	2.47	**7.48**	6.53	7.16	6.74

Table A7: Per-category mean score on the post-cutoff *The Tortured Kingdom*, all questions, Sonnet 4.6 judge. pLLM=pureLLM, LR=LightRAG, LR15=LightRAG-15K; best per row in bold.

```
      require a password or a specific tickle of a
      doorknob to open. This constant shifting confuses
      new students like Harry, making it easy to get lost
      early on.",
      "type": "Causal Consistency"
    }
  ]
}

It is also important to note that do not explicitly
      disclose the title/chapter/section number from which
       the context of the question is used. Simply
      described the related event and the characters
      involved to make the context clear.
```

Category blocks (one per generation run)

```
Category: Character Consistency
Description: Questions probing character motivations,
      development, or psychology. Especially valuable:
      questions whose correct answer depends on WHERE in
      the story the reference point falls, because
      characters develop and a single character's "right"
      response can change between early and late story.
Set "type" field to: "Character Consistency"

Category: Causal Consistency
Description: Questions about cause and effect, logical
      sequences of events, or explanations of how events
      unfolded. Include some questions that test whether
      the system avoids leaking future events (events that
       have not yet happened relative to the query's
      reference point) into its answer.
```

```
Set "type" field to: "Causal Consistency"

Category: Symbolism, Imagery & Motifs
Description: Questions about symbolic elements, recurring
      imagery, or metaphorical representations in the
      story.
Set "type" field to: "Symbolism, Imagery & Motifs"

Category: Thematic, Philosophical & Moral
Description: Questions about deeper meanings,
      philosophical ideas, or ethical implications raised
      by the story.
Set "type" field to: "Thematic, Philosophical & Moral"

Category: Narrative & Plot Structure
Description: Questions about story organization, plot
      devices, or narrative techniques (e.g., framing,
      foreshadowing, point of view shifts).
Set "type" field to: "Narrative & Plot Structure"

Category: Setting, Environment & Atmosphere
Description: Questions about physical locations, time
      periods, or mood/atmosphere of specific scenes.
Set "type" field to: "Setting, Environment & Atmosphere"

Category: Social, Cultural & Political
Description: Questions about societal contexts, cultural
      elements, or political dimensions of the story.
Set "type" field to: "Social, Cultural & Political"

Category: Emotional & Psychological
Description: Questions about emotional responses,
      psychological states, or mental processes of
      characters at specific moments.
Set "type" field to: "Emotional & Psychological"
```

E.2 Extraction prompts (Phase 1)

The three sequential per-chunk calls of §3.2: QA generation (Step 1), entity extraction (Step 2), and event extraction with intra-chunk relations (Step 3).

QA-generation prompt (Step 1)

```
## ROLE
You are a **Comprehensive Knowledge Archivist** who
      converts the [Full Document] into detailed, document-
      grounded QA pairs.

## OBJECTIVE
Extract as many meaningful Question-Answer pairs as
      possible from the document.
Use the 5W1H perspectives (Who, What, When, Where, Why,
      How) **as analytical lenses** to help you identify
      and expand potential questions, but do NOT restrict
      yourself to producing only 5W1H-type questions.
Your goal is to maximize informational coverage, capturing
       every explicit fact, relation, event, definition,
      rationale, and process described in the document.

## INPUT
Full Document: "{chunk_text}"

## CONSTRAINTS
1. **Context-Independent**
   - Each QA must be self-contained and understandable
      without referencing the original text.
   - Replace pronouns with explicit entities.

2. **No Hallucination**
   - Use only facts explicitly stated in the document.

3. **Expansion-Oriented Thinking**
   - For each sentence or factual unit, consider the 5W1H
      perspectives as prompts to explore:
     - WHO is involved?
     - WHAT happened or is described?
     - WHEN did it occur?
     - WHERE did it occur?
     - WHY did it occur?
     - HOW was it carried out?
   - These perspectives are **guides** to inspire multiple
       possible QA pairs, even if they are implicit or
      only partially expressed.

4. **Coverage**
   - Extract all possible QA pairs that can be reasonably
      derived from the document.

## OUTPUT FORMAT
Return a JSON list of QA objects:
[
  {"question": "...", "answer": "..."},
  ...
]
```

Entity-extraction prompt (Step 2)

```
---Role---
You are a Named Entity Specialist responsible for
      identifying and describing entities from a set of
      document-grounded Question-Answer pairs.

---Context---
The QA pairs below were generated from a single chunk of a
       source document using 5W1H-guided analysis. Each QA
       pair is self-contained with pronouns resolved to
      explicit entity names.

---Entity Definition & Boundary Rules---

An entity is a specific, nameable person, place,
      organization, object, concept, or other distinct
      thing that participates in, is affected by, or is
      otherwise central to the events and facts described
      in the QA pairs.

**Inclusion criteria** -- extract it if:
- It is a named or uniquely identifiable thing (proper
      noun, specific concept, named method, titled work,
      etc.).
- It plays an active role (agent, patient, instrument) in
      at least one event or fact in the QA pairs.
- It would warrant its own node in a knowledge graph
      extracted from these QA pairs.

**Exclusion criteria** -- do NOT extract:
- Generic temporal expressions ("six months", "recently",
      "last year") unless they name a specific period (e.g
      ., "Q3 2023", "The Great Depression").
- Bare quantities or metrics ("18%", "$5 million", "three
      times") unless they are the name of something
      specific.
- Vague or generic noun phrases that do not refer to a
      distinct entity ("the situation", "the issue", "the
      problem", "the process").
- Adjectives, descriptors, or properties used in passing
      ("large-scale", "innovative").
- Common nouns used generically ("a company", "some
      researchers") rather than referring to a specific
      entity.


---Entity Type Definitions---
Classify each entity using exactly one of the following
      types. Use "Other" only if none fit.
- Person: A specific named individual (real or fictional).
- Creature: A specific named non-human living being or
      species discussed as a distinct entity.
- Organization: A named company, institution, agency, team
      , or formal group.
- Location: A named geographic place, region, facility, or
       address.
- Concept: A specific named idea, theory, framework, field
      , or principle that the text discusses substantively
       (not merely references in passing).
- Method: A specific named technique, algorithm, process,
      or methodology.
- Content: A named creative or informational work (article
      , book, report, dataset publication, speech, etc.).
- Data: A specific named dataset, database, corpus, or
      data collection.
- Artifact: A named physical or digital object, tool,
      system, product, or piece of infrastructure.
```

```
- NaturalObject: A named natural entity (river, mountain,
      species, chemical compound, celestial body, etc.).


**Special guidance for Concept and Method types:**
Only extract a Concept or Method if it is specific and
      named enough to be a distinct knowledge graph node.
      "Machine learning" as a general field mentioned in
      passing: do not extract. "Retrieval-Augmented
      Generation" as a specific technique discussed in the
       text: extract. When in doubt, ask: "Does the text
      say something *about* this concept, or merely use it
       as a descriptor?"


---Canonicalization & Deduplication Rules---

If the same real-world entity is referred to by different
      surface forms within the QA pairs, unify them under
      a single canonical entry.

- **Canonical name:** Use the most complete, formal, or
      unambiguous version. Prefer full names over
      abbreviations (e.g., "World Health Organization"
      over "WHO"), but use the form most recognizable in
      context.
- **Aliases:** List ALL surface-form variants found in the
       QA pairs, including the canonical name itself.

**Specific merge/split rules:**
- **Name variants:** "Barack Obama", "Obama", "Barack
      Hussein Obama" -> merge. Canonical: "Barack Obama".
- **Abbreviations:** "WHO" and "World Health Organization"
       -> merge. List both as aliases.
- **Role-based references:** If the text uses a role ("the
       CEO") and a name ("Sarah Chen") to refer to the *
      same individual*, merge them. Include the role-based
       form in aliases. However, if a role refers to the
      position generically (not a specific person), do not
       merge it with any person entity.
- **Part-whole relationships:** Keep as SEPARATE entities
      if they are genuinely distinct things that the text
      treats differently (e.g., "U.S. Senate" and "U.S.
      Congress" are separate if the text discusses them
      independently). Merge only if the text uses them
      interchangeably for the same referent.
- **Never create two entries for the same real-world
      entity.**

---Instructions---

**Step 1: Entity Identification**
Read through ALL the QA pairs. Identify every distinct
      entity that meets the inclusion criteria above.
      Apply the exclusion rules strictly.

**Step 2: Cross-QA Synthesis**
If the same entity appears across multiple QA pairs,
      combine all information into a single entry. If
      different surface forms refer to the same entity,
      canonicalize per the rules above.

**Step 3: Entity Details**
For each entity, extract:
- `entity_id`: A short stable identifier (EN1, EN2, EN3,
      ...).
- `entity_name`: The canonical name. Use title case for
      proper nouns.
- `entity_type`: One of the defined entity types above.
- `aliases`: A list of all surface-form variants found in
      the QA pairs (including the canonical name).
- `description`: 1-3 sentences (up to 80 words). Summarize
       this entity's role, key attributes, and
      relationships as described in the QA pairs. Write in
       third person using explicit entity names (never
      pronouns). Preserve important facts, names, dates,
      and relationships. Every word must carry information
       -- no filler.

**Step 4: Objectivity & Language**
- Write all descriptions in the third person. Use explicit
       entity names, never pronouns.
- Output must be in {language}. Retain proper nouns in
      their original language.

---Pre-Output Validation Checklist---

Before returning your output, verify all of the following:
- Every `entity_type` is one of the defined types above.
- No two entries refer to the same real-world entity (
      deduplication check).
- Every alias in the `aliases` list actually appears in
      the QA pairs.
- No entry is a generic noun phrase, bare metric, temporal
       expression, or passing descriptor.
- All descriptions use entity names, not pronouns.

---Worked Example---

**Example QA Pairs:**
[
  {"question": "What did Arcadia Corp announce in March
      2023?", "answer": "Arcadia Corp announced the launch
       of the Greenfield Initiative in March 2023."},
  {"question": "Who led the Greenfield Initiative?", "
      answer": "Sarah Chen, VP of Sustainability at
      Arcadia Corp, led the Greenfield Initiative."},
  {"question": "Why did Arcadia Corp launch the Greenfield
       Initiative?", "answer": "Arcadia Corp launched the
      Greenfield Initiative in response to new
      environmental regulations imposed by the City of
      Millbrook in January 2023."},
  {"question": "What happened after the Greenfield
      Initiative launched?", "answer": "After the
      Greenfield Initiative launched, Arcadia Corp reduced
       carbon emissions by 18 percent within six months."},

  {"question": "What framework did the Greenfield
      Initiative use?", "answer": "The Greenfield
      Initiative used the ISO 14001 Environmental
      Management framework to structure Arcadia Corp's
      sustainability efforts."}
]

**Example Output:**
[
  {
    "entity_id": "EN1",
    "entity_name": "Arcadia Corp",
    "entity_type": "Organization",
    "aliases": ["Arcadia Corp"],
    "description": "Arcadia Corp announced and executed
      the Greenfield Initiative in March 2023 in response
      to environmental regulations from the City of
      Millbrook. Under the Greenfield Initiative, Arcadia
      Corp achieved an 18 percent reduction in carbon
      emissions within six months."
  },
  {
    "entity_id": "EN2",
    "entity_name": "Greenfield Initiative",
    "entity_type": "Concept",
    "aliases": ["Greenfield Initiative", "the initiative
      "],
    "description": "The Greenfield Initiative is a
      sustainability program launched by Arcadia Corp in
      March 2023, led by Sarah Chen. The Greenfield
      Initiative was structured around the ISO 14001
      Environmental Management framework and resulted in
      an 18 percent reduction in carbon emissions for
      Arcadia Corp."
  },
  {
    "entity_id": "EN3",
    "entity_name": "Sarah Chen",
    "entity_type": "Person",
    "aliases": ["Sarah Chen"],
    "description": "Sarah Chen served as VP of
      Sustainability at Arcadia Corp and led the
      Greenfield Initiative, which achieved an 18 percent
      carbon emission reduction."
  },
  {
    "entity_id": "EN4",
    "entity_name": "City of Millbrook",
    "entity_type": "Location",
    "aliases": ["City of Millbrook", "Millbrook"],
    "description": "The City of Millbrook imposed new
      environmental regulations in January 2023 that
      prompted Arcadia Corp to launch the Greenfield
      Initiative."
  },
  {
```

```
    "entity_id": "EN5",
    "entity_name": "ISO 14001 Environmental Management",
    "entity_type": "Method",
    "aliases": ["ISO 14001 Environmental Management", "ISO
       14001"],
    "description": "ISO 14001 Environmental Management is
      the framework used to structure the Greenfield
      Initiative's sustainability efforts at Arcadia Corp
      ."
  }
]

Note how:
- "the initiative" is captured as an alias for Greenfield
      Initiative, not as a separate entity.
- "18 percent" and "six months" are NOT extracted -- they
      are bare metrics/temporal expressions, not entities.
- Descriptions are concise but preserve key facts,
      relationships, and context.

---Input QA Pairs---
{qa_pairs_json}

---Output Format---
Return a JSON list:
[
  {
    "entity_id": "...",
    "entity_name": "...",
    "entity_type": "...",
    "aliases": ["...", "..."],
    "description": "..."
  },
  ...
]
```

Event-extraction prompt (Step 3)

```
---Role---
You are a Narrative Event Specialist responsible for
      extracting events and the causal, temporal, and
      consequential relationships between them from a set
      of document-grounded Question-Answer pairs.

---Context---
The QA pairs below were generated from a single chunk of a
       source document using 5W1H-guided analysis. Each QA
       pair is self-contained with pronouns resolved to
      explicit entity names.

You are also provided with a list of entities already
      extracted from these same QA pairs. **You MUST use
      the canonical entity names from this list when
      referring to participants/entities in event
      descriptions.** This ensures consistent linking
      between entities and events downstream.

---Event Definition & Boundary Rules---

An event is a discrete action, occurrence, state change,
      or happening that can be placed on a timeline.

**Inclusion criteria** -- extract it if it has at least
      one of:
- A clear agent performing an action.
- An observable state change (before -> after).
- A temporal anchor (a point or interval in time).

**Exclusion criteria** -- do NOT extract:
- Static descriptions or attributes of entities (e.g., "
      Company X is headquartered in Berlin").
- Definitions, classifications, or taxonomic statements.
- Ongoing background conditions with no identifiable start
       or change point.

**Litmus test:** Can you place it on a timeline? If not,
      it is not an event.

---Granularity Guidance---
Extract events at the granularity present in the QA pairs.
       Do not summarize multiple distinct events into one
      mega-event, and do not split a single described
      action into sub-steps unless the QA pairs treat them
       as separate occurrences.

---Cross-QA Synthesis & Deduplication Rules---
A single event may be referenced across multiple QA pairs
      (e.g., one covers *who*, another *why*). You must
      synthesize these into a unified event entry.

- **Merge condition:** Two QA references describe the SAME
       event if they share the same core action, the same
      participants, and the same approximate time/place.
      Merge them into one entry. Use the most specific
      event_name and union all details into the
      description.
- **Split condition:** Two references describe DIFFERENT
      events if the action, outcome, or temporal position
      differs -- even if participants overlap. Keep them
      as separate entries.
- **Never create duplicate events.** If in doubt, merge.

---Event Type Definitions---
Classify each event using exactly one of the following
      types. Use "Other" only if none fit.

- Action: A deliberate, purposeful act by an agent that
      does not fit a more specific type below. Use as a
      fallback only when no other event type applies.
- Transaction: An exchange of resources, value, or
      ownership between parties -- something changes hands
       (e.g., purchases, acquisitions, grants, donations).
- Communication: A deliberate transmission of information
      to an audience (e.g., announcements, publications,
      testimonies, press releases). Emphasis is on *
      sharing* information, not *finding* it.
- Movement: A physical relocation of people, forces, goods
      , or materials from one place to another (e.g.,
      deployments, evacuations, migrations, shipments).
- Creation: The bringing into existence of something new
      that did not previously exist (e.g., founding,
      building, developing, inventing). If the thing
      already existed but was unknown, use Discovery
      instead.
- Destruction: The permanent elimination, dismantling, or
      dissolution of something (e.g., demolitions,
      decommissionings, dissolutions). If the thing still
      exists in altered form, use Change instead.
- Decision: A deliberate choice, ruling, or judgment that
      selects a course of action from alternatives (e.g.,
      court rulings, votes, approvals, strategic choices).
       Emphasis is on the *act of choosing*, not the
      subsequent execution.
- Discovery: The finding or first identification of
      something that already existed but was previously
      unknown (e.g., scientific discoveries, detection of
      threats, uncovering evidence). If the thing was *
      made new*, use Creation instead.
- Conflict: A confrontation, dispute, or antagonistic
      interaction between opposing parties (e.g., wars,
      lawsuits, protests, strikes, sanctions).
- Agreement: A convergence of parties on shared terms or
      mutual commitments (e.g., treaties, partnerships,
      settlements, cease-fires). If the emphasis is on the
       *exchange of value* rather than *reaching terms*,
      use Transaction instead.
- Change: A transformation or alteration to something
      already existing (e.g., restructurings, reforms,
      promotions, state transitions). If something *new*
      comes into being, use Creation; if something is *
      permanently eliminated*, use Destruction.
- Observation: A passive recording, measurement, or
      witnessing of a phenomenon without causing it (e.g.,
       surveys, audits, inspections, measurements). If the
       observed thing is *new or previously unknown*, use
      Discovery instead.


---Relationship Type Definitions---
Use exactly one of the following relationship types.
      Choose the most precise fit.

- `causes`: A is a necessary or sufficient condition for B.
       Without A, B would not occur.
- `triggers`: A is the proximate catalyst that initiates B
       at a specific moment. B might have occurred
      eventually, but A determined the timing.
- `leads_to`: A contributes to B but is not solely
      responsible. Other factors also play a role.
```

```
- `enables`: A creates the conditions that make B possible, but does not initiate B.
- `prevents`: A stops B from occurring or continuing.
- `follows`: B occurs after A with an implied but not explicitly stated causal link. Use when temporal sequence is clear but causality is uncertain.
- `in_response_to`: B is a deliberate reaction to A by an agent.
- `escalates`: B is an intensification or expansion of the same dynamic as A.
- `resolves`: B brings the situation created by A to a conclusion.

**Direction convention:** Relationships always point forward in narrative time -- `source_event` is the earlier or causal event, `target_event` is the later or consequent event.

---Instructions---

**Step 1: Event Extraction**

Read through ALL the QA pairs and extract clearly defined, meaningful events that meet the inclusion criteria above. Apply the exclusion rules strictly.

For each event, provide:
- `event_id`: A short stable identifier (EV1, EV2, EV3, ...).
- `event_name`: A concise, title-case name.
- `event_type`: One of the defined event types above.
- `description`: A comprehensive description synthesized from all relevant QA pairs. **Name all participants using exact canonical names from the entity list.** Include what happened, who was involved, when, where, why, and how -- as available. Write in the third person using entity names, never pronouns.
- `participant_entities`: A list of canonical entity names involved. Every name MUST match an `entity_name` from the provided entity list exactly.
- `evidence_strength`: "direct" if the event is explicitly stated in a QA pair; "inferred" if synthesized from indirect references across multiple pairs.

**Step 2: Event Relationship Extraction**

Identify causal, temporal, or consequential relationships between the events extracted in Step 1.

- Only extract relationships that reflect genuine narrative causality or temporal sequence -- not mere co-occurrence or topical similarity.
- The *why* and *how* QA pairs often make these links explicit; leverage them.

For each relationship, provide:
- `source_event`: The `event_id` of the earlier or causal event.
- `target_event`: The `event_id` of the later or consequent event.
- `relationship_type`: One of the defined relationship types above.
- `description`: A concise explanation of the causal or temporal link. Use entity names, never pronouns.

**Step 3: Objectivity & Language**
- Write in the third person. Use explicit entity names, never pronouns.
- Output must be in {language}. Retain proper nouns in their original language.

---Pre-Output Validation Checklist---
Before returning your output, verify all of the following:
- Every name in `participant_entities` appears verbatim in the provided entity list.
- Every `source_event` and `target_event` matches an `event_id` from your extracted events.
- No two event entries describe the same occurrence (deduplication check).
- No event entry is actually a static fact, attribute, or definition.
- Every relationship reflects a genuine causal or temporal link, not mere topical similarity.
- Every `event_type` is one of the defined values above.
- Every `relationship_type` is one of the defined values above.
- All descriptions use entity names, not pronouns.

---Worked Example---

**Example Entity List:**
[
  {"entity_id": "EN1", "entity_name": "Arcadia Corp", "entity_type": "Organization", "aliases": ["Arcadia Corp"]},
  {"entity_id": "EN2", "entity_name": "Greenfield Initiative", "entity_type": "Concept", "aliases": ["Greenfield Initiative", "the initiative"]},
  {"entity_id": "EN3", "entity_name": "Sarah Chen", "entity_type": "Person", "aliases": ["Sarah Chen"]},
  {"entity_id": "EN4", "entity_name": "City of Millbrook", "entity_type": "Location", "aliases": ["City of Millbrook", "Millbrook"]},
  {"entity_id": "EN5", "entity_name": "ISO 14001 Environmental Management", "entity_type": "Method", "aliases": ["ISO 14001 Environmental Management", "ISO 14001"]}
]

**Example QA Pairs:**
[
  {"question": "What did Arcadia Corp announce in March 2023?", "answer": "Arcadia Corp announced the launch of the Greenfield Initiative in March 2023."},
  {"question": "Who led the Greenfield Initiative?", "answer": "Sarah Chen, VP of Sustainability at Arcadia Corp, led the Greenfield Initiative."},
  {"question": "Why did Arcadia Corp launch the Greenfield Initiative?", "answer": "Arcadia Corp launched the Greenfield Initiative in response to new environmental regulations imposed by the City of Millbrook in January 2023."},
  {"question": "What happened after the Greenfield Initiative launched?", "answer": "After the Greenfield Initiative launched, Arcadia Corp reduced carbon emissions by 18 percent within six months."},

  {"question": "What framework did the Greenfield Initiative use?", "answer": "The Greenfield Initiative used the ISO 14001 Environmental Management framework to structure Arcadia Corp's sustainability efforts."}
]

**Example Output:**
{
  "events": [
    {
      "event_id": "EV1",
      "event_name": "Millbrook Environmental Regulations Imposed",
      "event_type": "Change",
      "description": "The City of Millbrook imposed new environmental regulations in January 2023. These regulations prompted Arcadia Corp to take sustainability action.",
      "participant_entities": ["EN4"],
      "evidence_strength": "direct"
    },
    {
      "event_id": "EV2",
      "event_name": "Greenfield Initiative Launch",
      "event_type": "Creation",
      "description": "Arcadia Corp announced and launched the Greenfield Initiative in March 2023 in response to environmental regulations imposed by the City of Millbrook. Sarah Chen, VP of Sustainability at Arcadia Corp, led the Greenfield Initiative. The Greenfield Initiative adopted the ISO 14001 Environmental Management framework to structure Arcadia Corp's sustainability efforts.",
      "participant_entities": ["EN1", "EN3", "EN2", "EN5"],
      "evidence_strength": "direct"
    },
    {
      "event_id": "EV3",
      "event_name": "Arcadia Corp Emission Reduction",
      "event_type": "Change",
      "description": "Arcadia Corp reduced carbon emissions by 18 percent within six months following the launch of the Greenfield Initiative.",
```

```
      "participant_entities": ["EN1", "EN2"],
      "evidence_strength": "direct"
    }
  ],
  "event_relationships": [
    {
      "source_event": "EV1",
      "target_event": "EV2",
      "relationship_type": "triggers",
      "description": "The environmental regulations
      imposed by the City of Millbrook were the proximate
      catalyst for Arcadia Corp launching the Greenfield
      Initiative."
    },
    {
      "source_event": "EV2",
      "target_event": "EV3",
      "relationship_type": "leads_to",
      "description": "The launch of the Greenfield
      Initiative contributed to Arcadia Corp achieving an
      18 percent reduction in carbon emissions."
    }
  ]
}

Note how:
- QA pairs about "who launched it", "why it was launched",
       and "what framework it used" were merged into a
      single event (EV2), not split into separate entries
      -- they describe the same occurrence from different
      angles.
- EV1 (regulations) and EV3 (emission reduction) are kept
      separate from EV2 because they have different
      actions and temporal positions.
- Every name in `participant_entities` matches an `
      entity_id` from the entity list exactly.
- Relationships use `event_id` values, not event names.
- All descriptions use entity names, never pronouns.

---Extracted Entities (use these canonical names)---
{entities_json}

---Input QA Pairs---
{qa_pairs_json}

---Output Format---
Return a JSON object with exactly this structure:
{
  "events": [
    {
      "event_id": "...",
      "event_name": "...",
      "event_type": "...",
      "description": "...",
      "participant_entities": ["ENX", "ENY", ...],
      "evidence_strength": "direct | inferred"
    }
  ],
  "event_relationships": [
    {
      "source_event": "...",
      "target_event": "...",
      "relationship_type": "...",
      "description": "..."
    }
  ]
}
```

E.3 Answer and judge prompts

The shared answer prompt (adapted from LightRAG's default, with output-format constraints and the `References` section removed and external knowledge forbidden) and the judge prompt. The independent Qwen3-32B and DeepSeek-V3.2 re-scorings (Table 5) used this judge prompt unchanged, through AWS Bedrock (`qwen.qwen3-32b-v1:0`, `deepseek.v3.2`) at temperature 0 with reasoning modes off; every one of the 8,300 calls to each judge returned valid JSON, and the original scores were never overwritten.

Response (answer) prompt

```
---Role---
You are an AI assistant that answers questions using ONLY
      the information provided in the Context below. The
      context is the output of a knowledge-graph /
      retrieval pipeline run over a book --- it may
      contain entities, events, relationships, chunk
      excerpts, or other structured notes extracted from
      the source text.

---Goal---
Generate an accurate answer to the user query, grounded
      strictly in the provided Context.

---Instructions---
- Use ONLY facts explicitly stated in the Context. Do NOT
      invent, assume, or introduce external knowledge.
- Your own knowledge may only be used to form fluent
      sentences and connect ideas, never to add new
      information.
- If the Context does not contain enough information to
      answer, reply exactly with: "{fail_response}"
- Keep the response concise and focused on the query.

---Context---
{context}

---Query---
{question}
```

Scoring prompt (system)

```
You are tasked with evaluating a predicted answer against
      a reference answer
for a given question.

Scoring Rubric (1-10 scale):
10 - Matches ground truth exactly or with a faithful
      paraphrase; all key points present.
 7 - Mostly correct; minor omissions or wording
      differences that do not change meaning.
 5 - Partially correct; captures some key points but has
      major missing information or inaccuracies.
 3 - Mostly incorrect; only small or superficial overlap
      with the ground truth.
 1 - Off-topic, completely wrong, or hallucinated content.

Output - respond ONLY with valid JSON, no markdown fences:
{"score": <integer 1-10>, "reason": "<short rationale>"}
```

Scoring prompt (user)

```
Question: {question}
Predicted Answer: {predicted_answer}
Ground Truth: {ground_truth}
```

F Worked Example: Retrieved Context

For *"Who actually gave Hagrid the dragon egg in the Hog's Head pub, and why was it significant?"* (reference: *Professor Quirrell, disguised, gave Hagrid the egg; significant because Quirrell used Hagrid's eagerness for a dragon to learn how to get past Fluffy*), we reproduce what each graph method retrieved and answered, so the subgraphs of §5 can be read against the text the answer model received. `eSoKG`'s context is chunk-free structure

only; E^{2}RAG's and LightRAG's contexts additionally include verbatim source chunks and unmerged (E^{2}RAG) or name-merged (LightRAG) entities.

F.1 eSoKG (scored 9)

Judge rationale: *"correctly identifies Quirrell… and correctly explains the significance… goes beyond the ground truth but doesn't contradict it."*

eSoKG — retrieved context (excerpt)

```
This excerpt shows five entities and five events in full,
      with the remainder listed by name, and the first
      three relationships in full, with the rest as edges
      only. The retrieved context passed to the model
      contained every item in full.

## Entities (5 of 63 shown in full)

E3  Hagrid (Person): Rubeus Hagrid is the Keeper of Keys,
      Grounds, and gamekeeper at Hogwarts, a half-giant
      expelled in third year but kept on by Dumbledore. He
       owns Fang and Fluffy (lent to Dumbledore), secretly
       hatched dragon Norbert, gave Harry a flute used to
      put Fluffy to sleep, and inadvertently revealed
      Fluffy's weakness to Quirrell.

E6  The Cloaked Stranger (Person): The cloaked stranger is
       an unidentified individual who met Hagrid at the
      Hog's Head, kept his hood up so his face was never
      seen, and deliberately bought Hagrid drinks while
      steering conversation toward Fluffy. The stranger
      offered Hagrid a dragon egg in a card game after
      extracting the critical information that music could
       put Fluffy to sleep.

E7  Hog's Head (Location): The Hog's Head is a pub in the
      village near Hogwarts, described as a place that
      attracts 'funny folk.' It is where Hagrid met the
      cloaked stranger, played cards, won Norbert the
      dragon, and was manipulated into revealing Fluffy's
      weakness.

E18 Professor Quirrell (Person): Professor Quirinus
      Quirrell is the stammering, turban-wearing Defense
      Against the Dark Arts teacher at Hogwarts who became
       a servant of Voldemort. He let the Halloween troll
      into Hogwarts, cursed Harry's broom, and attempted
      to steal the Philosopher's Stone.

E19 Fluffy (Creature): Fluffy is Hagrid's giant three-
      headed dog, lent to Dumbledore to guard the trapdoor
       leading to the Sorcerer's Stone. Fluffy can be
      soothed to sleep by music, a weakness Hagrid
      accidentally revealed to Quirrell at the Hog's Head
      in exchange for a dragon egg.

Remaining entities (name only): E1 Hogwarts Library, E2
      Hogwarts, E4 Dragon Breeding for Pleasure and Profit,
       E5 Norbert, E8 Harry Potter, E9 Voldemort, E10
      Professor Snape, E11 Ron Weasley, E12 Hermione
      Granger, E13 Hagrid's Hut, E14 Warlocks' Convention
      of 1709, E15 Muggles, E16 Hedwig, E17 Herbology, E20
       Hogwarts Hospital Wing, E21 Great Hall, E22 The
      Apothecary, E23 Eeylops Owl Emporium, E24 James
      Potter, E25 Photo Album, E26 Hagrid, E27 Sorcerer's
      Stone, E28 Dumbledore, E29 Chocolate Frogs, E30
      Trapdoor, E31 Quidditch, E32 Albus Dumbledore, E33
      Neville Longbottom, E34 George Weasley, E35 Lee
      Jordan, E36 Madam Pomfrey, E37 Transfiguration, E38
      Nicolas Flamel, E39 Dragon Species of Great Britain
      and Ireland, E40 From Egg to Inferno A Dragon Keeper
      's Guide, E41 Draco Malfoy, E42 Gryffindor, E43
      Dungeons, E44 Crabbe, E45 Potions, E46 Goyle, E47
      Slytherin, E48 Professor Flitwick, E49 Professor
      McGonagall, E50 Knut, E51 Galleon, E52 Beetle Eyes,
      E53 Unicorn Horn, E54 Ollivanders, E55 Mr.
      Ollivander, E56 Ministry of Magic, E57 Mirror of
      Erised, E58 History of Magic, E59 Anti-Cheating
      Spell, E60 Forgetfulness Potion, E61 1637 Werewolf
      Code of Conduct, E62 Elfric the Eager, E63 Professor
       Binns.

## Events (5 of 30 shown in full)

V4  Hagrid Meets Cloaked Stranger at Hog's Head (
      Communication; participants E2, E3, E6, E7): Hagrid
      met The Cloaked Stranger at the Hog's Head. The
      Cloaked Stranger kept his hood up throughout the
      encounter, so Hagrid never saw his face. He asked
      Hagrid about his work, learning that Hagrid's
      greatest desire was to own a dragon, kept buying
      Hagrid drinks to impair his judgment, and offered to
       play cards for a dragon egg on the condition that
      Hagrid could prove he could handle the dragon.

V1  Hagrid Wins Dragon Egg in Card Game (Transaction;
      participants E1, E2, E3, E4, E5, E6, E7): Hagrid won
       a huge black Norwegian Ridgeback dragon egg in a
      card game with the cloaked stranger at the Hog's
      Head, who had first established that Hagrid could
      handle a dragon. The egg later hatched into the
      dragon Norbert.

V11 Hagrid Reveals Fluffy's Weakness to Cloaked Stranger (
      Communication; participants E2, E3, E6, E7, E30, E19,
       E18, E27): During the card game at the Hog's Head,
      Hagrid accidentally revealed to Professor Quirrell,
      disguised as a cloaked stranger, that Fluffy could
      be put to sleep by playing music. Quirrell had kept
      buying Hagrid drinks to extract this information,
      offering a dragon egg in exchange.

V2  Harry Potter Realizes Hagrid's Dragon Egg Was
      Suspicious (Discovery; participants E2, E3, E6, E8,
      E9, E10, E11, E12): While by the lake during exams,
      Harry realized the encounter at the Hog's Head was a
       deliberate setup: a stranger had conveniently
      offered a dragon egg to Hagrid, who wanted a dragon
      above all else. Harry concluded the stranger had
      targeted Hagrid to extract information about the
      Stone's defenses.

V10 Harry, Ron, and Hermione Visit Hagrid (Communication;
      participants E2, E3, E6, E7, E19, E5, E8, E11, E12):
       The trio visited Hagrid and questioned him about
      what the cloaked stranger looked like on the night
      he won Norbert. Hagrid revealed that he had told the
       stranger Fluffy could be put to sleep with music,
      then looked horrified on realizing he had let the
      weakness slip again.

Remaining events (name only): V3 Dragon Egg Discovered in
      Hagrid's Fire, V5 Norwegian Ridgeback Egg Hatching,
      V6 Hagrid Visits Harry in the Hospital Wing, V7
      Hedwig Delivers Hagrid's Note to Harry, V8 Hagrid
      Buys Harry a Snowy Owl, V9 Hagrid Gives Harry the
      Photo Album, V12 Trio Relax by Lake After Exams, V13
       Hagrid's Suspicious Visit to Library Dragons
      Section, V14 Hagrid Instructs Trio to Visit His Hut,
       V15 Hermione Creates Study Schedules, V16 Malfoy
      Spies on Dragon Hatching, V17 Norbert Grows Rapidly
      After Hatching, V18 Malfoy Overhears Hatching
      Conversation, V19 Trio Rushes to Hagrid's Hut, V20
      Harry Recovers in Hospital Wing, V21 Harry Shouts
      Voldemort's Name at Hagrid, V22 First Potions Lesson,
       V23 Harry and Ron Navigate to Great Hall, V24 Harry
       and Hagrid Visit The Apothecary, V25 Harry and
      Hagrid Arrive at Ollivanders, V26 Hagrid Collects
      Photographs, V27 Trio Attempt to See Dumbledore, V28
       Dumbledore Receives Urgent Owl, V29 Harry Retrieves
       Stone from Mirror of Erised, V30 Year-End
      Examinations.

## Event Relationships (first 3 shown in full)

V4 (Hagrid Meets Cloaked Stranger)  --enables-->  V11 (
      Hagrid Reveals Fluffy's Weakness): The stranger's
      strategy of buying Hagrid drinks and drawing him
      into conversation created the conditions under which
       Hagrid revealed Fluffy's weakness.

V11 (Hagrid Reveals Fluffy's Weakness)  --follows-->  V1 (
      Hagrid Wins Dragon Egg in Card Game): After the
      stranger extracted the information about Fluffy, the
```

```
      card game concluded with Hagrid winning the dragon
      egg.

V1 (Hagrid Wins Dragon Egg in Card Game)  --triggers-->
      V2 (Harry Realizes Egg Was Suspicious): Harry's
      reflection that Hagrid had won the egg from a
      suspicious stranger triggered his realization that
      the encounter was a deliberate setup.

Remaining relationships (edges only):
V2  --causes-->     V10
V10 --causes-->     V27
V11 --leads_to-->   V6
V11 --leads_to-->   V28
V13 --enables-->    V3
V14 --leads_to-->   V3
V3  --leads_to-->   V5
V5  --triggers-->   V16
V29 --leads_to-->   V20
```

eSoKG — generated answer

```
## The Cloaked Stranger Who Gave Hagrid the Dragon Egg

Based on the context, the cloaked stranger who gave Hagrid
      the dragon egg in the Hog's Head was actually **
      Professor Quirrell**, Voldemort's servant, though he
      was disguised and kept his hood up throughout the
      encounter so Hagrid never saw his face.

### How It Happened

The encounter was a **deliberate and calculated setup**.
      The cloaked stranger:
- Kept buying Hagrid drinks to impair his judgment
- Steered the conversation toward Hagrid's work at
      Hogwarts
- Learned that Hagrid's greatest desire was to own a
      dragon
- Offered to play cards for a dragon egg, on the condition
      that Hagrid could prove he could handle a dragon

### Why It Was Significant

The dragon egg was essentially a **bribe to extract
      critical information**. In exchange for the egg, the
      stranger successfully manipulated Hagrid into
      revealing that **Fluffy -- the three-headed dog
      guarding the trapdoor protecting the Sorcerer's
      Stone -- could be put to sleep with music**.

Harry Potter later realized the entire encounter was a
      deliberate setup, concluding that the stranger had
      specifically targeted Hagrid because he "desperately
      wanted a dragon above all else," making him
      vulnerable to manipulation. This information
      ultimately enabled Quirrell to bypass Fluffy and
      attempt to steal the Sorcerer's Stone.

Hagrid later felt **deeply guilty**, acknowledging he
      should never have revealed Fluffy's weakness.
```

F.2 E^{2}RAG (scored 6)

Judge rationale: *"correctly identifies the significance... however, it fails to correctly identify the person as Professor Quirrell (disguised), instead suggesting 'either Snape or Voldemort'..."* The unmerged context surfaces the dragon's aftermath rather than the pub exchange.

E2RAG — generated answer

```
## Who Gave Hagrid the Dragon Egg

Based on the context provided, the identity of the person
      who gave Hagrid the dragon egg was never
      definitively revealed to Hagrid himself -- the
      stranger **kept his cloak on and his hood up the
      entire time**, so Hagrid never saw his face. As
      Hagrid explained: *"he wouldn' take his cloak off"*
      and *"I never saw his face, he kept his hood up."*

However, Harry suspected -- and the context strongly
      implies -- that this was **either Snape or Voldemort
      ** in disguise. Harry concluded: *"it was either
      Snape or Voldemort under that cloak."*

## Why It Was Significant

The dragon egg was significant for several reasons:

1. **It was used as a manipulation tool.** The cloaked
      stranger deliberately exploited Hagrid's greatest
      wish -- owning a dragon -- to gain his trust. The
      stranger bought Hagrid drinks, got him drunk, and
      used the egg as a lure in a card game.

2. **It extracted critical information.** In exchange for
      the egg (and under the influence of alcohol), Hagrid
       unwittingly revealed that **playing music would put
       Fluffy to sleep** -- the one piece of information
      the stranger needed to get past the three-headed dog
       guarding the Sorcerer's Stone.

3. **It was illegal.** Possessing dragon eggs violated
      wizard law, making the stranger's convenient
      possession of one deeply suspicious.

As Hagrid himself later admitted with great remorse: *"All
       fer a dragon egg! I'll never drink again!"*
```

F.3 LightRAG (scored 10)

Judge rationale: *"correctly identifies Professor Quirrell... accurately explains the significance... all key points present."* Its name-merged *Fluffy* description explicitly names Quirrell, which the answer draws on.

LightRAG — generated answer

```
Based on the context provided, the mysterious person who
      gave Hagrid the dragon egg was a **cloaked stranger
      ** who kept his hood up throughout their meeting at
      the Hog's Head pub. The stranger played cards with
      Hagrid, plying him with drinks, and Hagrid won the
      Norwegian Ridgeback egg in the card game.

The significance of this encounter was profound and
      deliberately orchestrated. The cloaked stranger --
      **later identified as Professor Quirrell** (acting
      under Voldemort's influence) -- used the dragon egg
      as a lure to extract critical information from
      Hagrid. During their conversation, while keeping
      Hagrid drunk, Quirrell manipulated Hagrid into
      revealing the secret of **how to get past Fluffy** --
       namely, that playing music would lull the three-
      headed dog to sleep.

As the context states, the dragon egg was "the item the
      cloaked stranger used to lure Hagrid into a card
      game, ultimately resulting in Hagrid revealing the
      secret of how to calm Fluffy." Harry also grew
      suspicious because a stranger "conveniently had [a
      dragon egg] in his pocket," recognizing it as too
      convenient to be coincidental.

This inadvertent disclosure by Hagrid was devastating, as
      it compromised the security protecting the Sorcerer'
      s Stone and allowed Quirrell to bypass Fluffy.
      Hagrid later felt **great guilt** for his role in
      enabling Quirrell's attempt to steal the Stone.
```

HTTP for Meaning: Towards Linguistically Complete Knowledge Representations with KNOWLEDGE STATE PROTOCOL

Ian Yuen
Independent Researcher
jardinianr@gmail.com

Abstract

The choice of knowledge representation (KR) determines what a knowledge base (KB) can store, and how it can be retrieved, and reasoned over. Recent Knowledge-Graph (KG) Retrieval-Augmented Generation (RAG) systems show that *explicit symbolic structure* can complement the implicit knowledge encoded in LLMs. However, KGs capture only part of a text's linguistic meaning and often leave **conceptual** and **temporal** context implicit, leading to recurrent reasoning failures. Existing approaches address these gaps through task-specific extensions, but do not provide a unified framework for coordinating conceptual, instance-level, and temporal structure. We envision the KNOWLEDGE STATE PROTOCOL (KSP), which unifies canonical KRs to make the known linguistic dimensions of a text explicit. By standardizing the interoperation between KRs, KSP provides a common, extensible, and open foundation for *structured* agent memory and neuro-symbolic training across modalities and languages.

1 Introduction

Natural language encodes multiple layers of meaning, whereas any single knowledge representation (KR) externalizes only part of that structure (Table 1). Ontologies make domain concepts, taxonomies, and logical constraints explicit; KGs represent facts stated in source text; and Abstract Meaning Representation (AMR) models sentence-level events and the semantic relations between participants (Banarescu et al., 2013). Each representation has independently supported question answering (QA) systems (Lopez et al., 2005; Berant et al., 2013; Kapanipathi et al., 2021).

However, no single KR captures all *linguistic dimensions* (Table 1), as in the case of KG-based RAG. Although it improves on text-based RAG by representing entities and factual relations symbolically (Edge et al., 2024), it still leaves two recurring gaps: **conceptual context** (Zhong et al., 2024) and **temporal and event context** (Cai et al., 2023). KGs encode object-relation-object triples which can capture the explicitly stated facts in text, but not always the implied conceptual (e.g., "X is a type of *Y*") and temporal context (e.g., "A occurred before B") with which humans interpret text.

The limitations of relying on partial or implicit representations are especially consequential in LLM-based systems. Although LLMs achieve strong performance on QA benchmarks (OpenAI, 2026), their internal knowledge is difficult to inspect, attribute, constrain, or update (Lewis et al., 2020). Moreover, in enterprise and specialized settings, the required knowledge may be private, domain-specific, or continuously changing and therefore unavailable during model training (Kandpal et al., 2023; Mallen et al., 2023). Addressing the **conceptual** and **temporal context** gaps in KG-RAG could therefore improve QA performance in LLM systems.

Recent systems address these omissions by augmenting KGs with specialized structures for additional context. Temporal QA methods add timestamps, intervals, dynamic updates, or event-centric relations to preserve how facts evolve (Li et al., 2025; Zhang et al., 2026; Sun et al., 2025; Han et al., 2025; Wang et al., 2026); Conceptual QA methods augment graph-based retrieval and reasoning with ontology-derived concepts, relations, and schemas (Liu et al., 2025; Sharma et al., 2025). Other systems go further by combining entity, concept, event, and temporal structures within a bespoke domain-specific graph or meaning representation (de Martim, 2025; Bai et al., 2026).

Although effective, these approaches add missing capabilities separately or re-encode conceptual, factual, and event-temporal distinctions within a single custom graph schema, rather than coordinating the established KRs that already formalize them. Ontologies can contextualize data as

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 128–138
October 28, 2026 ©2026 Association for Computational Linguistics

Linguistic capacity in natural language	KG representation	AMR representation	Ontology representation	Best-fit KR
Factual reference: language refers to entities, relations, and states of affairs.	**Strong**: represents entity–relation facts.	**Moderate**: represents sentence-level concepts and roles.	**Moderate**: defines conceptual types and relations.	**KG** is strongest for durable factual memory, but not full sentence meaning or abstraction.
Compositional event meaning: predicates, arguments, modifiers, and scope combine into structured meanings.	**Weak to moderate:** triples often flatten nested or scoped meaning.	**Strong:** explicitly represents predicate–argument and event structure.	**Moderate:** provides reusable conceptual primitives, but not full sentential structure.	**AMR** captures local semantic structure that KG and ontology often miss.
Conceptual abstraction: language expresses classes, roles, norms, constraints, and generalizations.	**Moderate:** can attach types, but schemas are often incomplete.	**Weak to moderate:** can express abstract predicates, but not a domain theory.	**Strong**: represents classes, properties, constraints, inheritance, and equivalence.	**Ontology** supplies the conceptual layer missing from instance-oriented graphs.
Displacement and temporality: language refers to past, future, absent, hypothetical, or changing states.	**Weak** unless time and state are explicitly modeled.	**Moderate to strong**: captures event-level temporal cues, but needs coordination.	**Moderate**: can define states, events, intervals, and temporal relations.	**AMR**s structured as a ledger can provide both event extraction and explicit temporal modeling.
Modality, uncertainty, and falsehood: language expresses beliefs, claims, hypotheticals, negation, fiction, and error.	**Weak**: plain triples risk converting claims into facts.	**Moderate to strong**: naturally represents negation, attribution, and selected modal meanings.	**Strong if designed**: can model Claim, Belief, Evidence, Source, and Hypothesis.	Requires **AMR**-style cues plus **ontology**-style claim modeling.
Reflexiveness and provenance: language can talk about statements, sources, and representations.	**Moderate**: possible through reification, named graphs, or metadata.	**Moderate**: can represent reporting and saying events.	**Strong**: can define source, evidence, statement, and provenance classes.	**Ontology** enables provenance as a first-class concept rather than metadata.
Productivity: language can express novel meanings beyond a fixed inventory.	**Strong**: easily adds new entities and relations, though semantics may be underspecified.	**Moderate**: composes novel event structures but bounded by frames and relation inventory.	**Strong but costly**: adds precise concepts and relations through governed schema evolution.	**KGs** maximize extensibility, while **ontologies** provide semantic control.

Table 1: For each *linguistic dimension* (Hockett, 1960), a different KR is best suited to represent it.

instances of domain concepts and support abstract multi-hop reasoning through logical hierarchies and constraints. AMR graphs can isolate relationships between entities for each sentence event, while an AMR event ledger can model how entity relationships change over time. What is missing is a shared framework that coordinates these complementary structures through a common integration protocol.

2 Knowledge State Protocol (KSP)

We envision KSP to be a common protocol that captures the known *linguistic dimensions* of a text by coordinating explicit symbolic KRs. Such coordination is feasible: prior work has translated AMR into ontology-aligned KGs and used ontologies to guide KG construction from text (Gangemi et al., 2026; van Cauter and Yakovets, 2024).

2.1 Coordinating KRs

Because different KRs make different *linguistic dimensions* of meaning explicit, KSP operationalizes "linguistic completeness" by coordinating representations derived from source text across three layers: (1) a conceptual layer for domain concepts and constraints, (2) an instance layer for entities and facts, and (3) a temporal layer for events and state changes (Fig. 1). This organization parallels cognitive and text-based agent-memory architectures that distinguish semantic, episodic, and procedural knowledge (Tulving, 1972; Sumers et al., 2023). As the choice of representation shapes both the form and interpretation of knowledge (McLuhan, 1964), KSP adopts ontology, KG, and AMR as representative KRs for these layers while allowing KR form to be extended or replaced as representation research evolves.

2.2 Aligning Concepts and Instances

KSP is representation-agnostic: it aligns KRs to support "linguistic completeness" without prescribing their internal structure. Like HTTP, which standardizes communication without dictating how participating systems store their data (Fielding et al., 2022), KSP defines shared primitives and mappings through which heterogeneous KRs can interoperate. Cross-layer traversal relies on one such *shared primitive* common to symbolic KRs —- an object or entity represented as a node.

This alignment enables traversal from ontology concept nodes to their corresponding KG instance nodes. In conceptual QA, the queried concept may not appear explicitly in the KG or source text. As shown in Fig. 1, KSP can first map the query to

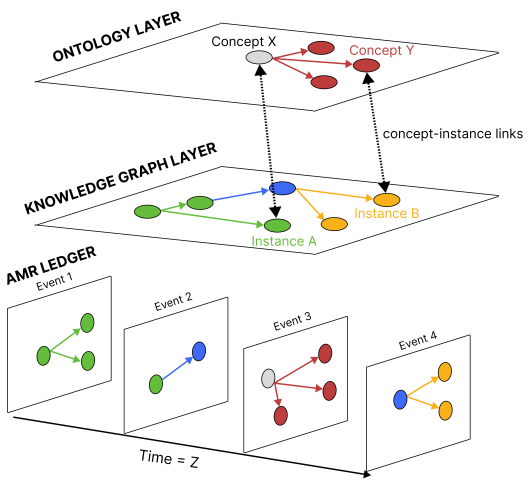


Figure 1: At time Z, each AMR ledger event is sequentially applied to the KG layer. Instance nodes in the KG layer are linked to concept nodes in the ontology layer.

Note. AMR is visualized as an entity graph to illustrate KSP *shared primitive* alignment; predicates and relations are represented as directed edges.

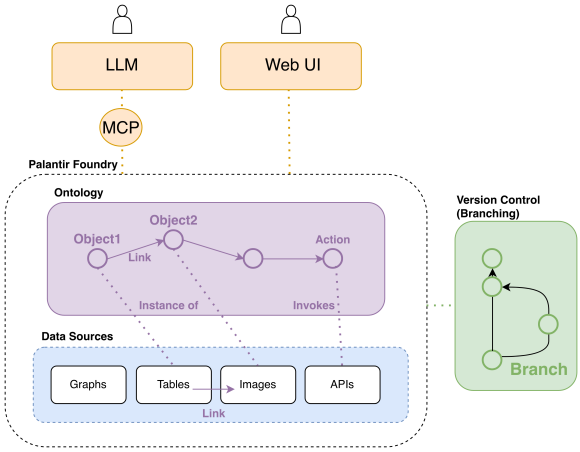


Figure 2: Palantir Foundry as an analogue to KSP. Foundry uses Ontology as a semantic index for multimodal data sources such as KG. Change over time is managed by Version Control.

Concept Y, traverse the ontology, and then identify Instance A in the KG to provide an answer.

2.3 Modeling Temporal Changes

KSP makes the sequence of events and their effects on entities explicit through an event ledger. The ledger serves as the version-control entry point, indexing the KG state associated with each point in time and supporting provenance and reasoning over how knowledge evolves.

Each sentence-level event is parsed into AMR, whose predicate–argument structure captures what happened, who participated, and under what conditions, including distinctions such as negation and modal predicates. KSP anchors AMR event participants to corresponding KG entity nodes and translates the event predicate into a graph operation that updates those entities and their relations.

For a temporal query at a specific time interval (Time=Z in Fig. 1), the relevant operations for each event can be applied forward or reversed to reconstruct the corresponding KG state. Each reconstructed KG therefore represents the knowledge state at a particular point in time.

2.4 Aligning KSP Instances

A KSP instance is a concrete deployment of the protocol, with its own KR realizations, schemas, and data. To support interoperability, each instance implements KSP's shared primitives and exposes explicit, versioned schemas. KSP uses an alignment module to construct versioned *schema mappings* between instances. Candidate mappings are checked against ontology constraints and, when uncertain, reviewed by humans before publication. Existing techniques can already propose many correspondences automatically, although fully reliable alignment remains an open problem (Lushnei et al., 2026). The OBO Foundry provides practical precedent for governed interoperability among independently maintained ontologies (Smith et al., 2007). Because KSP is alignment-method agnostic, improved techniques can be adopted without changing the protocol.

3 Analogue: Palantir Foundry

Palantir Foundry provides a production analogue at scale for KSP's central premise: knowledge becomes more useful for retrieval, reasoning, and action when semantic structure implicit in raw data is made explicit (Fig. 2). Foundry's Ontology maps heterogeneous enterprise data to a defined concept object space and vocabulary, creating a shared symbolic semantic layer used by humans, agents and systems (Palantir Technologies Inc., 2022, 2026c). This parallels KSP's use of Ontology as a concept layer for instances in KGs.

Foundry also illustrates the value of representing state changes explicitly. Version Control and Branching make operational updates and system evolution inspectable and controlled (Palantir Technologies Inc., 2026a,b). This is analogous to KSP's AMR event ledger.

KSP generalizes these principles into an open,

representation-independent protocol in which conceptual, instance, and event representations can be linked, inspected, and extended across KBs.

4 Challenges

4.1 KSP Benchmark and Metrics

KSP requires a benchmark of private and dynamic data that jointly tests conceptual, instance-level, and temporal reasoning while localizing errors by layer. We propose *Full-Chain Accuracy*, which counts a prediction as correct only if the system selects the appropriate knowledge state, follows a valid ontology-constrained reasoning path, returns the correct answer, and provides a valid rule-and-evidence derivation. This is feasible because prior work has evaluated intermediate reasoning through explicit temporal evidence chains and verifiable KG paths (Liu et al., 2026; Sui et al., 2025).

4.2 KSP Standardization and Adoption

KSP requires community agreement on shared primitives, interfaces, and governance, while ontology and dataset construction impose upfront costs. However, it can build on existing infrastructure: the OBO Foundry provides openly maintained ontologies and shared development principles, while the Knowledge Graph Change Language supports standardized, version-controlled ontology updates (Jackson et al., 2021; Hegde et al., 2025). Recent LLM-based methods may further reduce adoption costs by supporting the scalable construction of domain knowledge graphs with low human oversight (Chen et al., 2024). These investments are justified if standardization prevents each system from reconstructing the same semantic connections independently. Like HTTP, KSP reduces coordination costs with upfront standardization– KSP can enable knowledge to move reliably across systems.

4.3 KSP Scalability and Implementation

KSP's event ledger enables past states to be reconstructed by replaying recorded events, but rebuilding derived state from a long history can be costly (Erb et al., 2016; Overeem et al., 2021). Checkpointing can reduce this cost by limiting reconstruction to events recorded after the latest cached state (Burckhardt et al., 2022). Palantir Foundry likewise provides production precedent for maintaining current ontology states alongside action histories at scale (Palantir Technologies, 2026; Palantir Technologies Inc., 2026a). We therefore treat scalable AMR-to-KG reconstruction as an implementation challenge; efficient event translation and temporal retrieval remain open research questions.

5 Research Directions

We envision KSP to make representation of knowledge complete, extensible and modular. KSP can address the failure modes of current KG-RAG and agent-memory systems that lack *structured* **temporal** and **conceptual** grounding.

Neuro-symbolic Training: KSP suggests a direction for neuro-symbolic training by turning aligned text and knowledge representations into ordered state transitions. DRAGON and KG-BiLM provide precedents for jointly learning from text and graph structure (Yasunaga et al., 2022; Chen et al., 2026). By representing AMR events as graph updates, KSP could extend link-prediction objectives to forecasting future events, the edges they modify, and the resulting knowledge state. Temporal KG models already support future-fact prediction and edge-transition modeling from graph histories (Jin et al., 2020; Han et al., 2021). KSP could further use ontology-level constraints to guide which predicted transitions are semantically valid.

Multi-Modality and Cross-Lingual Representation: Because the *shared primitive* of an object or entity as a node exist across modalities (i.e. scene graphs), KSP can align visual and textual entities and events within shared knowledge states (Ma et al., 2022). The same primitives can connect meaning across languages without erasing language-specific distinctions (Wein and Schneider, 2024). By coordinating these representations, KSP preserves semantic connections across modalities, languages, and changing knowledge, providing a richer foundation for retrieval and reasoning.

6 Conclusion

We identify conceptual and temporal context as persistent gaps in KG-based retrieval and show that existing approaches address them through fragmented, specialized structures. KSP instead coordinates complementary knowledge representations to make these dimensions explicit within a shared knowledge state. This provides a foundation for richer, more interpretable reasoning and for extending structured knowledge across modalities and languages.

7 Limitations

KSP coordinates complementary symbolic KRs to make multiple dimensions of textual meaning explicit, but its performance depends on the accuracy, latency, and cost of the components that construct and align each layer. AMR parsing, text-to-KG construction, and ontology learning have advanced substantially (Bai et al., 2022; Zhang and Soh, 2024; Al-Aswadi et al., 2020), but remain imperfect. Because KSP combines multiple processing stages, errors may also propagate across layers (Bouziani et al., 2024).

The design and optimization of these components fall outside the scope of the protocol definition. Instead, KSP specifies interfaces between KRs without prescribing particular representations or parsers, allowing improved components to be adopted without changing the protocol. Large-scale KG deployments and collaborative ontology ecosystems demonstrate the practical viability of the underlying representations (Noy et al., 2019; Jackson et al., 2021), while AMR has supported a broad range of downstream applications (Wein and Opitz, 2024). Confidence propagation, cross-layer validation, and adaptation to noisy real-world data nevertheless remain open implementation challenges.

References

Fatima N. Al-Aswadi, Huah Yong Chan, and Keng Hoon Gan. 2020. Automatic ontology construction from text: A review from shallow to deep learning trend. *Artificial Intelligence Review*, 53:3901–3928.

Jiaxin Bai, Wei Fan, Qi Hu, Qing Zong, Chunyang Li, Hong Ting Tsang, Hongyu Luo, Yauwai Yim, Haoyu Huang, Xiao Zhou, Feng Qin, Tianshi Zheng, Xi Peng, Xin Yao, Huiwen Yang, Leijie Wu, JI Yi, Gong Zhang, Renhai Chen, and Yangqiu Song. 2026. AutoSchemaKG: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20557–20584, San Diego, California, United States. Association for Computational Linguistics.

Xuefeng Bai, Yulong Chen, and Yue Zhang. 2022. Graph pre-training for AMR parsing and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6001–6015, Dublin, Ireland. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Nacime Bouziani, Shubhi Tyagi, Joseph Fisher, Jens Lehmann, and Andrea Pierleoni. 2024. REXEL: An end-to-end model for document-level relation extraction and entity linking. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 119–130, Mexico City, Mexico. Association for Computational Linguistics.

Sebastian Burckhardt, Badrish Chandramouli, Chris Gillum, David Justo, Konstantinos Kallas, Connor McMahon, Christopher S. Meiklejohn, and Xiangfeng Zhu. 2022. Netherite: Efficient execution of serverless workflows. *Proceedings of the VLDB Endowment*, 15(8):1591–1604.

Borui Cai, Yong Xiang, Longxiang Gao, He Zhang, Yunfeng Li, and Jianxin Li. 2023. Temporal knowledge graph completion: A survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 6545–6553.

Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. 2024. SAC-KG: Exploiting large language models as skilled automatic constructors for domain knowledge graph. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4345–4360, Bangkok, Thailand. Association for Computational Linguistics.

Zirui Chen, Xin Wang, Zhao Li, Wenbin Guo, Dongxiao He, Yanbing Li, and Wushour Silamu. 2026. KG-BiLM: Knowledge graph embedding via bidirectional language models. In *Proceedings of the ACM Web Conference 2026*, pages 4290–4301, New York, NY, USA. Association for Computing Machinery.

Hudson de Martim. 2025. An ontology-driven graph RAG for legal norms: A hierarchical, temporal, and deterministic approach. In *Legal Knowledge and Information Systems – JURIX 2025: The Thirty-Eighth Annual Conference*, volume 416 of *Frontiers in Artificial Intelligence and Applications*, pages 282–287. IOS Press.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Benjamin Erb, Gerhard Habiger, and Franz J. Hauck. 2016. On the potential of event sourcing for retroactive actor-based programming. In *Proceedings of the First Workshop on Programming Models and Languages for Distributed Computing*, PMLDC '16, New York, NY, USA. Association for Computing Machinery.

Roy T. Fielding, Mark Nottingham, and Julian Reschke. 2022. HTTP semantics. RFC 9110, Internet Engineering Task Force.

Aldo Gangemi, Arianna Graciotti, Antonello Meloni, Andrea Giovanni Nuzzolese, Valentina Presutti, Diego Reforgiato Recupero, and Alessandro Russo. 2026. Text2AMR2FRED, converting text into RDF/OWL knowledge graphs via abstract meaning representation. *Knowledge and Information Systems*, 68:47.

Jiale Han, Austin Cheung, Yubai Wei, Zheng Yu, Xusheng Wang, Bing Zhu, and Yi Yang. 2025. RAG meets temporal graphs: Time-sensitive modeling and retrieval for evolving knowledge. *Preprint*, arXiv:2510.13590.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. 2021. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8352–8364, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Harshad Hegde, Jennifer Vendetti, Damien Goutte-Gattat, J. Harry Caufield, John B. Graybeal, Nomi L. Harris, Naouel Karam, Christian Kindermann, Nicolas Matentzoglu, James A. Overton, Mark A. Musen, and Christopher J. Mungall. 2025. A change language for ontologies and knowledge graphs. *Database*, 2025:baae133.

Charles F. Hockett. 1960. The origin of speech. *Scientific American*, 203(3):88–97.

Rebecca Jackson, Nicolas Matentzoglu, James A. Overton, Randi Vita, James P. Balhoff, Pier Luigi Buttigieg, Seth Carbon, Melanie Courtot, Alexander D. Diehl, Damion M. Dooley, William D. Duncan, Nomi L. Harris, Melissa A. Haendel, Suzanna E. Lewis, Darren A. Natale, David Osumi-Sutherland, Alan Ruttenberg, Lynn M. Schriml, Barry Smith, and 6 others. 2021. OBO foundry in 2021: Operationalizing open data principles to evaluate ontologies. *Database*, 2021:baab069.

Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6669–6683, Online. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

Pavan Kapanipathi, Ibrahim Abdelaziz, Srinivas Ravishankar, Salim Roukos, Alexander Gray, Ramón Fernandez Astudillo, Maria Chang, Cristina Cornelio, Saswati Dana, Achille Fokoue, Dinesh Garg, Alfio Gliozzo, Sairam Gurajada, Hima Karanam, Naweed Khan, Dinesh Khandelwal, Young-Suk Lee, Yunyao Li, Francois Luus, and 11 others. 2021. Leveraging Abstract Meaning Representation for knowledge base question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3884–3894, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Dong Li, Yichen Niu, Ying Ai, Xiang Zou, Biqing Qi, and Jianxing Liu. 2025. T-GRAG: A dynamic GraphRAG framework for resolving temporal conflicts and redundancy in knowledge retrieval. *Preprint*, arXiv:2508.01680.

Runxuan Liu, Bei Luo, Jiaqi Li, Baoxin Wang, Ming Liu, Dayong Wu, Shijin Wang, and Bing Qin. 2025. Ontology-guided reverse thinking makes large language models stronger on knowledge graph question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15269–15284, Vienna, Austria. Association for Computational Linguistics.

Shihao Liu, Xiaofei Zhou, Bo Wang, and Geyuan Zhang. 2026. Temporal evidence chain for temporal knowledge graph question answering with large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29515–29529, San Diego, California, United States. Association for Computational Linguistics.

Vanessa Lopez, Michele Pasin, and Enrico Motta. 2005. AquaLog: An ontology-portable question answering

system for the semantic web. In *The Semantic Web: Research and Applications*, volume 3532 of *Lecture Notes in Computer Science*, pages 546–562, Berlin, Heidelberg. Springer.

Sviatoslav Lushnei, Dmytro Shumskyi, Severyn Shykula, Ernesto Jiménez-Ruiz, and Artur d'Avila Garcez. 2026. Large language models as oracles for ontology alignment. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2435–2449, Rabat, Morocco. Association for Computational Linguistics.

Yubo Ma, Zehao Wang, Mukai Li, Yixin Cao, Meiqi Chen, Xinze Li, Wenqi Sun, Kunquan Deng, Kun Wang, Aixin Sun, and Jing Shao. 2022. MMEKG: Multi-modal event knowledge graph towards universal representation across modalities. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 231–239, Dublin, Ireland. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Marshall McLuhan. 1964. *Understanding Media: The Extensions of Man*. McGraw-Hill, New York.

Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale knowledge graphs: Lessons and challenges. *Communications of the ACM*, 62(8):36–43.

OpenAI. 2026. GPT-5.6: Frontier intelligence that scales with your ambition. Model release report and system card.

Michiel Overeem, Marten Spoor, Slinger Jansen, and Sjaak Brinkkemper. 2021. An empirical characterization of event sourced systems and their schema evolution: Lessons from industry. *Journal of Systems and Software*, 178:110970.

Palantir Technologies. 2026. Action types: Overview. Palantir Foundry Documentation. Accessed 2026-07-26.

Palantir Technologies Inc. 2022. Ontology: Finding meaning in data. Palantir Blog. Accessed: 2026-07-24.

Palantir Technologies Inc. 2026a. Action types: Action log. Palantir Foundry Documentation. Accessed: 2026-07-24.

Palantir Technologies Inc. 2026b. Global branching: Overview. Palantir Foundry Documentation. Accessed: 2026-07-24.

Palantir Technologies Inc. 2026c. Ontology building: Overview. Palantir Foundry Documentation. Accessed: 2026-07-24.

Kartik Sharma, Peeyush Kumar, and Yunqing Li. 2025. OG-RAG: Ontology-grounded retrieval-augmented generation for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32962–32981, Suzhou, China. Association for Computational Linguistics.

Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J. Mungall, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H. Scheuermann, Nigam Shah, Patricia L. Whetzel, and Suzanna Lewis. 2007. The OBO foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2025. FiDeLiS: Faithful reasoning in large language models for knowledge graph question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8315–8330, Vienna, Austria. Association for Computational Linguistics.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*.

Qingyun Sun, Jiaqi Yuan, Shan He, Xiao Guan, Haonan Yuan, Xingcheng Fu, Jianxin Li, and Philip S. Yu. 2025. DyG-RAG: Dynamic graph retrieval-augmented generation with event-centric reasoning. *Preprint*, arXiv:2507.13396.

Endel Tulving. 1972. Episodic and semantic memory. In Endel Tulving and Wayne Donaldson, editors, *Organization of Memory*, pages 381–403. Academic Press, New York.

Zeno van Cauter and Nikolay Yakovets. 2024. Ontology-guided knowledge graph construction from maintenance short texts. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 75–84, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoman Wang, Yaoze Zhang, Wenzhuo Fan, Hongwei Zhang, Ding Wang, Guohang Yan, Song Mao, Botian Shi, Yunshi Lan, and Pinlong Cai. 2026. IA-RAG: Interval-algebra-driven temporal reasoning for dynamic knowledge retrieval. *Preprint*, arXiv:2606.06044.

Shira Wein and Juri Opitz. 2024. A survey of AMR applications. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6856–6875, Miami, Florida, USA. Association for Computational Linguistics.

Shira Wein and Nathan Schneider. 2024. Assessing the cross-linguistic utility of Abstract Meaning Representation. *Computational Linguistics*, 50(2):419–473.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language–knowledge graph pretraining. In *Advances in Neural Information Processing Systems*, volume 35, pages 37309–37323. Curran Associates, Inc.

Bowen Zhang and Harold Soh. 2024. Extract, define, canonicalize: An LLM-based framework for knowledge graph construction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9820–9836, Miami, Florida, USA. Association for Computational Linguistics.

Ze Yu Zhang, Zitao Li, Yaliang Li, Bolin Ding, and Bryan Kian Hsiang Low. 2026. Respecting temporal-causal consistency: Entity-event knowledge graph for retrieval-augmented generation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2017–2054, Rabat, Morocco. Association for Computational Linguistics.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2024. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4).

Appendices

A Worked Example: Complementary Semantic Coverage Across Representations

This appendix illustrates the rationale behind Table 1 why KSP coordinates multiple knowledge representations rather than selecting a single universal representation. We use one sentence containing an entity reference, a nested event, temporal displacement, negation, attribution, and a previously unseen object:

> *On Tuesday, Maya claimed that the prototype drone SkyCart had not entered the restricted zone the previous night.*

The comparison concerns each representation's *native representational bias*, not its theoretical expressive limit. A sufficiently enriched knowledge graph can encode events, claims, and temporal qualifiers, just as an ontology can be populated with individuals and AMR graphs can be linked across documents. The relevant question is which representation makes each dimension explicit most directly, with the least additional modeling machinery.

A.1 Three Complementary Views

Knowledge-graph view. An enriched KG can represent the sentence by assigning stable identifiers to the entities, claim, and reported event:

```
:Maya        rdf:type :Person .
:SkyCart     rdf:type :PrototypeDrone .
:Zone17      rdf:type :RestrictedZone .

:Claim42     rdf:type :Claim ;
:speaker :Maya ;
:content :EntryEvent17 ;
:assertedAt :Tuesday .

:EntryEvent17 rdf:type :ZoneEntry ;
:agent :SkyCart ;
:destination :Zone17 ;
:polarity :Negative ;
:time :PreviousNight .
```

This representation is intentionally richer than a conventional subject–predicate–object extraction. A flat triple such as `Maya -claimed-> SkyCartEnteredZone17` would not adequately distinguish the claiming event from its propositional content. The enriched graph therefore reifies both the claim and the entry event. This preserves the distinction, but only because the KG has adopted an explicit event and assertion model.

AMR view. A simplified AMR graph directly represents the compositional structure of the sentence:

```
(c / claim-01
:ARG0 (p / person
:name (n / name :op1 "Maya"))
:time (d / date-entity
:weekday (t / tuesday))
:ARG1 (e / enter-01
:polarity -
:ARG0 (s / drone
:mod (p2 / prototype)
:name (n2 / name :op1 "SkyCart"))
:ARG1 (z / zone
:ARG1-of (r / restrict-01))
:time (n3 / night
:mod (p3 / previous))))
```

The outer predicate is the claiming event, with Maya as its speaker. Its `:ARG1` is a separate entry event whose agent is SkyCart and whose destination is the restricted zone. Negation scopes over the entry event rather than the claim: Maya made a claim, but the content of that claim is that the

entry did not occur. The temporal expressions are likewise attached to the events they modify.

Ontology view. The ontology provides reusable concepts and constraints under which these instances can be interpreted:

```
PrototypeDrone  SubClassOf Drone
RestrictedZone  SubClassOf Zone
ZoneEntry       SubClassOf MotionEvent
Claim        SubClassOf CommunicationAct

hasSpeaker      Domain Claim
hasSpeaker      Range Person
hasContent      Domain Claim
hasContent      Range Proposition

hasAgent        Domain Event
hasAgent        Range PhysicalObject
hasDestination  Domain MotionEvent
hasDestination  Range Location
```

The ontology may additionally distinguish claims, observations, hypotheses, and verified facts, and define the evidence required to move information between these categories. For example, a system policy can specify that the content of a `Claim` must not automatically be promoted to an asserted world fact. Such a policy depends on the conceptual distinction between the communication act and the event it describes.

The three representations can be aligned through shared primitives. The AMR node for Maya links to the KG object `:Maya`, which instantiates the ontological class `Person`. The AMR `enter-01` node links to `:EntryEvent17`, which instantiates `ZoneEntry`. The representations therefore describe the same objects and events while preserving different aspects of their meaning.

A.2 Factual Reference: Knowledge Graph

The sentence refers to three identifiable objects: Maya, SkyCart, and a particular restricted zone. The KG is the best fit for maintaining these referents as durable objects across sentences and documents. Once SkyCart has been assigned a stable identifier, subsequent mentions such as *the drone*, *the prototype*, or *it* can be linked to the same object. The graph can then accumulate properties and relations concerning SkyCart independently of the sentence in which it was first mentioned.

AMR represents the relevant participants, but its variables are initially local to the sentence. Additional entity linking or coreference resolution is required to determine that the AMR node `s` denotes the same SkyCart represented elsewhere. The ontology can specify that SkyCart is a prototype drone, but an ontology primarily defines what kinds of objects may exist; it does not by itself provide durable memory for this particular drone. The KG is therefore strongest for preserving instance identity and factual reference over time.

A.3 Compositional Event Meaning: AMR

The sentence contains two nested events: Maya's act of claiming and SkyCart's alleged non-entry. Their participants, polarity, and temporal modifiers must remain correctly scoped. AMR expresses this composition directly. The entry event is the content of the claim, Maya is the participant in the claiming event, SkyCart is the participant in the entry event, and negation applies only to the latter.

A conventional KG extraction can easily flatten this structure. Recording only that Maya "claimed" an association between SkyCart and the zone obscures the internal event roles. Recording that SkyCart did not enter the zone without retaining the surrounding claim incorrectly treats the reported content as an established fact. An enriched KG can avoid these errors by reifying propositions and events, but doing so requires reconstructing much of the predicate–argument and scope structure that AMR provides directly.

The ontology supplies definitions for claims and entry events, but it does not determine how the predicates and arguments are composed in this particular utterance. AMR is therefore the best fit for preserving the sentence-local structure of who did what, to what, under which modifiers, and within whose scope.

A.4 Conceptual Abstraction: Ontology

The sentence presupposes conceptual distinctions that are not fully stated in its surface form. SkyCart is a drone and a physical object; a restricted zone is a kind of location; entering is a motion event; and claiming is a communication act rather than direct evidence that its content is true. These distinctions enable generalization beyond the individual sentence.

The ontology can encode inheritance, property restrictions, domain and range constraints, and equivalences among domain vocabularies. A query for all physical objects alleged to have entered controlled locations can therefore retrieve SkyCart even when the source text uses the more specific

terms *prototype drone* and *restricted zone*. The ontology can also detect incompatible assignments, such as representing a communication act as the physical agent of a motion event.

The KG may attach type labels to its objects, but those labels provide limited conceptual structure unless they are governed by an ontology or comparable schema. AMR identifies sentence-level concepts such as `drone`, `zone`, and `claim-01`, but it does not by itself define their domain-wide inheritance relations or constraints. The ontology is therefore the best fit for abstraction and conceptual reasoning.

A.5 Displacement and Temporality: AMR Event Ledger

The sentence describes an event displaced from the time of utterance. The claiming occurred on Tuesday, whereas the alleged non-entry occurred the previous night. AMR preserves both temporal expressions and attaches each to the event it modifies. This prevents *the previous night* from being incorrectly interpreted as the time of the claim.

AMR alone, however, does not necessarily resolve *Tuesday* or *the previous night* to calendar intervals. KSP therefore stores the AMR-derived events in an event ledger and resolves their relative times against document time, discourse context, and other events. Once Tuesday is anchored to a date, the previous night can be mapped to the corresponding interval. The ledger can then order the claim relative to the reported event and relate both to later observations.

A KG can store timestamps on event nodes, but temporal structure is not guaranteed by an ordinary entity–relation graph, and a current-state KG may discard the path by which its state was reached. An ontology can define intervals, temporal relations, and event categories, but it does not itself provide the sequence of observed events. The AMR-derived ledger is therefore best suited to capturing event-level temporal cues and organizing them into a history, while the KG materializes durable state and the ontology supplies the temporal vocabulary.

A.6 Modality, Negation, and Epistemic Status: AMR and Ontology

The sentence does not assert that SkyCart stayed outside the zone. It asserts that Maya made a claim whose content was that SkyCart did not enter. The distinction is essential: Maya may be correct, mistaken, deceptive, or reporting incomplete information.

AMR preserves the epistemic boundary structurally. The entry event remains embedded under `claim-01`, and its negative polarity remains attached to that embedded event. The ontology then classifies the outer event as a `Claim` and can associate it with a source, confidence, evidence status, or verification requirement. Together, these representations prevent the reported proposition from being silently promoted into the system's asserted world state.

A KG can store the same distinction, as the enriched example does, but only if claims and propositions are explicitly reified. A plain triple such as `SkyCart -didNotEnter-> Zone17` would erase the fact that this information came from Maya's statement. AMR is therefore strongest for extracting the relevant scope and polarity, while the ontology is strongest for determining how claims should be interpreted and governed.

A.7 Reflexiveness and Provenance: Ontology

The sentence is partly about another representation of the world: Maya's statement. Answering questions such as "Who reported the non-entry?", "What evidence supports it?", or "Has the claim been independently verified?" requires the system to reason about statements and their sources as first-class objects.

AMR identifies the reporting event and its speaker, but it does not establish a general provenance model. The KG can attach source metadata or reify the statement, but the interpretation of that metadata depends on an explicit schema. The ontology can distinguish `Claim`, `Observation`, `Evidence`, `Source`, and `VerifiedAssertion`, and define the relations among them. It can also specify that a claim made by an agent is not equivalent to an observation produced by a sensor.

The ontology is therefore the best fit for provenance as a domain concept. The KG instantiates the provenance record, while AMR extracts the source relationship from the sentence.

A.8 Productivity: Extensible Capture and Controlled Schema Evolution

SkyCart may be a newly introduced prototype absent from both the KG and ontology. The KG can immediately add SkyCart as a new entity, while AMR captures its role in the sentence-level event,

including its name and *prototype* modifier, without requiring a predefined domain category.

This flexibility does not ensure shared semantics. The KG may lack an established type or relation for the new object, and AMR remains bounded by its general frames and relation inventory. The ontology can align SkyCart with an existing concept or introduce a class such as `PrototypeDrone`, but doing so requires validation and schema evolution.

KSP therefore treats productivity as a coordinated workflow: the KG admits new entities and facts, AMR preserves their event-level meaning, and the ontology determines which recurring distinctions should enter the shared conceptual model. Unresolved meanings can remain source-specific and provenance-linked until an appropriate alignment is established.

A.9 Combined Interpretation

No individual representation provides the complete interpretation. The KG establishes that Maya, SkyCart, and the restricted zone are durable objects. AMR reconstructs the two events, their participant roles, temporal modifiers, embedding, and negation. The ontology explains that a claim is a communication act whose content should not be treated as independently verified evidence.

KSP coordinates these contributions through aligned object and event identifiers. The sentence can therefore be stored without either flattening its compositional meaning or sacrificing durable retrieval. A later sensor observation that SkyCart did enter the zone would not overwrite Maya's earlier statement. Instead, the event ledger would retain both records, the ontology would distinguish the claim from the observation, and the KG would connect them to the same drone, zone, and time interval. The resulting knowledge state supports factual retrieval, conceptual reasoning, temporal reconstruction, and provenance-sensitive inference over a shared representation of the underlying objects and events.

When String Matching Stands In for a Knowledge Base: A Role-Aware Audit of WikiBigEdit

Edison Yang[1], Neel Marripalapu[2], Akshaj Agadi[3]
[1]Lambert High School [2]University of Texas at Dallas [3]Texas A&M University
Correspondence: edisonly2200@gmail.com

Abstract

Lifelong knowledge-editing benchmarks turn changes in a structured knowledge base into natural-language edits and evaluation questions. Their value depends on that conversion preserving what each fact says and which entity it concerns. We audit WikiBigEdit, a benchmark of more than 500,000 Wikidata updates, and trace its released construction pipeline to a string scan over a text rendering of Wikidata rather than a parse of its data model. From a uniform random sample of 1,500 rows, we audit the 1,458 resolvable rows against their source revisions and find three consequences: qualifier and reference values are promoted to main facts; 23.1% of resolvable rows attach correct answer text to the wrong entity identifier; and locality probes are chosen by surface-string similarity rather than semantic relatedness. Overall, 22.8% of resolvable rows carry unusable text supervision. In a matched reconstruction that holds the relation and question template fixed, similarity-selected probes show 7–12 percentage points more edit-target leakage, that is, the edited answer surfaces in a neighboring question. This effect persists under parameter editing with ROME, while the aggregate locality score remains unchanged. WikiBigEdit's official metric does not fully register the leakage. We validate the audit with an independent parser and release the audit code and a provenance-bearing repair set. Our results do not show that published scores are incorrect; they identify structural errors those scores were not designed to detect.

1 Introduction

Lifelong knowledge editing asks whether a model can absorb a stream of factual updates while preserving unrelated knowledge (De Cao et al., 2021; Mitchell et al., 2022a). WikiBigEdit (Thede et al., 2025) addresses a real gap left by small, synthetic, or pre-cutoff editing benchmarks, operationalizing this at scale: more than 500k question–answer edits mined from consecutive Wikidata dumps, each accompanied by locality, rephrase, and multi-hop probes. Its usefulness does not require every row to be a one-to-one historical replacement, but it does require the supervision to mean what its generated question asserts.

Concretely, each row is a Wikidata triple (s, p, o) rendered as a question–answer pair, for instance (George Harinck, *employer*, Free University of Amsterdam), paired with a *locality probe*: a separate question whose answer should remain unchanged after the edit. For this row the released probe asks "For whom did George Zarnecki work?" (gold: Courtauld Institute of Art), a different person selected because that label is the closest string to the edited subject; after the edit, the Qwen-7B in-context editor answers "Free University of Amsterdam". We call this *edit-target leakage*. On the Wikidata side, a fact is a *statement* whose *main snak* is the property–value pair asserted as the statement's primary fact. A *qualifier* is information that limits or contextualizes that fact, such as its start date; a *reference* records its source; and a *rank* (preferred, normal, or deprecated) marks which of several values is current. The audit asks whether each released (s, p, o) is a main-snak fact about s at the time the row claims, and whether the probe concerns a semantically related entity.

The supervision frequently fails this requirement, for an architectural reason: the released pipeline does not parse Wikidata's statement structure. It runs a Wikipedia text segmenter over the MediaWiki page dump and applies string matching to the result. Wikidata's data model, comprising main snaks, qualifiers, references, and rank (MediaWiki, 2024), is invisible to that process, as is entity identity, which the pipeline discards after extraction and reconstructs from surface labels.

Contributions. (1) A mechanistic audit of a published benchmark's construction, verified against

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 139–150
October 28, 2026 ©2026 Association for Computational Linguistics

its released code and reproduced with a deterministic extraction-trace replay that needs no unreleased intermediates. (2) Prevalence estimates for the resulting defect families on a documented probability sample, with relation-clustered intervals, a sensitivity analysis over three endpoint semantics, a timestamp-robustness check, and an independent-parser validation. (3) A downstream experiment on the benchmark's own data, whose matched reconstruction separates the probe-selection contribution to locality from relation-level generalization, and whose parameter-editing arm (FT-L, ROME) excludes a retrieval artifact. (4) An audited, provenance-bearing repair set with corrected object identifiers.

Although the prevalence estimates are specific to WikiBigEdit, the audit procedure is portable. For any benchmark generated from versioned structured data, it checks whether the construction pipeline (1) distinguishes main values from qualifiers, references, and ranks; (2) preserves entity identifiers rather than reconstructing them from labels; (3) traces each example to the source state at the claimed time; and (4) selects evaluation probes with semantic controls rather than surface similarity alone. Appendix D summarizes these checks, the evidence each produces, and the corresponding repair.

2 Benchmark construction and structural loss

The released `download_wikidata.sh` does not fetch a Wikidata JSON dump; it fetches the MediaWiki page-text dump and pushes it through `gensim.scripts.segment_wiki` (Řehůřek and Sojka, 2010), a Wikipedia-article segmenter (Figure 1). Extraction runs over that text.

Role-blind scan. The extractor `extract_triplets.py` locates each occurrence of the literal `wikibase-entity` (the prefix of the datavalue type `wikibase-entityid`, which is emitted for every entity-valued snak) and reads a property identifier and an entity identifier from the preceding 200 characters. Main, qualifier, and reference snaks are structurally identical at that level, and rank is never read. A qualifier value therefore becomes a fact about the page subject (Figure 2). The authors' own code special-cases a snak-boundary artifact (`'"},"type":'`), which indicates that window overlap was a recognized hazard.

Identity loss. `extract_triplets.py` writes both an identifier file and a label file per interval, but `process_triplets.py` reads only the label file. All downstream stages (ambiguity filtering, locality-probe selection, multi-hop chaining, and QA assembly) operate on ambiguous surface strings, with `fuzzywuzzy` fallbacks. The object identifiers in the public release were reattached afterwards, and because `entity_dict` maps identifier→label, inverting it to recover label→identifier collapses homonyms under a last-write-wins rule.

Provenance recovery. The release is interval-partitioned and carries `subject_id`, `relation_id`, and `object_id`, so a stable audit key and the interval-end timestamp t_1 are constructible from public data. The MediaWiki API returns the exact stored page text at any revision, so we replay the scanner deterministically; serialized marker occurrences align one-to-one with structural snaks, letting us attribute each released pairing to a role.

3 Audit method

3.1 Sampling frame and inclusion flow

The population is the full public release: 502,382 rows across eight interval files. We draw a uniform simple random sample of rows without replacement (seed fixed in advance), so every row has equal inclusion probability and no sampling weights are required. A released triple (s, p, o) that recurs across intervals is a distinct row with a distinct temporal context and is treated as such. Of 1,500 rows drawn, 42 are unresolvable (37 carry no subject identifier, 4 have no Wikidata revision at or before t_1, and 1 is a missing page); these are reported, not silently dropped, and the 37 rows with no subject identifier are themselves release defects. The remaining 1,458 rows are classified. We report prevalences conditional on resolvability, and separately give the unconditional decomposition over all 1,500 rows in §4.

Uncertainty. The estimand is the prevalence in the fixed 502,382-row release, and we sampled rows by SRS, so the primary uncertainty is the design-based interval for a finite-population proportion. With $n/N{=}0.29\%$ the finite-population correction is 0.9985, i.e. negligible, so we report the row-level Wilson interval as primary. Outcomes do correlate within relations, so we additionally report a relation-clustered bootstrap (4000 replicates)

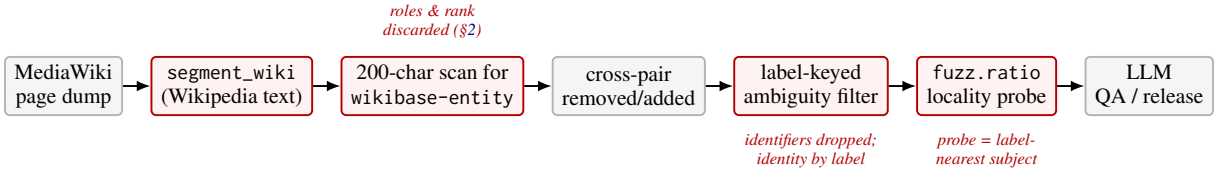


Figure 1: The released pipeline does not parse Wikidata's data model. Red stages are those at which structure is discarded: a Wikipedia text segmenter and a 200-character string scan remove statement role and rank; after extraction only surface labels remain, so ambiguity filtering, probe selection, and QA all operate on strings. Object identifiers in the release are re-attached afterwards.

```
{"mainsnak":{"property":"P39",
   "datavalue":{"value":{"id":"Q30185"}}},
   "rank":"normal",
   "qualifiers":{"P642":[{"property":"P642",
     "datavalue":{"value":{"id":"Q515"}}}]}}
```

Structure: main snak P39→Q30185 (*position held: mayor*), scoped by qualifier P642 (*of: a city*). A valid main fact.

200-char scan: finds `wikibase-entity`, reads the nearest `property` and `id` in its window, emits (P39, Q30185) and, at the next marker, (P642, Q515), in which the qualifier is promoted to a main fact, "*the city is a mayor*".

Figure 2: Worked role-erasure trace. The scanner treats every entity-valued snak identically, so a qualifier value (P642) is extracted as though it were a main fact about the subject. Rank and role, present in the JSON, are invisible to it.

as a robustness analysis for relation heterogeneity; it targets a different estimand, generalization over a population of relations rather than the fixed release, and is wider (relation design effect ≈ 10.7). Appendix A gives its construction and why subject and interval clustering are not used.

3.2 Endpoint semantics and classification

For a row (s, p, o) from interval file `wiki_big_edit_`t_0`_`t_1, we fetch the revision of s current at t_1 and decompose its claims into role- and rank-tagged entity-valued snaks. The primary criterion is $\mathcal{A}_{t_1}^{\rho_{\mathrm{WBE}}}(s,p)=\{o\}$, i.e. the non-deprecated main-snak set for (s, p) is exactly the singleton $\{o\}$. Because this is strict for naturally multi-valued properties, we report two sensitivity variants: ρ_{all} (membership among non-deprecated main snaks) and ρ_{truthy} (membership among best-rank main snaks). We distinguish a row whose subject carries the property but not the value from one whose subject lacks the property in every role, the latter being the signature of a wrong-entity attachment. For the former, we resolve whether the entity's actual value shares the released label (a misattached identifier) or does not (a genuinely incorrect value).

Class	k	%	Wilson	rel-clu
endpoint-valid (ρ_{WBE})	719	49.3	46.8–51.9	42.2–56.2
object-id misattached	337	23.1	21.0–25.3	16.0–31.2
property absent (ident.)	149	10.2	8.8–11.9	7.8–13.0
role-erased (qual.)	82	5.6	4.6–6.9	3.0–9.5
main-supp., non-unique	69	4.7	3.8–5.9	2.2–7.5
object absent, unresolv.	44	3.0	2.3–4.0	0.6–7.4
object genuinely wrong	43	2.9	2.2–3.9	1.0–4.9
role-erased (ref.)	15	1.0	0.6–1.7	0.1–2.5

Table 1: Class prevalence at the interval-end snapshot, n=1458, with counts k, row-level Wilson, and relation-clustered bootstrap 95% intervals.

3.3 Validation

We froze the classifier after a stratified adjudication (24/24) and 11 unit tests, then checked it against an independent re-parser. For 164 rows stratified across all classes, we re-fetched each row's pinned revision and re-derived the verdict with a separate minimal parser that shares no code with the audited pipeline. Taking "invalid under ρ_{WBE}" as the positive class, the confusion matrix is TP=123, FP=0, FN=0, TN=41: positive predictive value 100% $[97.0, 100]$, false-omission rate 0% $[0, 8.6]$ (Clopper–Pearson), and 29/29 agreement on the role-erasure cases that exercise the qualifier/reference traversal. This establishes that the extraction and classification faithfully implement the ρ_{WBE} criterion over the source revision. It does not adjudicate the choice of ρ_{WBE} itself, since the re-parser shares that definition and the source revisions; probing that choice requires the alternative endpoint criteria ρ_{all} and ρ_{truthy}, whose sensitivity we report separately.

4 Population findings

Table 1 reports class prevalence on the n=1458 resolvable rows. The Wilson interval is the primary (design-based) uncertainty for the fixed release; the

	%	rel-clu 95%
Endpoint semantics (valid)		
ρ_{WBE} (unique main $=\{o\}$)	49.3	[42.2, 56.2]
ρ_{all} (member of main set)	54.0	[45.5, 62.1]
ρ_{truthy} (member, best rank)	54.0	[45.5, 62.1]
Four-way decomposition		
A. fully valid (unique main)	49.3	[42.2, 56.2]
B. structurally valid, non-unique	4.7	[2.2, 7.5]
C. valid text, wrong identifier	23.1	[16.0, 31.2]
D. unusable text supervision	22.8	[17.1, 30.4]

Table 2: Endpoint-semantics sensitivity and a four-way split. A+B is validity under ρ_{all} (54.0%); A+B+C is defensible text supervision (77.2%); D is the genuinely unusable share.

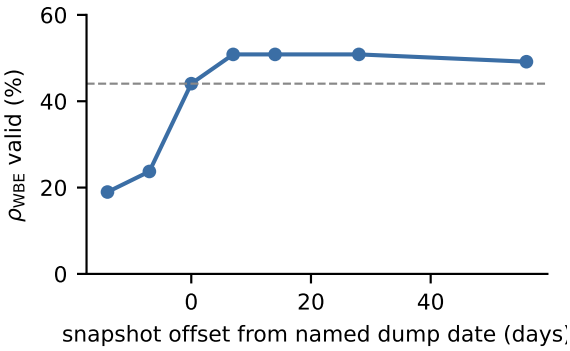


Figure 3: ρ_{WBE} validity versus snapshot offset. The curve plateaus, so the defect is not a timestamp artifact.

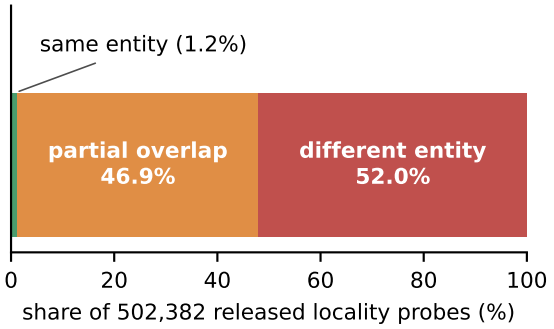


Figure 4: Entity provenance of the 502,382 released locality probes, by content-token overlap between the probe question and the edited subject label. *Same entity*: the subject string appears verbatim, or all its content tokens appear, in the probe. *Partial overlap*: the probe shares at least one but not all of the subject's content tokens. *Different entity*: no shared content token.

relation-clustered interval is the wider robustness bound for generalization across relations. The finding is also stable across intervals: the per-interval invalid rate ranges from 38.8% to 52.1%, and leaving any single interval out moves the pooled invalid rate only within $[43.8, 47.0]\%$ around 46.0%, so no interval dominates.

Endpoint sensitivity and decomposition. Table 2 reports validity under all three endpoint semantics and separates the release into four categories, which are best interpreted jointly rather than through the single strict number. The most consequential quantity is the 22.8% of rows whose text supervision is unusable, rather than the 49.3% strict-validity rate, because a misattached identifier (23.1%) leaves the question–answer pair correct even though the structured object is wrong, and a non-unique main answer (4.7%) remains valid under ρ_{all}. Rank policy is immaterial here, as ρ_{all} and ρ_{truthy} coincide.

Table 2 is conditional on resolvability. Computed unconditionally over all 1,500 sampled rows, and adding an "unresolvable release record" fifth class (2.8%, dominated by the 37 rows with no subject identifier), the shares become A 47.9%, B 4.6%, C 22.5%, D 22.2%, and E 2.8%. Because the no-subject-identifier rows are themselves defects, the genuinely unusable share rises to 24.7% $[22.6, 26.9]$. The prevalence estimates therefore generalize directly to the resolvable subpopulation and slightly understate malformation for the release as a whole.

Robustness to snapshot timing. Sweeping the assumed snapshot date from -14 to $+56$ days (Figure 3), ρ_{WBE} validity peaks at 50.9% and the object-present rate plateaus at 54.2% from $+7$ onward, while the role-erasure signal is invariant. At least approximately 46% of rows are never uniquely main-supported anywhere in that window. The benchmark's own tags are internally consistent at t_0 (`update` rows: property present in 13/13), confirming the alignment.

Wrong entity identifiers. Within the largest non-valid block, window misassociation is excluded (the object is absent from the entity entirely in 98.4% of cases), and 79.5% carry the released label on a different, homonymous identifier: `Finland=Q2580855` rather than the canonical `Q33`, and `Canada=Q14624136` rather than `Q16`. The text supervision is correct, whereas the structured object identifier is not.

5 Provenance of the locality probes

`create_locality_probes` is intended to select a same-subject unchanged fact but falls back to the

`fuzz.ratio` argmax over subject labels; an earlier ambiguity filter renders the intended branch nearly unreachable. A locality probe should target an entity other than the edited one, so a different-entity probe is not itself a defect; our concern is narrower, that the selection is driven by surface-string similarity rather than by any semantic relation to the edit. Over all 502,382 rows (Figure 4), the probe plausibly concerns the edited entity in only 1.15% of rows; 52.0% share no content token with the edited subject and are selected purely on character-level string overlap, and the remaining 46.9% share some but not all of its tokens. Against a null that pairs each edit with a random probe, the observed edit–probe string similarity is 66.9 versus 31.4, and the fraction above similarity 70 is 48.2% versus 0.4%, an enrichment of approximately 120×. Representative instances include Limni→Lin, Lynnville→Lunéville, and Coldwater→Clearwater. We do not claim such probes are useless as difficulty items; we claim that string-similarity selection is not a semantically controlled way to measure locality, and §6 shows it induces edit-target leakage that the counterfactual design rules out as legitimate relatedness and that the official metric does not register.

6 Downstream consequence

This section shows that the structural defect has measurable consequences for how the benchmark is used; the prevalence audit of §4 stands independently of it. It asks two separate questions. First, does an edit cause the model to repeat the new target when answering a supposedly neighboring question? We measure this as *edit-target leakage*. Second, does the edit stop the model from preserving the probe's original answer? We measure this as *locality* (the answer is unchanged by the edit), and report WikiBigEdit's own metric (the model still produces the gold answer `loc_ans`) alongside it. Three experiments address these questions. The *as-deployed* contrast (§6.2) evaluates WikiBigEdit's released probes but confounds shared relation with subject similarity; the *matched reconstruction* (§6.3) isolates similarity-associated probe selection by holding relation and question template fixed; and the *parameter-editing* experiment (§6.4) tests whether the effect persists without any retrieval step.

Editor, backbone	N	Succ.	Leak	Loc	WBE
ctx/retr., Qwen-7B	400	n/a	$+33.8^{***}$	-27.5^{***}	-6.0^{**}
ctx/retr., Llama-8B	400	n/a	$+23.8^{***}$	-16.5^{***}	-1.5
FT-L, Qwen-1.5B	300	53	$+22.3^{***}$	-13.7^{**}	-4.7
ROME, Qwen-1.5B	300	83	$+20.3^{***}$	$+2.3$	$+0.7$

Table 3: As-deployed difference-in-differences in percentage points (own released probe versus random probe, edit versus null). N is the number of edits; Succ. is edit success (%); *Leak* is edit-target leakage; *Loc* is locality (answer unchanged); *WBE* is WikiBigEdit's own metric (the model still produces `loc_ans`). Significance is from relation-clustered paired sign-flip permutation tests (20,000 sign assignments, so the smallest reportable p is 5×10^{-5}): $^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$; unmarked is not significant. This contrast confounds shared relation with subject similarity; see Table 4.

6.1 Setup and definitions

We evaluate whether the probe design alters locality behavior using the benchmark's own `loc`/`loc_ans` probes and edits. Answers are compared after normalization (lowercasing, punctuation stripping, article removal), with a token-boundary containment test. For an edit that asserts object $o^{\dagger}$ and a probe q with baseline answer a_0, the two per-probe binary outcomes are *locality* $\mathbb{1}[a(q){=}a_0]$ and *leakage* $\mathbb{1}[o^{\dagger} \in a(q)]$, the edit's target appearing in the probe answer; the WikiBigEdit metric is $\mathbb{1}[\text{gold} \in a(q)]$. Because each item contributes both an own probe and a matched comparison probe under an edit condition and a null (unrelated-edit) condition, we report the difference-in-differences

$$\mathrm{DiD} = \tfrac{1}{N}\textstyle\sum_i \left[(y^{\mathrm{own}}_{i,\mathrm{edit}} - y^{\mathrm{cmp}}_{i,\mathrm{edit}}) - (y^{\mathrm{own}}_{i,\mathrm{null}} - y^{\mathrm{cmp}}_{i,\mathrm{null}})\right],$$

which removes any edit-independent difference between the probes. Because the per-item DiD terms are discrete and heavily tied, significance uses a paired sign-flip (Rademacher) permutation test rather than Wilcoxon, run both at the item level and clustered by relation (20,000 sign assignments, seed fixed in advance; Appendix A). Confidence intervals use an item-level bootstrap and a relation-clustered bootstrap (4000 replicates each). The edit target is counterfactual (§6.3), so leakage cannot be a pre-existing correct answer.

6.2 As-deployed contrast

Using the released locality probe as the own probe and a random other row's probe as the comparison, applying an edit causes the model to answer

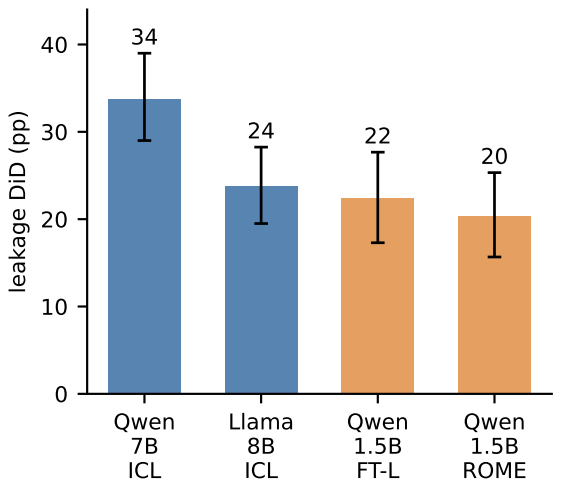


Figure 5: As-deployed leakage difference-in-differences (95% CI). The edit's target leaks into the released neighbor probe in all four settings, including the two parameter editors, which have no retrieval step.

the own probe with the edit's target in 20–35% of cases, against approximately 1% for the comparison probe (Figure 5, Table 3). This contrast reflects the benchmark as deployed, but it confounds two factors: the released probe shares the edit's *relation* by construction, whereas the random comparison shares neither relation nor subject. It therefore overstates the probe-selection effect in isolation, which we separate in the following subsection.

6.3 Matched reconstruction

To separate the contribution of a shared relation from that of a similar subject, we construct a matched design: for each edit we build two comparison probes drawn from the *same relation*, one chosen as the fuzzy-nearest subject label and one at random. The edit target is *counterfactual*, sampled from the relation's object pool and required to differ from the true object and from every probe answer, so leakage can only arise if the model emits the specific injected object. Relation and question template are held fixed by construction, and the strata are constructed to differ primarily in subject similarity (0.643 versus 0.259): they are closely balanced on answer length (2.47 versus 2.43 tokens), subject-label length (3.39 versus 3.61 tokens), target-in-subject token overlap (7.0% versus 7.8%), and near-floor baseline accuracy (1.8% versus 2.8%), the last a small residual difference we note rather than fully control.

In this matched design (Table 4), similarity-associated probe selection is associated with $+11.5$ percentage points of additional leakage on Qwen-7B (relation-clustered permutation $p{=}3.5{\times}10^{-4}$) and $+7.0$ on Llama-8B ($p{=}0.03$) after matching on relation and template, while locality is unchanged (DiD near zero, $p{>}0.4$). Because the strata are matched but not randomized, we read this as the difference associated with similarity-driven selection under this reconstruction rather than as an unconditional causal effect. Leakage rises monotonically with probe-subject similarity when the real edit is applied but stays flat under the null (Figure 6). The counterfactual target further disfavors legitimate semantic relatedness, since a related neighbor would not produce the specific false object. The design is observational rather than randomized, however: fuzzy-nearest labels may still differ from random ones in entity type, pretraining frequency, or morphology, so we attribute the effect to similarity-associated selection rather than to lexical similarity in isolation.

Editor (backbone)	measure	DiD	95% CI
in-ctx Qwen-7B	leakage	$+11.5^{***}$	$[+6.6, +16.8]$
in-ctx Qwen-7B	locality	-2.3	$[-7.2, +3.1]$
in-ctx Llama-8B	leakage	$+7.0^{*}$	$[+1.3, +12.4]$
in-ctx Llama-8B	locality	-2.8	$[-9.3, +5.1]$
ROME Qwen-1.5B	leakage	$+12.3^{***}$	$[+7.0, +17.7]$
ROME Qwen-1.5B	locality	-1.7	$[-6.3, +3.0]$

Table 4: Matched reconstruction, relation and template held constant ($N{=}399$ per in-context backbone, 300 for ROME). Similarity-associated selection is associated with additional leakage but leaves locality unchanged, including under parameter editing. Item-level bootstrap CIs; significance from relation-clustered sign-flip permutation tests (20,000 sign assignments) for every row: $^{*}p{<}0.05$, $^{**}p{<}0.01$, $^{***}p{<}0.001$; locality rows are all $p{>}0.4$. The ROME row spans 99 relations; its relation-clustered leakage interval $[+7.2, +17.1]$ and permutation ($p{=}2{\times}10^{-4}$) match the item-level values shown.

Backbone, probe	n	null	own	loss (CI)
Qwen-7B, own	61	80.3	44.3	$+36.1$ $[23.0, 49.2]$
Qwen-7B, comp.	65	73.8	78.5	-4.6 $[-15.4, 6.2]$
Llama-8B, own	90	80.0	74.4	$+5.6$ $[-4.4, 15.6]$
Llama-8B, comp.	99	86.9	86.9	$+0.0$ $[-7.1, 7.1]$

Table 5: Corruption of previously correct knowledge on released probes. Among probes answered correctly at baseline, gold-answer retention (%) under an unrelated (null) edit versus the probe's own edit; *loss* is the within-probe difference (null − own) in percentage points with a 95% paired-bootstrap interval. Only the Qwen-7B own probe shows significant loss (paired sign test $p{=}1.0{\times}10^{-5}$); all other contrasts are non-significant.

Leakage versus corruption of held knowledge. Because baseline accuracy on these reconstructed probes is near the floor (1.8% versus 2.8%), the effect above is edit-target leakage into probes the model could rarely answer to begin with: the injected object appears where there was seldom a correct neighboring fact to preserve. Corruption of *previously correct* knowledge is a separate claim, which we test on the released probes, where baseline accuracy has headroom (15–25%). We restrict to probes the model answered correctly before any edit and contrast two conditions on those same probes: applying the row's own edit versus applying an unrelated (null) edit. The null condition, not a 100% reference, is the baseline, because applying any in-context edit perturbs a few answers on its own; the edit-attributable loss is the within-probe difference between the two (Table 5). On Qwen-7B this loss is 36.1 pp (80.3% retained under the null edit versus 44.3% under the own edit, n=61), with a paired sign test over the 24/2 discordant probes giving p=1.0×10^{-5}; this is the within-probe null-versus-own contrast, not the full own-versus-comparison difference-in-differences. On Llama-8B the same loss is 5.6 pp (n=90, p=0.38), not significant. The comparison probe shows no loss on either backbone (−4.6 pp and 0.0 pp). Corruption of held knowledge is therefore demonstrated on one backbone but not the other, whereas edit-target leakage is large and consistent across both.

6.4 Parameter editing

The as-deployed effect might be attributed to similarity-based retrieval, but it persists under parameter editing, which has no retrieval step (Table 3), and it also holds under the matched design: running ROME on the reconstruction (relation and template fixed) gives a leakage DiD of +12.3 pp (p=2×10^{-4}; Table 4), close to the in-context backbones, so the effect is not an artifact of the confounded as-deployed contrast. These 300 matched items span 99 relations, so we apply the same relation-clustered inference used for the in-context rows; the clustered interval [+7.2, +17.1] and permutation (p=2×10^{-4}) coincide with the item-level values, confirming the effect is not carried by a few large relation blocks. All parameter-editing DiDs are intention-to-treat over every attempted edit; conditioning on the subset with a verified successful edit raises leakage further (FT-L +22.3 → +35.4, ROME as-deployed +20.3 → +24.1, ROME matched +12.3 → +15.2), so the reported figures are the conservative choice given FT-L's 53% success rate. ROME also leaves aggregate behavior unchanged, moving locality by only +2.3 pp (p>0.5) as-deployed and −1.7 pp matched, yet it leaks the edit's target to the neighbor probe. This pattern is difficult to explain solely through generic relation-level degradation, although aggregate locality alone cannot exclude heterogeneous offsetting item-level effects, and leakage and unchanged-answer rates measure distinct events. These results do not constitute a change to WikiBigEdit's reported score. Its own metric (the WBE column of Table 3, the model still producing `loc_ans`) moves by −6.0, −1.5, −4.7, and +0.7 pp; only the Qwen-7B in-context setting is significant under the relation-clustered test (FT-L is borderline, p=0.05 item-level and 0.08 relation-clustered), and the surgical ROME editor is null. That metric therefore under-registers the large underlying leakage.

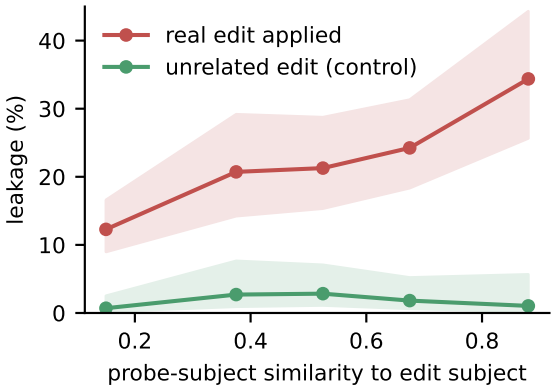


Figure 6: Leakage increases with probe-subject string similarity when the real edit is applied but remains flat under an unrelated edit; the effect is therefore edit-driven rather than a property of similar strings. A Cochran-Armitage trend test over the ordered bins confirms the increase under the real edit (z=4.85, p=1.2×10^{-6}) and finds no trend under the null (p=0.48). Bands are 95% Wilson intervals.

7 Audited repair set

We do not reconstruct the full benchmark; a deterministic screen over all 502,382 rows needs the historical entity state at each edit's revision, which the live MediaWiki API serves only per revision at a low rate. Bulk history dumps or batched revision retrieval could remove this per-row bottleneck, but at substantial storage and engineering cost, so a full screen is left to future work; Wikidata's full-history

dumps, or derived resources such as Wikidated (Schmelzeisen et al., 2021), would supply the per-revision entity state without per-row API calls. The extraction defect itself is straightforward to avoid: the Wikidata JSON dump exposes role and rank explicitly, and existing structured parsers consume qualifiers and ranks directly, so a role- and rank-aware rebuild of the pipeline is an engineering task rather than a research problem. We instead release an audited, provenance-bearing repair set for the sampled rows, organized into three tracks: *atomic* (role-correct unique main facts, 49.3%), *endpoint* (role-correct but non-unique, 4.7%), and *invalid* (46.0%, retained with its diagnosis). Each row carries per-row provenance (snapshot revision, endpoint answer sets, and role evidence), and 337 object identifiers within the audited subset are corrected. The manifest states this scope explicitly: per-row labels exist for the sampled rows, and the prevalence estimates generalize directly to the resolvable subpopulation under the sampling design of §3.1. Appendix B shows repaired records.

8 Related work

Knowledge editing and its benchmarks. Parameter editors such as ROME (Meng et al., 2022) and MEMIT (Meng et al., 2023), hypernetwork editors (De Cao et al., 2021; Mitchell et al., 2022a), and memory-based methods such as SERAC (Mitchell et al., 2022b), GRACE (Hartvigsen et al., 2023), and WISE (Wang et al., 2024a) are evaluated on locality, the preservation of neighboring knowledge, and are commonly implemented in toolkits such as EasyEdit (Wang et al., 2024b). Benchmarks range from single-fact suites derived from zsRE (Levy et al., 2017) and CounterFact (Meng et al., 2022) to multi-hop (Zhong et al., 2023) and survey collections (Yao et al., 2023). Closest to our target, WikiFactDiff (Ammar Khodja et al., 2024) derives atomic factual updates from two Wikidata snapshots, and WikiBigEdit (Thede et al., 2025) scales this to real, temporally ordered Wikidata updates. Our work is orthogonal to the design of an editor: we audit the benchmark on which such methods are scored.

Specificity and side effects of editing. Several studies show that edits leak into related content: RippleEdits (Cohen et al., 2024) measures ripple effects on entailed facts, improved specificity benchmarks reveal unwanted collateral changes (Hoelscher-Obermaier et al., 2023), and editing can distort knowledge the model already holds (Li et al., 2024). Although factual associations can be localized to specific neurons (Dai et al., 2022), localization has been shown not to predict editability (Hase et al., 2023). Our leakage measure is in this tradition, but the leakage we identify is induced by the *benchmark's own probe selection* rather than by an editor.

Dataset and benchmark audits. Systematic auditing of NLP resources has exposed label errors (Northcutt et al., 2021), split artifacts (Gorman and Bedrick, 2019), and problems in dataset development and use (Paullada et al., 2021), motivating broader construct-validity and measurement concerns (Bowman and Dahl, 2021; Bender and Friedman, 2018; Raji et al., 2021; Jacobs and Wallach, 2021). We contribute a population-level, role-aware audit of a knowledge-editing benchmark against its source knowledge base. The two failure modes we isolate, a text scan substituting for structured parsing and string similarity substituting for semantic selection, are not specific to WikiBigEdit and are worth checking in any benchmark generated automatically from a structured knowledge base.

Wikidata quality and temporal knowledge. Wikidata (Vrandečić and Krötzsch, 2014) data quality has been surveyed extensively (Piscopo and Simperl, 2019), and Wikidated (Schmelzeisen et al., 2021) reconstructs its revision history as triple additions and deletions. Temporal and multi-valued knowledge evaluation (Dhingra et al., 2022; Jang et al., 2022; Elazar et al., 2021) motivates role- and rank-aware semantics. We add a role-aware decomposition keyed on recovered interval timestamps that the triple view does not capture.

9 Limitations

Our central finding is a prevalence claim about the released supervision. Sampling error is bounded by the reported intervals and implementation correctness by an independent re-parser (PPV 100%, §3.3); beyond these, the estimate could be wrong in only two substantive ways, if the revisions we audit differed materially from those the pipeline consumed, or if ρ_{WBE} were an inappropriate criterion. We test both, with the snapshot-offset robustness check (Figure 3) and the three endpoint-semantics variants (Table 2). The number of editors or backbones evaluated downstream is not among the ways

this finding could be wrong; the limitations below bound secondary claims rather than this one.

The route by which incorrect identifiers reach the release is inferred from the code, whereas the consequence is measured directly, so the prevalence estimate holds whether or not our mechanistic account is exact. Because the validation shares the source revisions and the ρ_{WBE} definition with the audited pipeline, it establishes that the implementation faithfully executes the criterion against the reference state rather than adjudicating the criterion itself, for which we report ρ_{all} and ρ_{truthy} sensitivity. The downstream experiment demonstrates that the defect has measurable consequences; it is not the basis of the audit. It covers in-context and retrieval editing and two gradient editors (FT-L, ROME) on a single 1.5B model, with the matched reconstruction run for in-context editing and, for ROME, under weight editing; MEMIT (Meng et al., 2023) and larger backbones would extend its external validity but cannot revise the prevalence estimate. Within that experiment the edit-target leakage effect is large and consistent, while corruption of previously correct knowledge is demonstrated on one backbone but not the other. Critically, we do **not** claim that WikiBigEdit's reported locality numbers are materially incorrect or that any published ranking changes, since the effect on its own metric is small and backbone-dependent. What we establish is that the supervision is substantially malformed and the locality probes are selected by an uncontrolled string-similarity rule, that both problems are measurable and mechanistically explained, and that an audited repair set is available.

References

Hichem Ammar Khodja, Frédéric Béchet, Quentin Brabant, Alexis Nasr, and Gwénolé Lecorvé. 2024. WikiFactDiff: A large, realistic, and temporally adaptable dataset for atomic factual knowledge update in causal language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING)*.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics (TACL)*.

Samuel R. Bowman and George E. Dahl. 2021. What will it take to fix benchmarking in natural language understanding? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. 2024. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics (TACL)*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics (TACL)*.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics (TACL)*.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Thomas Hartvigsen, Swami Sankaranarayanan, Hamid Palangi, Yoon Kim, and Marzyeh Ghassemi. 2023. Aging with GRACE: Lifelong model editing with discrete key-value adaptors. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? Surprising differences in causality-based localization vs. knowledge editing in language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL*.

Abigail Z. Jacobs and Hanna Wallach. 2021. Measurement and fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*.

Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. 2022. TemporalWiki: A lifelong benchmark for training and evaluating ever-evolving

language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL)*.

Zhoubo Li, Ningyu Zhang, Yunzhi Yao, Mengru Wang, Xi Chen, and Huajun Chen. 2024. Unveiling the pitfalls of knowledge editing for large language models. In *International Conference on Learning Representations (ICLR)*.

MediaWiki. 2024. Wikibase/DataModel: statements, main snaks, qualifiers, references, and rank. https://www.mediawiki.org/wiki/Wikibase/DataModel. Accessed 2026.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning (ICML)*.

Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. In *NeurIPS Datasets and Benchmarks Track*.

Amandalynne Paullada, Inioluwa Deborah Raji, Emily M. Bender, Emily Denton, and Alex Hanna. 2021. Data and its (dis)contents: A survey of dataset development and use in machine learning research. *Patterns*, 2(11):100336.

Alessandro Piscopo and Elena Simperl. 2019. What we talk about when we talk about Wikidata quality: a literature survey. In *Proceedings of the 15th International Symposium on Open Collaboration (OpenSym)*.

Inioluwa Deborah Raji, Emily M. Bender, Amandalynne Paullada, Emily Denton, and Alex Hanna. 2021. AI and the everything in the whole wide world benchmark. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Lukas Schmelzeisen, Corina Dima, and Steffen Staab. 2021. Wikidated 1.0: An evolving knowledge graph dataset of Wikidata's revision history. *arXiv preprint arXiv:2112.05003*.

Lukas Thede, Karsten Roth, Matthias Bethge, Zeynep Akata, and Thomas Hartvigsen. 2025. WikiBigEdit: Understanding the limits of lifelong knowledge editing in LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *PMLR*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. WISE: Rethinking the knowledge memory for lifelong model editing of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Peng Wang, Ningyu Zhang, Bozhong Tian, Zekun Xi, Yunzhi Yao, Ziwen Xu, Mengru Wang, Shengyu Mao, Xiaohan Wang, Siyuan Cheng, Kangwei Liu, Yuansheng Ni, Guozhou Zheng, and Huajun Chen. 2024b. EasyEdit: An easy-to-use knowledge editing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A Inference details

Relation-clustered bootstrap intervals resample whole relations uniformly with replacement: for each of 4000 replicates (a count fixed in advance, seed 20260722) we draw R relations with replacement, where R is the number of distinct relations, pool all rows in the drawn relations, and recompute the statistic; cluster sizes therefore enter naturally and larger relations are more influential in proportion to their size. Because it pools

the resampled relations' rows, it preserves the empirical relation-size distribution and stays row-weighted; it widens the interval to reflect within-relation dependence rather than reweighting toward an equal-per-relation target. Subject clustering is negligible, as each subject appears once in the sample, and the 8 intervals are too few to bootstrap. The paired sign-flip permutation test for the downstream difference-in-differences draws 20,000 Rademacher sign vectors (seed 20260723); the item-level test flips each per-item term independently, and the relation-clustered test flips all items in a relation together. Reported p-values are two-sided, $(\#\{|\text{permuted}| \geq |\text{observed}|\}+1)/(B+1)$.

Leakage and locality use token-boundary containment after normalization (lowercasing, punctuation stripping, article removal). Token-boundary matching avoids substring false positives within a token, but a short target that is a distinct token inside a longer entity name can still register; we did not observe this affecting the counterfactual leakage measure, since counterfactual targets are sampled to differ from every probe answer.

B Repaired-record examples

Table 6 shows representative repaired records with the released row, the pinned source revision, the diagnosis, and the corrected object identifier where applicable.

object-id misattached (rev. 2103510427). released: Reinhard Merkel (Q2140863), place of birth, Hof (Q1852145). corrected: object Q1852145 → **Q6938** (label "Hof"; ρ_{WBE} set {Q6938}).
object-id misattached (rev. 2082862946). released: Thunbergia grandiflora (Q138397), subclass of, vine (Q2135068). corrected: object Q2135068 → **Q917284** (label "vine").
role-erased, qualifier (rev. 2179477364). released: solar eclipse of 1836 BC (Q5693372), issued by, NASA (Q23548). diagnosis: value occurs only as a qualifier on a P528 (catalog code) statement; ρ_{WBE} main set empty. Retained in the invalid track.
role-erased, qualifier (rev. 2126904748). released: P Rajeev (Q16190976), electoral district, Ernakulam constituency (Q3595497). diagnosis: qualifier on a P3602 (candidacy) statement, not a main fact.

Table 6: Repaired-record examples. Each carries its source revision, so the verdict is reproducible.

C Reproducibility

Code, the per-row audit manifest (sampled row identity, inclusion status, audit label, pinned revision, and corrected identifier), the three-track repair set, and the script generating each table and figure are released at `https://github.com/edisonly1/wikibigedit-audit-artifact`.

Seeds. Audit sampling and the relation-clustered bootstrap use seed 20260722; probe-set construction, the downstream experiments, and the sign-flip permutation test use 20260723; the extraction-verification subsamples use 20260724 and 20260725. No result depends on an unseeded draw.

Table and figure sources. Table 1 (prevalence and relation-clustered intervals), Table 2 (endpoint sensitivity), Table 3 (as-deployed effects), Table 4 (matched reconstruction), and Figures 3–6 are each produced by a named script in the repository, listed in its reproducibility guide.

Prompts and decoding. The question-answering system prompt is *"You answer factual questions with a short noun phrase and nothing else. Output only the answer, at most a few words."* Edits are supplied as optional context ("You may use the following fact if, and only if, it is relevant…"); the SERAC-style scope gate uses *"Does the stored fact directly answer this exact question?"* (selected against labeled cases: accuracy 0.80, specificity 6/6). Decoding is greedy and deterministic: for Ollama backbones, temperature 0, top-k 1, top-p 1, seed 0, 24 new tokens; for the Hugging Face backbone, `do_sample=False`, one beam.

Determinism protocol. Under greedy decoding the *first* evaluation of a prompt can differ from all subsequent ones, which we traced to `llama.cpp` taking a different prefill path on a cold KV cache; it is a decoding-engine artifact, not sampling. We characterized this before running the reported experiments (see `repeat_stability.py`): the first-to-second call mismatch is 6.7% (Qwen) and 3.3% (Llama), while every later call is identical (0/120 mismatches per backbone). The mismatches are idiosyncratic near-variants (e.g. "Munich"/"Ulm", "naturalism"/"naturalist"), not a systematic bias toward a particular answer. The adopted protocol, fixed in advance, discards the first evaluation and uses the second, so every reported answer is from the stable regime.

Provenance. Ollama 0.32.1 with `qwen2.5:7b` (digest `845dbda0ea48`) and `llama3.1:8b` (digest `46e0c10c039e`), both `Q4_K_M`; `Qwen2.5-1.5B-Instruct` (fp32) for parame-

ter editing. Python 3.12, `torch` 2.13 (CUDA 13.0), `transformers` 5.14, on one NVIDIA RTX 5070 (12.8 GB).

Editor hyperparameters. FT-L edits the layer-12 MLP down-projection with Adam (lr 10^{-3}, 30 steps) under an L_∞ constraint of 5×10^{-3}; edit success 52.7%. ROME applies a rank-one update to the same layer, optimizing the value vector (lr 0.5, 40 steps) with key covariance C (ridge $\lambda = 0.05 \cdot \mathrm{mean}(\mathrm{diag})$) estimated from 40,005 key positions; it is validated so that the edited layer maps $k^* \to v^*$ to relative error 10^{-7} with exact weight restoration; edit success 83.0%.

D Portable audit checklist

Table 7 restates the audit as four checks that apply to any benchmark generated from versioned structured data, with the evidence each check produces in this paper and the corresponding repair.

Risk	Diagnostic	Evidence here	Repair
Role erasure	Compare extracted facts with structured snaks	Role-specific error rates (Table 1)	Parse roles and ranks
Identity loss	Track identifiers through every stage	Label–ID mismatch rate (Table 2, row C)	Preserve canonical IDs
Temporal mismatch	Replay pinned revisions	Revision agreement (§3.3; Figure 3)	Store source timestamps
Lexical probe selection	Compare against semantic controls	Matched leakage effect (Table 4)	Relation-aware matching

Table 7: The audit as a portable checklist.

MUSART: A Scalable Multi-Domain Benchmark for Measuring Factual Recall in Large Language Models

Rajaa El Hamdani,[1] **Thomas Bonald**,[1] **Fragkiskos Malliaros**[2,3]
[1]Institut Polytechnique de Paris
[2]Université Paris-Saclay, CentraleSupélec, [3]Inria
rajaa.elhamdani@telecom-paris.fr, thomas.bonald@ip-paris.fr
fragkiskos.malliaros@centralesupelec.fr

Abstract

Reliable evaluation of factual recall in large language models (LLMs) requires ground truth that is broad, clean, and complete. Yet existing benchmarks often rely on narrow domains or incomplete knowledge bases that penalise models for generating correct facts absent from the reference. We introduce **MUSART**, a benchmark of 21,775 subject–relation questions over 11 domains, created via a schema-free extraction pipeline that can be cheaply regenerated from public dumps. To overcome knowledge-base incompleteness, we contribute a source-grounded augmentation pipeline that mines verifiable facts directly from article text. Evaluating twenty-one open-weight LLMs against a dual-reference pair (original vs. augmented) yields three findings: (i) completeness correction systematically raises precision and lowers recall across all models; (ii) factual recall varies widely across subject domains, demonstrating that single-domain evaluation confounds model capability with subject matter; and (iii) the precision gap between the two references separates instances where models lack knowledge from instances where the knowledge base lacks facts, proving that completeness-blind evaluation penalises both identically. All code, data, and evaluation scripts are publicly released at https://github.com/Rajjaa/MUSART.

1 Introduction

Evaluating large language models as knowledge bases (LM-as-KB) is fundamentally confounded by the references used to score them. Existing benchmarks suffer from two structural blind spots: narrow domain coverage, which locks in arbitrary difficulty baselines, and knowledge-base incompleteness, which penalises models for generating correct facts absent from the ground truth. For instance, when an LLM correctly names an ensemble cast member of a television series whom Wikipedia documents but Wikidata omits, standard evaluation records a false positive—marking the model wrong for being right.

We address both challenges with **MUSART** (**MU**lti-domain **SA**lient **R**elation **T**riples). Specifically, our contributions are threefold:

1. **A multi-domain factual benchmark:** We release 21,775 subject–relation questions across 11 domains, created via a schema-free extraction pipeline that can be cheaply regenerated from public dumps.

2. **A dual-reference evaluation paradigm:** To mitigate reference incompleteness, we contribute a source-grounded augmentation pipeline that mines verifiable facts directly from article text. We propose evaluating models against a conservative–optimistic pair of references to expose missing facts.

3. **A large-scale empirical study:** We evaluate twenty-one open-weight LLMs against this dual-reference setup, isolating factual recall from evaluation artefacts.

Our evaluation yields three principal findings:

1. **Completeness correction is systematic:** Moving from the original to the augmented ground truth raises precision and depresses recall in every single run, showing that reference incompleteness is a major, predictable confounder in factuality scoring (Section 5).

2. **Domain variance is large and shared:** Factual recall varies widely across subject domains (from 50.1% down to 10.4% mean precision), and models show strong agreement on which subjects are hard (mean pairwise Spearman $\rho = 0.68$). This suggests the variation reflects subject matter and the shared inductive biases of decoder-only Transformers

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 151–168
October 28, 2026 ©2026 Association for Computational Linguistics

rather than individual model behavior, demonstrating that single-domain evaluation cannot serve as a general study of parametric knowledge (Section 5.3).

3. **Correction separates model failure from KB failure:** The precision gap between Wikidata and article text varies dramatically across domains (from 14.9 to 0.6 points). This variance isolates where models lack knowledge versus where the knowledge base lacks facts, proving that completeness-blind evaluation penalises both identically (Section 5.3).

All code, data, question templates, and evaluation scripts are publicly available at `https://github.com/Rajjaa/MUSART`.

2 Related Work

Factual Recall and Knowledge Bases. The probing paradigm established by LAMA (Petroni et al., 2019) has scaled from single-token cloze tests to full generative benchmarks (Jiang et al., 2020; Elazar et al., 2021; Hu et al., 2024). Triple-aligned resources like T-REx (Elsahar et al., 2018) and KAMEL (Kalo and Fichtel, 2022) broadened relation coverage, while benchmarks like PopQA (Mallen et al., 2023), Head-to-Tail (Sun et al., 2024), and SimpleQA (Wei et al., 2024) select facts based on entity popularity. Popularity-based selection risks under-sampling domain-specific knowledge; MUSART instead seeds from thematic Wikipedia categories and ranks relations by graph salience to ensure broad multi-domain coverage.

Evaluation under Imperfect Ground Truth. Knowledge bases follow the open-world assumption, meaning absent triples generate false negatives during evaluation (Galárraga et al., 2013). Existing approaches handle this through statistical correction of label noise (Jain et al., 2016; Ramola et al., 2019) or calibrated-judge frameworks that estimate error rates (Feng et al., 2026). MUSART takes a different approach: rather than producing a single corrected point estimate, it evaluates models against a conservative–optimistic pair of references. Furthermore, while recent benchmarks increasingly rely on LLMs as judges (Min et al., 2023; Zheng et al., 2023) or embedding metrics (Zhang et al., 2020), MUSART holds the scorer fixed and deterministic via dense alias matching. This ensures that any score movement between references is attributable to knowledge-base completeness rather than grader instability. Finally, because MUSART is strictly single-hop, it cleanly isolates factual recall from the complex effects of test-time reasoning (Wei et al., 2022; Snell et al., 2024; Chen et al., 2025; Yao et al., 2025).

3 The MUSART Benchmark

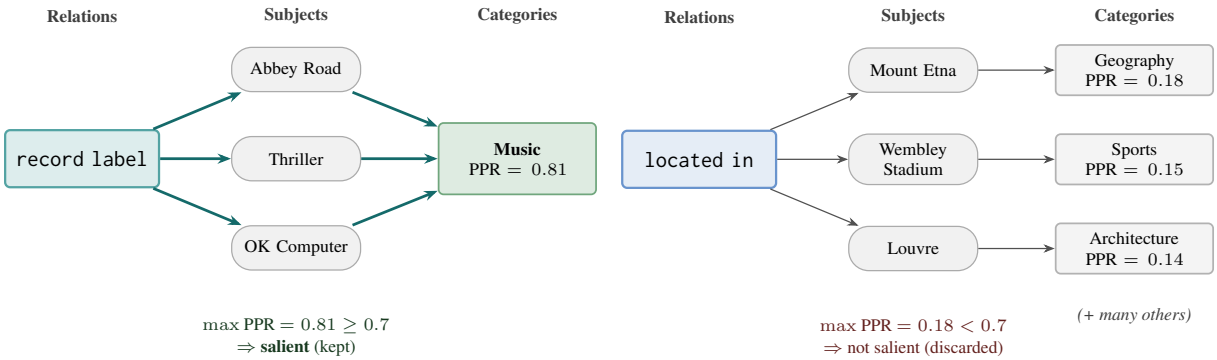
3.1 Triple Acquisition and Filtering

The initial phase of benchmark construction identifies high-quality source articles, links them to Wikidata, retrieves associated structured triples, and rigorously filters them before computing domain salience (see Appendix A for full technical details):

- **Source Article Selection:** We extract 38,279 English Wikipedia Good Articles (GAs) across 65 second-level domain categories (e.g., "Science and technology" → "Biology and medicine").
- **Wikidata Linking:** Using Wikimedia dump mappings, we align GA titles to Wikidata QIDs, excluding 11,353 unmapped articles to retain 26,926 linked entities, of which 26,210 yield statement triples in Wikidata.
- **Triple Acquisition:** We extract 302,179 truthy statement triples (s, r, o) across 1,001 relations from Wikidata dumps associated with these 26,210 subjects.
- **Data Filtering & Salience:** We remove 24 generic or administrative relations (Table 6 in Appendix D) along with ID properties, and apply a frequency threshold (≥ 50 occurrences) to prune 689 sparse relations. Subsequent Personalized PageRank salience selection (Section 3.2) is followed by manual domain selection and cardinality filtering to isolate the final benchmark of 44,722 triples across 8,407 subject–article pairs and 37 salient relations over 11 domains, grouped into 21,775 set-valued questions (Section 5).

3.2 Saliency Scoring

Salient Relation Identification. The core of our methodology is to automatically identify relations that are particularly salient or characteristic for each domain. The goal is to filter out generic relations that appear uniformly across many categories—such as "designed by" or "author"—and to retain relations that are specific to the semantic field of a category. For instance, an ideal system



(a) Domain-specific relation: PPR mass *concentrates* on one category.

(b) Generic relation: PPR mass is *spread* across many categories.

Figure 1: The graph-based salience model behind MUSART, contrasting the two regimes it must separate. A heterogeneous graph links each relation to the subjects in which it occurs and each subject to its Wikipedia category; a Personalized PageRank walk is initiated from each relation node. (a) For a *domain-specific* relation such as `record label`, the subjects overwhelmingly belong to one category, so the walk concentrates probability mass there (Music, 0.81) and the relation is retained as salient. (b) For a *generic* relation such as `located in`, the subjects are scattered across many domains, so the mass is diluted and no single category exceeds the global threshold (0.7); the relation is discarded. This is precisely the distinction that purely frequency-based scoring fails to make.

should identify that the relation "record label" is highly salient for music-related categories, but not for geography. Table 4 in Appendix A presents a set of manually curated examples that illustrate the desired quality and domain-specificity of the relations we aim to extract automatically.

Manually creating such lists for all domains is not scalable. Therefore, we developed an automated pipeline to approximate this level of quality. We first explored a simple frequency-based statistical approach, but found it had significant limitations, which led us to develop a more robust graph-based method using Personalized PageRank (PPR) for the final dataset construction.

Baseline: Frequency-based Saliency. We initially scored each relation r in category c by its in-domain frequency ratio: $\text{Saliency}(r, c) = \frac{\text{count}(r,c)}{\text{count}(r)}$. While an empirical threshold of 0.2 balanced specificity and coverage for certain categories, score distributions varied wildly across domains. Consequently, a single global threshold left several domains with zero salient relations, proving unworkable without non-scalable, per-category manual tuning.

Final Method: Personalized PageRank-based Saliency. To overcome the limitations of the frequency-based baseline, we employ a graph-based method using Personalized PageRank (PPR). The intuition is that relations disproportionately used within a specific set of categories will primarily link to subjects belonging to those categories. Therefore, by constructing a graph representing these connections (`Relation` $\rightarrow$ `Subject` $\rightarrow$ `Category`), a random walk starting from such a relation is expected to frequently visit the nodes representing its most relevant categories.

We construct a heterogeneous directed graph $G = (V, E)$ from the filtered Wikidata triples and their associated Wikipedia categories, where nodes represent subjects, relations, and second-level Wikipedia categories, and edges link subjects to their categories and relations to their subjects (see Appendix A for formal node-set and edge-set definitions). We run PPR from each unique relation r to compute importance scores across all category nodes. Sorting relations by this score yields a ranked list for each category, where an empirical threshold of 0.7 effectively separates highly relevant domain relations from peripheral ones. Figure 1 illustrates the intuition behind this graph-based salience model. Table 1 summarises the resulting dataset statistics before and after salience selection.

Qualitatively, PPR recovers domain-specific relations (e.g., `hymenium attachment` for *Biology and medicine*; `MPA film rating` for *Film*). Nine categories yield no salient relations—five of them sports-related—because their GA representation relies on shared generic properties, validating the method's ability to distinguish topical relevance from true domain salience. From the 41 Good-

	Initial	Final
Articles	38,279	8,407
Categories	65	11
Triples	302,179	44,722
Subjects	26,210	8,407
Relations	1,001	37
Questions	—	21,775

Table 1: MUSART statistics. *Initial* = after triple acquisition, before relation filtering and PPR salience selection; *Final* = the released benchmark, after salience selection, manual domain selection and cardinality filtering. Triples sharing a subject–relation pair are grouped into one set-valued question, giving 2.05 gold objects per question.

Articles categories retained by the saliency step, we manually selected a subset of domains for evaluation to limit the scope of the paper, as the full dataset would be too large to evaluate and study manually, yielding 11 domains after relation-frequency and cardinality filtering.

4 Ground-Truth Augmentation

4.1 Motivation: Knowledge-Base Incompleteness

Wikidata is not exhaustive, especially for high-cardinality relations (e.g. a film's full cast). An LLM may produce a correct fact that is absent from the ground truth and be wrongly penalised, systematically underestimating precision. We therefore enrich the Wikidata ground truth with facts mined from the GA article text.

4.2 Augmentation Pipeline

To enrich Wikidata without sacrificing transparency, we perform extractive question answering on the Good Article texts using `langextract` with Gemini 3.1 Flash Lite. We filter extractions to retain only verbatim, character-grounded quotes (Tier 1 candidates), mapping hyperlinked spans to QIDs via Wikimedia dumps and retaining unlinked surface forms after string deduplication (see Appendix B and Figure 5 for full technical specifications). Processing 21,662 article–relation pairs yields 69,208 grounded extractions (61.4% QID-linked), raising the mean gold objects per question from 2.05 to 4.13.

Reading the Two Ground Truths. This pipeline yields two complementary reference regimes: (1) the **original ground truth** (Wikidata-only), a *conservative* reference that penalises correct facts missing from Wikidata; and (2) the **augmented ground truth**, an *optimistic* reference that credits article-supported answers while inheriting unlinked noise and deduplication limits (detailed in Appendix B, Table 5). Together, they measure factual sensitivity to reference completeness rather than point estimates of precision.

5 Evaluating Large Language Models

5.1 Experimental Setup

The evaluation set comprises **21,775 questions** spanning eleven domains (3,373 single-valued, 18,402 multi-valued; 2.05 gold objects/question on average), converted from structured triples via relation-specific templates following the LM-as-KB paradigm (Petroni et al., 2019). Models are evaluated under five-shot prompting without access to source texts (testing parametric knowledge only) and scored using set-based *alias matching* (AM), where an object is recalled if and only if any of its surface forms is generated as an exact string match (substring matches are rejected). All metrics (precision, recall, and F1) take values in $[0, 1]$ and are reported as percentages (%), micro-averaged against both reference regimes; full template generation, serving, and scoring mechanics are detailed in Appendix G.

Models and Evaluation Scope. While the MUSART benchmark and its dual-reference evaluation paradigm are model-agnostic and apply equally to closed APIs, our empirical study focuses on twenty-one open-weight models, spanning 0.8B to 70B parameters across four families (Qwen3.5, Gemma-2/Gemma-3/Gemma-4, Llama-3, and Mistral/Ministral). Using open-weight models ensures reproducibility and execution without proprietary API drift or unannounced backend modifications. The slate systematically varies three usually confounded factors: *scale* within a family and generation (e.g., Llama-3 from 1B to 70B), *generation* at matched size (Gemma-2, 3, and 4 across 2B–9B scales), and architectural *family* at matched scale in the 7–12B band (see Appendix G for full slate details). While main results use direct-answer configurations supported across all twenty-one models, thinking modes for the Qwen3.5 and Gemma-4 series are evaluated separately in Appendix F.

5.2 Completeness Correction: Precision Up, Recall Down

Evaluating against the augmented (vs. original) ground truth raises precision but lowers recall (Table 3). For `Llama-3.3 70B`, precision rises from 39.71% on the original ground truth to 46.47% on the augmented one, while recall falls from 52.47% to 30.53%. The direction is uniform: precision rises and recall falls in every row of Table 3. Figure 6 shows the resulting spread for every model; its width is how far that model's measured precision moves when the reference changes. Precision is the right cross-ground-truth comparator; recall and F1 fall mechanically because the augmented reference roughly doubles the gold objects to enumerate while models emit roughly the same number of answers. Because the augmented ground truth is un-audited and admits extraction and linking noise (Section 4), this rise is *consistent with*—but does not by itself establish—genuine underestimation of model precision by the original benchmark: part of the gain may come from answers that match noisy unlinked spans. The recall drop, by contrast, reflects the larger object count of the augmented set (2.05 to 4.13 gold objects per question; Section 4), including unlinked objects that may duplicate existing ones, and exposes a long-tail gap between what models produce and what the source text supports.

Where the reference gap is widest. Splitting the evaluation set by answer cardinality localises both effects (Table 2). Single-valued questions (3,373) are easier than multi-valued ones (18,402) under either reference—averaged over the slate, F1 is 5.99 points higher on the original ground truth and 7.59 points higher on the augmented one—but the interesting quantity is how augmentation moves each subset. Precision gains are more than three times larger on multi-valued questions ($+3.04$ points on average, against $+0.90$ for single-valued), and the recall penalty is also larger there (-12.81 against -8.97). Both movements point the same way: high-cardinality relations are where Wikidata is most incomplete, so they are simultaneously where the original benchmark most under-credits correct answers and where augmentation adds the most gold objects to enumerate. The reference gap is therefore not uniform; it is widest exactly where the knowledge base is thinnest, which is also where a completeness-blind evaluation is most misleading.

		Single val.	Multiple val.
Count		3,373	18,402
Precision	original	33.40	28.63
	augmented	34.30	31.67
	Δ	+0.90	+3.04
Recall	original	35.21	30.09
	augmented	26.24	17.27
	Δ	−8.97	−12.81

Table 2: The reference gap by answer cardinality: alias-match precision and recall (%) against each ground truth, averaged over the twenty-one models. Both effects of completeness correction are larger on multi-valued questions—the precision gain is more than three times larger, and the recall penalty is also steeper—reflecting where a knowledge base is most likely to be missing objects.

5.3 Factual Recall Varies Widely Across Subject Domains

MUSART's eleven evaluated domains reveal that measured factual recall is highly sensitive to the subject matter being queried. On the original reference, mean precision varies from 50.1% for *Physics and astronomy* down to 10.4% for *Transport*, and a similarly wide dispersion holds under the augmented one. We analyse precision rather than F1 throughout this section: the augmented reference roughly doubles the gold objects per question, so recall and F1 fall mechanically, whereas a precision gain means an answer the model actually produced is now credited (Section 5.2). This wide cross-domain variance is an empirical observation and a consequence of how pre-training data density and knowledge-base coverage differ by topic. Because baseline difficulty varies so dramatically from subject to subject, evaluating factuality on a single domain locks in an arbitrary difficulty level that cannot serve as a general measure of a model's factual recall.

This domain profile is a shared property across architectures, not a product of individual model behavior. Figure 2 shows every model–domain cell under both ground truths: the columns are visibly banded, and the domain rankings of any two models agree at mean pairwise Spearman $\rho = 0.67$ on the original ground truth (0.62 on the augmented). Models largely find the same subjects easy and the same subjects hard, which points to the composition of pre-training data and of Wikidata coverage rather than to model-specific gaps. The exceptions are legible in the same figure and are informative:

	Original GT			Augmented GT		
Model	P	R	F1	P	R	F1
Qwen3.5 0.8B	16.90	8.75	11.53	18.68	4.81	7.65
Qwen3.5 2B	21.42	15.16	17.76	23.63	8.32	12.30
Qwen3.5 4B	22.54	30.91	26.07	25.06	17.09	20.32
Qwen3.5 9B	26.53	35.42	30.34	29.33	19.48	23.41
Gemma-3 1B	15.13	12.19	13.50	16.57	6.64	9.48
Gemma-3 4B	24.15	29.64	26.61	26.79	16.35	20.30
Gemma-3 12B	34.62	36.11	35.35	38.63	20.04	26.39
Gemma-3 27B	31.50	43.83	36.65	36.49	25.25	29.85
Gemma-4 E2B	25.83	18.04	21.24	27.66	9.61	14.26
Gemma-4 E4B	30.75	24.12	27.04	34.01	13.26	19.08
Gemma-4 12B	30.12	34.70	32.25	33.74	19.33	24.58
Gemma-4 31B	**42.46**	40.01	41.20	**47.63**	22.32	30.40
Gemma-2 2B	29.85	27.84	28.81	33.31	15.45	21.11
Gemma-2 9B	35.01	39.05	36.92	39.34	21.82	28.07
Llama-3.2 1B	15.04	13.51	14.24	16.22	7.25	10.02
Llama-3.2 3B	28.11	29.81	28.93	31.00	16.35	21.41
Llama-3.1 8B	34.19	38.75	36.32	38.61	21.76	27.84
Llama-3.3 70B	39.71	**52.47**	**45.21**	46.47	**30.53**	**36.85**
Ministral-8B	25.46	29.68	27.41	28.18	16.34	20.68
Mistral-7B v0.3	24.00	35.65	28.69	27.18	20.07	23.09
Mistral-Nemo	32.27	32.86	32.56	35.97	18.21	24.18

Table 3: Main results (alias-match micro-averages, %; few-shot, direct-answer). Note: F1 is comparable only *within* each ground truth; compare precision across ground truths.

`Ministral-8B` ranks 11th of 21 overall but first on *Physics and astronomy*, where `Mistral-Nemo` is second, while `Llama-3.3 70B`—first overall—places ninth in that domain, and `Gemma-3 27B` falls from fourth overall to eleventh. On a small domain (113 questions) this is an observation rather than a result, but it is the kind of pre-training signature that a single-domain benchmark cannot expose at all.

Where the knowledge base is incomplete. Comparing references per domain estimates Wikidata incompleteness as models experience it (Figure 7 in Appendix C). This gap varies widely across measured domains (Figure 3), ranging from 14.9 points in *Architecture* down to 0.6 in *Motorsport* (where functional relations like `pole position` admit a single correct answer) and 0.5 in *Biology and medicine*—a thirty-fold spread across the ten domains where augmentation ran. Consequently, knowledge-base incompleteness is highly domain-dependent, and a single global completeness correction would be inaccurate in both directions.

What predicts a wide gap is less clear than we expected. Across the ten domains where augmentation ran, the gap correlates most with the proportion of multi-valued questions ($r = 0.57$), then with the share of extracted mentions that hyperlink matching resolves to QIDs ($r = 0.47$), then with the factor by which augmentation multiplies the gold-object count ($r = 0.35$). With ten points none of these is decisive, and we report them as indicative rather than as an attribution.

Precision gain and recall loss are almost unrelated across domains ($r = 0.11$), and that dissociation is informative: they measure different incompletenesses. *Architecture* and *Biology and medicine* lose comparable recall (-13.4 and -15.9) but gain wildly different precision ($+14.9$ against $+0.5$). In *Architecture* the article text supplies objects that models already produce and Wikidata simply lacked; in *Biology and medicine* augmentation adds many objects, but ones the models do not know. The extreme case is *Battles, exercises, and conflicts*, where augmentation raises the gold set from 2.5 to 12.0 objects per question—a $4.8\times$ multiplication that produces both a wide gap and the largest recall penalty on the benchmark. The first pattern is a deficiency of the knowledge base, the second a deficiency of the models, and a completeness-blind evaluation reports both as the same low score.

Rank stability. Despite these shifts in level, the domain *ordering* is robust to the choice of ground truth. Computing the correlation separately for each model rather than aggregating (Figure 2, right), per-model Spearman ρ between the original and augmented domain rankings is positive and high throughout: 0.98 for `Qwen3.5 0.8B` at the top and 0.76 for `Ministral-8B` at the bottom, with a mean of 0.89. No model falls below 0.75. Since one of the two references is known to carry extraction and linking noise, agreement this high indicates that the domain profile reflects genuine variation in stored knowledge rather than an artefact of missing Wikidata entries.

5.4 Ranking Instability

Changing the reference also changes the leaderboard, though modestly. Between the original and augmented ground truths, seven of the twenty-one models change rank, and the largest move is two places (`Gemma-4 E4B`). The top of the table is stable—`Llama-3.3 70B` leads under either reference—and so is the bottom; the movement is mid-table, which is where leaderboard comparisons are usually contested and where the margins are thinnest. We note the effect but do not lean on it: a two-place shift on this slate corresponds to

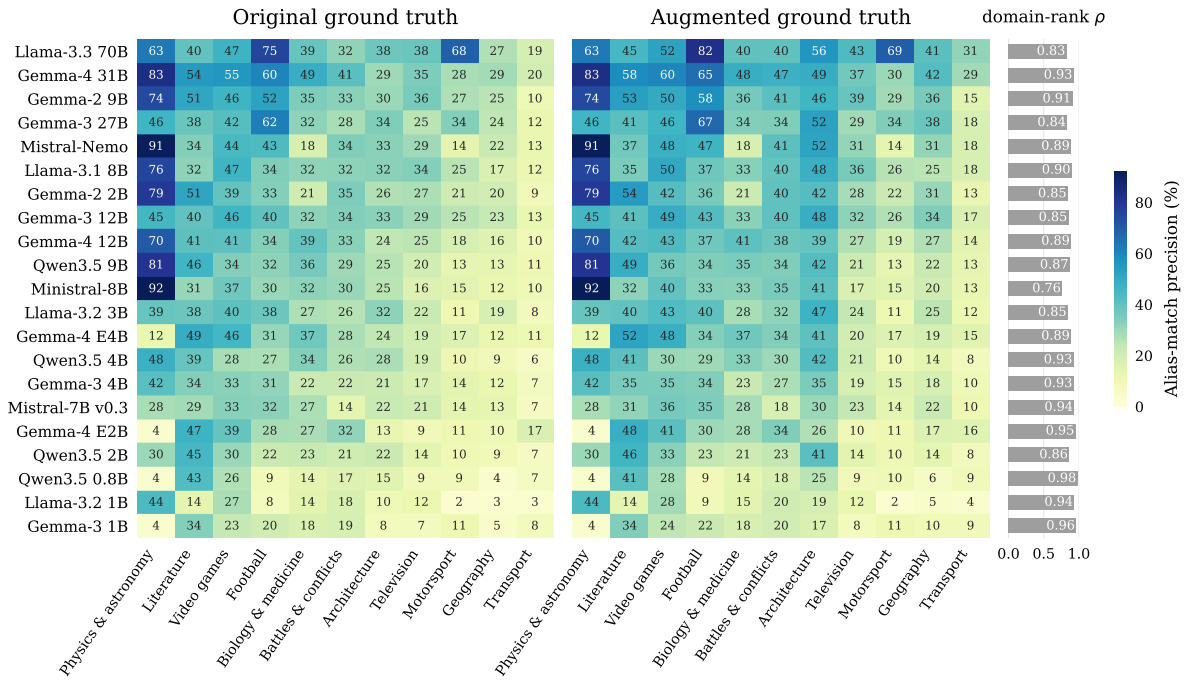


Figure 2: Alias-match precision for every model–domain pair under both references, with models and domains ordered by their original-reference means. Precision is shown because it is the quantity comparable across references: an extracted object the model also produced is very likely a genuine fact the knowledge base was missing, whereas recall falls by construction as the gold set grows. Right: each model's Spearman correlation between its original and augmented domain rankings. Columns are visibly banded across subjects—reflecting how baseline difficulty varies by domain—and the per-model ρ values show the domain ordering survives the change of reference for every model on the slate.

differences of one to two F1 points, and without per-item confidence intervals we cannot separate a genuine reordering from noise.

Neither parameter count nor model recency guarantees superior recall. On the original ground truth, `Llama-3.1 8B` (36.32 F1) outperforms the larger `Mistral-Nemo` (32.56), a gap that persists on the augmented reference (27.84 against 24.18). Newer architectural generations do not reliably outperform their predecessors either: `Gemma-2 9B` (36.92) beats both `Gemma-3 12B` (35.35) and `Gemma-4 12B` (32.25) despite being an older and smaller model. This variability suggests that differences in pretraining data curation and post-training alignment protocols may exert a stronger influence on factual recall than parameter count or release date alone.

A second protocol effect. Reference completeness is not the only protocol choice that moves a factuality score without the model changing. Evaluating thinking configurations exposes another: when a model interleaves a reasoning trace with its answer, the evaluator must decide what counts as the answer, and that decision alone moves alias-match F1 by between 3 and 22 points for the Qwen3.5 series—larger than the spread the same table attributes to model scale. We report those results in Appendix F rather than here, because they characterise decoding behaviour rather than factual recall; the point they contribute is that answer extraction belongs alongside reference completeness on the list of protocol choices a factuality benchmark must pin down before its numbers mean anything.

6 Discussion

MUSART shows that scalable, multi-domain ground truth can be built from categorical structure plus graph salience, and that source-grounded augmentation, read as a conservative and an optimistic reference, supports completeness-aware claims without requiring a manually audited ground truth. A more complete benchmark yields a more humbling assessment: measured precision rises while measured recall falls sharply, suggesting models know a smaller fraction of the available facts than standard evaluation implies.

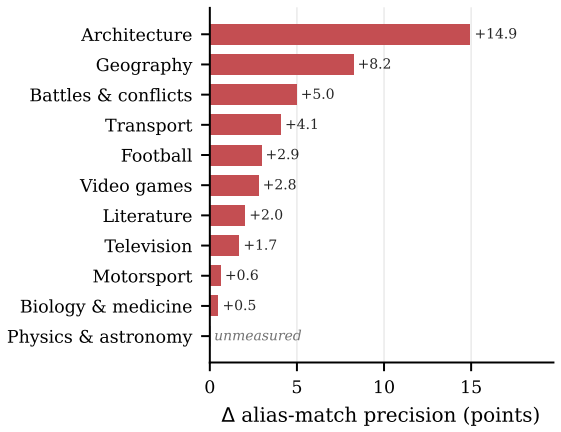


Figure 3: The correction gap by domain: augmented minus original alias-match precision, averaged over the twenty-one models. Wikidata's incompleteness is not a constant that a single global adjustment could absorb—it is worth 14.9 points of measured precision in *Architecture* and 0.5 in *Biology and medicine*, a thirty-fold range. Crucially, this ranking measures knowledge-base deficiency, not domain difficulty: *Architecture* tops the chart because article text supplies objects models already produce but Wikidata lacked, whereas *Biology and medicine* sits at the bottom because its heavy augmentation adds objects the models do not know. *Physics and astronomy* is shown as unmeasured rather than as a zero: the extractor produced no output for that domain (Section 4).

In practice we recommend reporting against *both* ground truths, never the augmented one alone: a single augmented-GT figure launders extraction noise into apparent fact, whereas the pair exposes how far the score moves with the reference, and the size of that movement is itself informative—a wide gap flags domains or relations where Wikidata coverage is thin and completeness correction matters most. Cross-ground-truth comparisons should be made on precision; recall and F1 comparisons are meaningful only within one ground truth.

Furthermore, because all evaluated models share the same fundamental decoder-only Transformer architecture, their highly correlated domain profiles likely reflect shared structural properties and inductive biases rather than merely overlapping pre-training corpora. We suggest that future work could use mechanistic interpretability tools to investigate how the Transformer's specific mechanisms for storing and retrieving relational facts give rise to this shared knowledge profile, and whether alternative architectures exhibit different factual recall patterns.

7 Conclusion

We presented a construction method for multi-domain factual benchmarks that needs no per-domain relation curation, MUSART as its instantiation on Wikipedia and Wikidata, a source-grounded augmentation pipeline whose paired references expose how far a factuality score moves with reference completeness, and a study of twenty-one open-weight LLMs that separates evaluation artefacts from genuine knowledge.

Three directions follow directly. A calibrated audit of the augmented ground truth would convert its reference gap from a sensitivity measurement into an estimate with guarantees. A second prompting condition would establish how much of what we report is specific to forced answering. And applying the construction method to a different corpus–knowledge-base pair would test the claim that its domain-discovery step is genuinely portable—the part of this work we expect to outlive the particular benchmark it produced.

Limitations

The augmented ground truth is un-audited under the pipeline configuration reported here (`gemini-3.1-flash-lite` extraction with hyperlink matching and string deduplication): it contains noise from extraction, and from unlinked single-surface-form objects that cannot be de-duplicated against Wikidata. Consequently, figures computed against the original and augmented ground truths should be read as two contrasting reference regimes rather than as point estimates, and a calibrated audit of the augmented ground truth is left to future work. Inferred (non-grounded) extractions are discarded, trading recall for precision. All results use a single prompting condition (few-shot), which narrows the generality of the findings; prompting sensitivity is left to future work. Domain-level estimates inherit the size of the domain: *Physics and astronomy* contributes only 113 questions and *Motorsport* 326, so their per-domain figures rest on thin samples and should not be over-read. MUSART is built from English Wikipedia and English question templates, so all findings concern English factual recall. Because Wikipedia is present in essentially every open-weight pre-training corpus, MUSART measures recall of likely-seen facts—the quantity of interest for the LM-as-KB setting—rather than generalisation to unseen knowledge. Finally, results derive from a fixed model set, and absolute num-

bers will change as newer open-weight LLMs are added.

Reproducibility

MUSART is regenerated end to end from public Wikipedia and Wikidata dumps by the pipeline of Section 3; the augmentation of Section 4 uses `langextract` with a pinned `gemini-3.1-flash-lite` version. All generation is greedy, so runs are deterministic given a fixed serving stack. All code, data, question templates, few-shot exemplars, scoring scripts, and the per-run summary statistics behind Table 3, Figure 2, and Figures 6 and 7 in Appendix C are publicly released at `https://anonymous.4open.science/r/MUSART/`.

Ethics Statement

MUSART is built from publicly available Wikipedia content (CC BY-SA 4.0) and Wikidata content (CC0), and will be released under a compatible license with attribution to the source projects. The benchmark concerns factual recall about encyclopaedic entities and poses no foreseeable harm beyond standard considerations for automatically constructed datasets.

References

Thomas Bonald, Nathan de Lara, Quentin Lutz, and Bertrand Charpentier. 2020. Scikit-network: Graph analysis in python. *Journal of Machine Learning Research*, 21(185):1–6.

Xilun Chen, Ilia Kulikov, Vincent-Pierre Berges, Barlas Oğuz, Rulin Shao, Gargi Ghosh, Jason Weston, and Wen-tau Yih. 2025. Learning to reason for factuality. *Preprint*, arXiv:2508.05618.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Chen Feng, Minghe Shen, Ananth Balashankar, Carsten Gerner-Beuerle, and Miguel R. D. Rodrigues. 2026. Noisy but valid: Robust statistical evaluation of llms with imperfect judges. *Preprint*, arXiv:2601.20913.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 413–422.

Yujia Hu, Shrestha Ghosh, Tuan-Phong Nguyen, and Simon Razniewski. 2024. GPTKB: Building very large knowledge bases from language models. *Preprint*, arXiv:2411.04920.

Shantanu Jain, Martha White, and Predrag Radivojac. 2016. Estimating the class prior and posterior from noisy positives and unlabeled data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Jan-Christoph Kalo and Leandra Fichtel. 2022. KAMEL: Knowledge analysis with multitoken entities in language models. In *4th Conference on Automated Knowledge Base Construction (AKBC)*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 12076–12100.

Fabio Petroni, Tim Rocktäschel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Rashika Ramola, Shantanu Jain, and Predrag Radivojac. 2019. Estimating classification accuracy in positive-unlabeled learning: Characterization and correction strategies. In *Pacific Symposium on Biocomputing (PSB)*, volume 24, pages 124–135.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *Preprint*, arXiv:2408.03314.

Kai Sun, Yifan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-tail: How knowledgeable are large language models (llms)? a.k.a. will llms replace knowledge graphs? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 311–325.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *Preprint*, arXiv:2411.04368.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pages 24824–24837.

Zijun Yao, Yantao Liu, Yanxu Chen, Jianhui Chen, Junfeng Fang, Lei Hou, Juanzi Li, and Tat-Seng Chua. 2025. Are reasoning models more prone to hallucination? *Preprint*, arXiv:2505.23646.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR)*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023) Datasets and Benchmarks Track*.

Appendix

A Construction Pipeline and Graph-Based Salience Formulation

Figure 4 illustrates the automated MUSART benchmark construction pipeline, which seeds from Wikipedia Good Article categories and aligns to Wikidata without manual per-domain schema curation.

Granular Triple Acquisition and Filtering Steps. To supplement the summary in Section 3.1, the initial triple collection proceeds as follows. First, we select articles designated as "Good Articles" (GAs) on the English Wikipedia, which constitute approximately 0.6% of total articles and satisfy explicit editorial criteria for factual accuracy, neutrality, verifiability, and stability. From the "Wikipedia:Good articles/all" page, we harvest 38,279 GA titles organised under 65 second-level domain categories. Second, using Wikimedia dump mapping tables, we link each GA title to its Wikidata QID; 11,353 articles lacking a QID are discarded to retain 26,926 linked entities, of which 26,210 yield statement triples in Wikidata. Third, from the Wikidata dump of truthy triples, we harvest 302,179 statement triples (s, r, o) across 1,001 relations where s is a linked GA QID. Finally, we apply two-stage filtering: (1) removing 24 generic, administrative, or Wikimedia-internal properties (e.g., `instance of`; see Table 6) and all ID-bearing properties, and (2) thresholding relation frequency at ≥ 50 occurrences across the corpus (removing 689 sparse relations whose median frequency is 13) so that subsequent salience scoring operates on representative properties.

Domain	Salient Relations
Biology & med.	hymenium attachment, IUCN conservation status, mushroom ecological type, has basionym, has cause, health specialty, studied in, mushroom cap shape, stipe character, edibility, spore print color, subject has role, original combination, diel cycle, hymenium type
Geography	located in or next to body of water, mountain range, continent
Places	historic county, shares border with, located in time zone, located in or next to body of water, capital of, twinned administrative body
Transport	terminus location, state of use, connects with, country, connecting service, maintained by, transport network, connecting line, located in the administrative territorial entity, adjacent station, owned by, terminus

Table 4: Examples of manually selected salient relations for four Good Articles categories, illustrating the desired output of our automated methodology.

Table 4 presents examples of manually curated salient relations across four representative domains, illustrating the target domain-specificity that our automated graph-based salience pipeline approximates.

Formal Mathematical Formulation of Graph-Based Saliency. To formalise the Personalized PageRank (PPR) salience selection introduced in Section 3, we construct a heterogeneous directed graph $G = (V, E)$ from the filtered Wikidata triples and their associated Wikipedia data. The node set V is the disjoint union of:

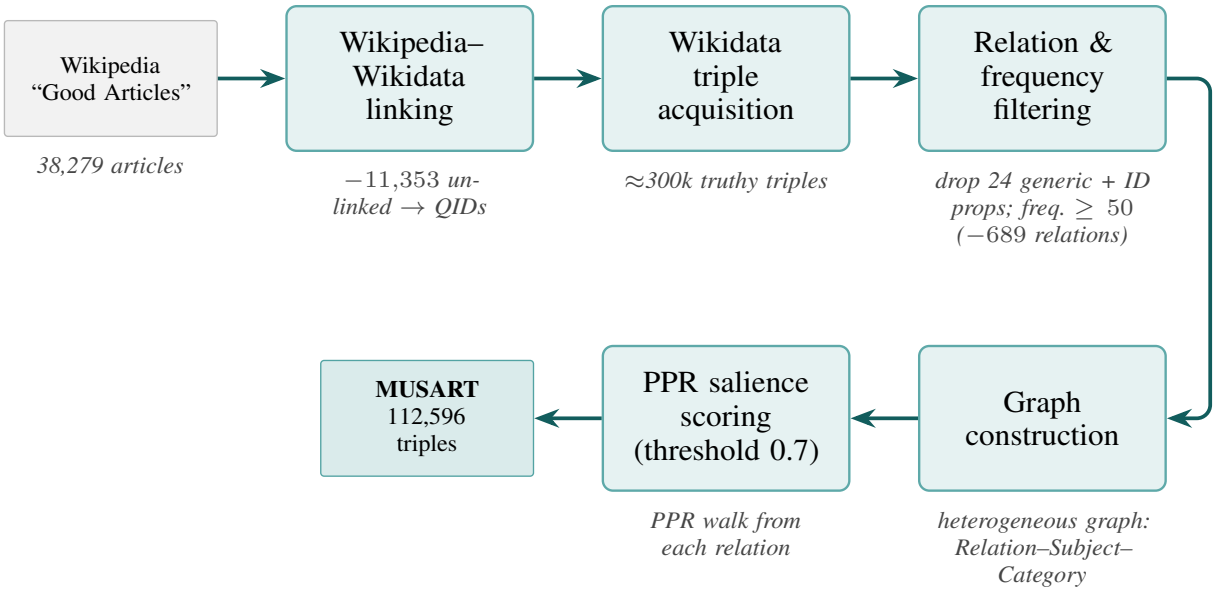


Figure 4: Construction of MUSART. An automated pipeline links high-quality Wikipedia "Good Articles" to Wikidata, acquires their triples, removes generic and rare relations, and scores the remainder with a Personalized PageRank salience model, distilling ≈300k raw triples and 1,001 relations down to 112,596 triples over 173 salient relations across 31 domains.

- Subject nodes V_S representing Wikidata subjects;
- Property nodes V_R representing Wikidata relations;
- Category nodes V_C corresponding to second-level Wikipedia categories extracted from the Good Article hierarchy.

Two typed edge sets are defined:

- $E_{SC} = \{(s, c) \mid s \in V_S, c \in V_C, s \text{ is labelled with } c\}$, linking each subject to its category;
- $E_{RS} = \{(r, s) \mid r \in V_R, s \in V_S, (s, r, \cdot) \text{ appears in the filtered triples}\}$, linking every relation to the subjects in which it occurs.

The complete edge set is $E = E_{SC} \cup E_{RS}$. We use the `scikit-network` library (Bonald et al., 2020) to represent the graph and compute the Personalized PageRank. We run PPR initiated from each unique relation node $r \in V_R$. The PageRank score obtained by a category node $c \in V_C$ during this random walk reflects the structural strength of the connection between relation r and category c. Sorting relations by this score yields a ranked list per category, where an empirical cutoff of 0.7 isolates salient domain relations.

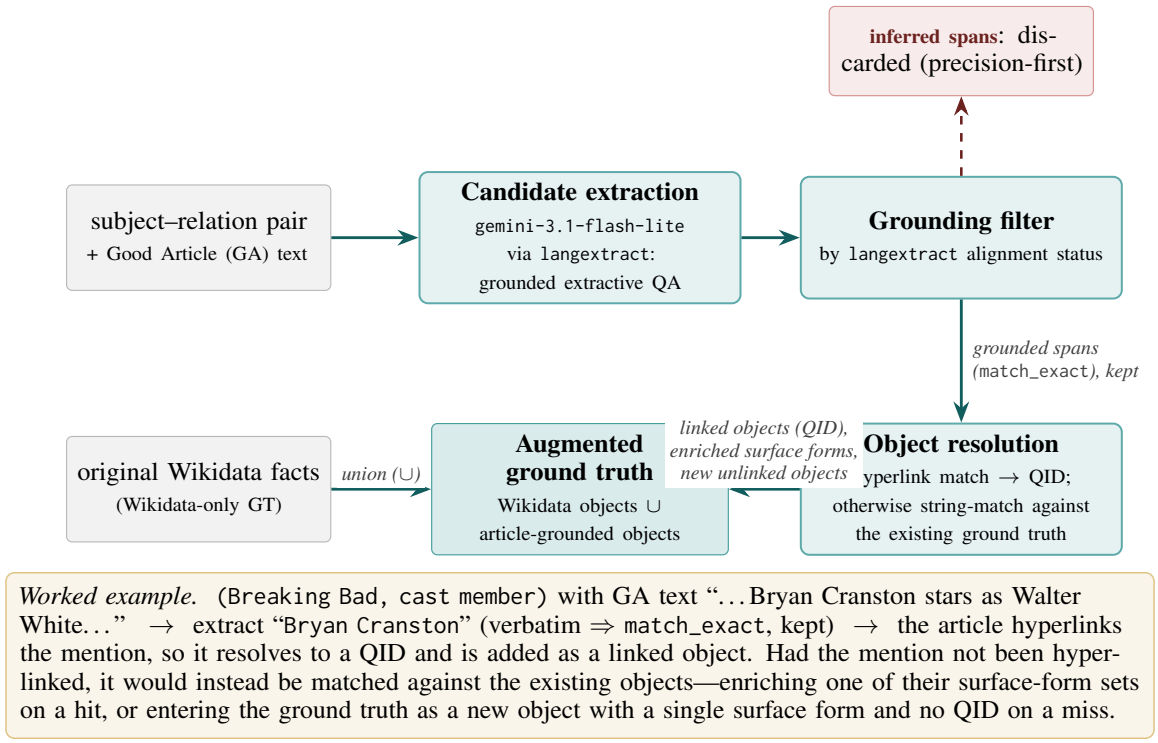
B Ground-Truth Augmentation Pipeline Details and Domain Statistics

Figure 5 depicts the ground-truth augmentation pipeline introduced in Section 4, which enriches Wikidata triples with character-grounded facts extracted verbatim from Wikipedia Good Article texts.

Granular Extraction and Linking Methodology. The augmentation pipeline operates via three transparent stages:

1. **Candidate Extraction:** For each subject–relation pair, we perform extractive question answering on the corresponding Good Article text using Gemini 3.1 Flash Lite via the `langextract` library.[1] A key feature of `langextract` is its **precise source grounding**, mapping every extraction to its exact character interval in the source document.
2. **Candidate Tiering:** Extractions are categorised into two tiers based on alignment metadata. **Tier 1 (Grounded)** candidates are those returning an `alignment_status` of `match_exact`, confirming that the extracted string is a direct, verbatim quote from the text. **Tier 2 (Inferred)** candidates lack exact character grounding and rely on model inference;

[1] https://github.com/google/langextract



Worked example. (Breaking Bad, cast member) with GA text "…Bryan Cranston stars as Walter White…" → extract "Bryan Cranston" (verbatim ⇒ match_exact, kept) → the article hyperlinks the mention, so it resolves to a QID and is added as a linked object. Had the mention not been hyperlinked, it would instead be matched against the existing objects—enriching one of their surface-form sets on a hit, or entering the ground truth as a new object with a single surface form and no QID on a miss.

Figure 5: Augmenting the MUSART ground truth. An automated, precision-first pipeline extracts candidate objects from "Good Article" text, keeps only verbatim-grounded candidates, and resolves them against Wikidata. Hyperlink matching is the only step that assigns QIDs; remaining grounded spans are string-matched against the existing ground truth, either enriching the surface forms of an object already present or entering as a new object carrying one surface form and no QID. The result is the augmented ground truth used throughout Section 5; it is not audited, and the consequences of that are set out in Section 4.

to prioritise precision, Tier 2 extractions are discarded.

3. **Entity Linking and Deduplication:** To resolve extracted spans to canonical Wikidata entity IDs (QIDs), we rely exclusively on hyperlink matching. We parse the source article HTML to determine if an extracted span coincides with an anchor hyperlink; if hyperlinked, we map the target page title to its QID via Wikimedia dumps. Unlinked extractions are retained as surface forms without external parametric linking models. To avoid duplicates, unlinked strings are matched against existing Wikidata labels and aliases for that subject–relation pair: matching strings are merged into the existing Wikidata entity, while non-matching strings are retained as newly discovered, unlinked object entities.

Domain Distributions and Evaluation Properties of Unlinked Objects. As reported in Table 5, augmentation mass varies widely across domains: it multiplies the gold-object count by $4.8\times$ in *Battles, exercises, and conflicts* but by $1.14\times$ in *Motorsport*, with linking rates ranging from 70% (*Geography*) to 16% (*Biology and medicine*). Ratio and absolute mass diverge: *Video game titles* has a modest $1.8\times$ multiplier but contributes 14,157 of the 45,205 objects added by augmentation. Extraction ran at full coverage on ten of the eleven evaluated domains; *Physics and astronomy* (113 questions, drawn entirely from `epoch`) produced no extraction output and is treated as unmeasured rather than measured-as-complete in Section 5.3. Furthermore, two properties of unlinked objects affect evaluation: (1) an unlinked object carrying a single surface form and no alias set can only be matched by that exact string, depressing mean surface forms per object (Table 5); and (2) lacking a QID, an unlinked object cannot be de-duplicated against existing Wikidata entities, which can slightly inflate object counts and depress measured recall while introducing a minor false-positive risk for noisy spans.

Table 5 reports the per-domain impact of this augmentation pipeline across all eleven evaluated MUSART domains.

	Objects/question			
Domain	orig.	aug.	×	% link
Battles & conflicts	2.52	12.01	4.8	37.9
Transport	1.42	3.88	2.7	68.1
Geography	1.18	3.17	2.7	70.0
Architecture	1.22	3.08	2.5	66.3
Biology & medicine	1.54	3.51	2.3	16.1
Literature	1.14	2.23	1.9	43.7
Video game titles	2.18	3.91	1.8	41.5
Television	3.36	5.77	1.7	61.3
Football	1.55	2.24	1.4	53.9
Motorsport	1.00	1.14	1.1	61.7
Physics & astronomy	1.00	1.00	1.0	—

Table 5: Augmentation mass by domain: gold objects per question before and after augmentation, the resulting multiplier, and the share of extracted mentions resolved to QIDs by hyperlink matching. *Physics and astronomy* yielded no retained extractions.

C Additional Evaluation Figures

This appendix provides supplementary visual analyses of model evaluation across reference ground truths and subject domains.

Figure 6 illustrates the reference sensitivity of measured alias-match precision for each evaluated open-weight LLM, comparing scores against the conservative Wikidata-only reference and the optimistic augmented reference.

Figure 7 displays per-domain precision and recall under both ground truths, illustrating where Wikidata is incomplete relative to model outputs (the precision gap) versus how much article text adds that models do not recall (the recall drop).

D Filtered Generic Relations

Table 6 lists the 24 relations excluded from our analysis. These encode Wikidata/Wikimedia bookkeeping or organisational structure (e.g. `instance of`, `subclass of`, `topic's main category`) rather than factual statements about the entity itself, and therefore do not constitute meaningful targets for factual-knowledge evaluation.

E Salient Relations by Domain

Below, we list the relations identified as salient by the PPR selection procedure (Section 3.2) for a broader intermediate set of 50 categories processed during benchmark construction. (As noted in Section 3.1, manual selection and cardinality filtering yielded the final 11 domains included in the released dataset). Categories for which no relation passed the salience threshold are listed at the end.

Agriculture, food, and drink. `this taxon is source of`, `product`, `material`, `or service produced or provided`, `taxon range`

Albums. `tracklist`, `place of publication`, `charted in`, `recorded at studio or venue`, `record label`, `form of creative work`, `performer`, `producer`, `language of`

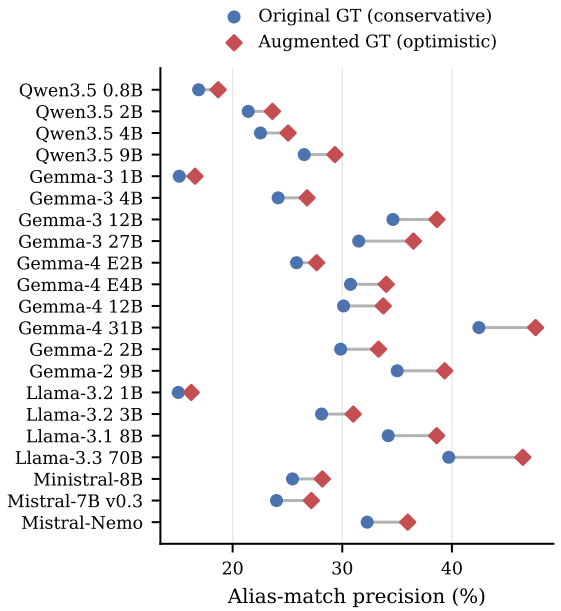


Figure 6: Reference sensitivity of measured precision. For each model, alias-match precision against the original (Wikidata-only) ground truth—a conservative reference—and against the unfiltered augmented ground truth—an optimistic one. The connecting segment is how far the score moves when only the reference changes; it is neither an improvement nor an interval containing the true value.

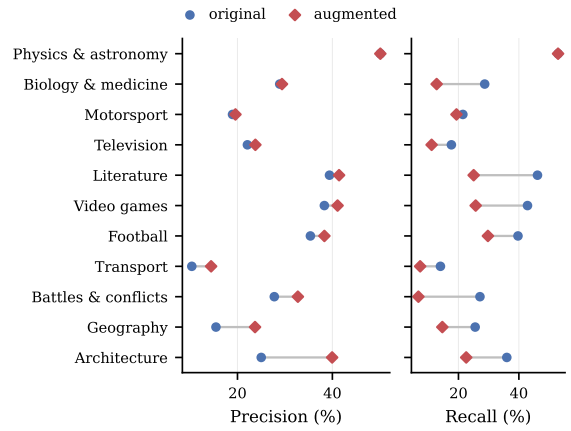


Figure 7: Per-domain alias-match precision and recall under both ground truths, averaged over the twenty-one models and ordered by the size of the precision gap. That gap measures where Wikidata is incomplete relative to what models produce; the recall drop measures how much the article text adds that models do not produce. The two orderings differ.

work or name, distribution format, cover art by, has characteristic, genre, followed by, follows

Architecture. diocese, located on street, architectural style, architect, dedicated to, occupant, heritage designation, has facility, commissioned by, made from material, crosses, historic county, located in the administrative territorial entity, has use

Armies and military units. military branch, parent organization, has subsidiary, allegiance, item operated

Art. collection, depicts, movement, made from material, creator, copyright status as a creator

Battles, exercises, and conflicts. participant, location, part of

Biology and medicine. health specialty, diel cycle, original combination, CITES Appendix, hymenium type, mushroom cap shape, hymenium attachment, stipe character, spore print color, mushroom ecological type, edibility, NZTCS conservation status, IUCN conservation status, endemic to, taxonomic type, has basionym, parent taxon, taxon rank, taxon synonym, taxon range, this taxon is source of, has cause, location of discovery, studied in, subject has role, has effect, opposite of, practiced by, has use, described by source, said to be the same as, uses

Chemistry and materials science. location of discovery, discoverer or inventor, has effect, subject has role

Compositions. movement, has characteristic, tonality

Computing and engineering. copyright license, operating system, product, material, or service produced or provided

Cue sports. sponsor, organizer, winner, sports season of league or competition

Earth science. located in/on physical feature, time period, part of the series

Economics and business. industry, product, material, or service produced or provided, legal form, headquarters location, founded by, has subsidiary

Education. category for alumni of educational institution, has subsidiary, language used, member of, parent organization

Film. RTC film rating, ICAA rating, CNC film rating (Romania), CNC film rating (France), EIRIN film rating, MPA film rating, Medierådet rating, JMK film rating, production designer, costume designer, RCQ classification, assessment, Filmiroda rating, FSK film rating, film editor, after a work by, director of photography, IFCO rating, ClassInd rating, NMHH film rating, Kijkwijzer rating, filming location, BBFC rating, IMDA rating, production company, fabrication method, executive producer, soundtrack release, cast member, color, nominated for, screenwriter, set in period, director, original language of film or TV show, composer, based on, narrative location, derivative work, main subject, distributed by, Australian Classification, producer, award received, inspired by, set during recurring event, set in environment, collection, said to be the same as, copyright status, has edition or translation, country of origin, distribution format, characters, genre, voice actor, depicts

Football. referee, head coach, season of club or team, home venue, league or competition, category for members of a team, participating team, competition class, sports season of league or competition, sport, organizer, time period, chairperson, occupant, winner, is a list of, owner of, history of topic, headquarters location

Relation
instance of
topic's main category
subclass of
KML file
sex or gender
on focus list of Wikimedia project
topic's main template
aspect ratio (W:H)
religion or worldview
facet of
maintained by WikiProject
category of associated people
category for people born here
category for people who died here
category for maps or plans
permanent duplicated item
topic's main Wikimedia portal
has part(s) of the class
category for the interior of the item
ethnic group
member category
category for the view from the item
FactGrid item ID
category for the view of the item

Table 6: The 24 excluded generic relations.

Geography. tributary, volcano observatory, mouth of the watercourse, mountain range, drainage basin, highest point, located in protected area, continent, located in or next to body of water, made from material, located in the administrative territorial entity, historic county

Law. court, applies to jurisdiction, published in

Literature. intended public, illustrator, author, has edition or translation, published in, cover art by, enemy, from narrative universe, derivative work, student of, main subject, present in work, hair color, writing language, copyright status as a creator, publisher, list of characters, copyright status, language of work or name, eye color, inspired by, creator, narrative role, notable work, owner of, form of creative work, relative, spouse, residence, native language, child, sibling, occupation, has list, family name, languages spoken, written or signed, country of origin, military or police rank, field of work

Mathematics and mathematicians. studied in

Media and drama. eye color, narrative role, hair color, unmarried partner, inspired by, relative, residence

Military aircraft. powered by, armament

Military people. military or police rank, allegiance

Motorsport. pole position, fastest lap, winner, uses, sport, organizer

Multi-sport event. flag bearer, participant in, sport, part of the series

Other music articles. discography, instrument, location of formation, presenter, influenced by, destination point, notable work, educated at, start point, native language, writing language, has list, languages spoken, written or signed, family name, based on, participant in, field of work, has part(s), award received

Other sports. destination point, winner, start point, competition class, participating team, has list, practiced by, sport

Philosophy. student of

Physics and astronomy. parent astronomical body, discovery method, constellation, epoch, discoverer or inventor, start point, crew member

Places. twinned administrative body, head of government, office held by head of government, capital of, shares border with, authority, located in time zone, official language, capital, language used, located in or next to body of water, highest point, history of topic, owner of, historic county, continent, replaces

Politics and government. office contested, chairperson, applies to jurisdiction, member of political party

Religion. canonization status, office contested, student of, writing language, position held, place of death, copyright status as a creator, spouse, mother, place of burial, father, occupation, unmarried partner, place of birth

Royalty, nobility, and heraldry. family, noble title, place of burial, position held, mother, sibling, spouse, father, child, member of political party, place of death, cause of death, manner of death, canonization status, relative, given name, country of citizenship, unmarried partner, place of birth, occupation, depicts, applies to jurisdiction, military or police rank, depicted by, languages spoken, written or signed, copyright status as a creator, time period, native language

Songs. lyricist, tonality, performer, record label, form of creative work, producer, charted in, recorded at studio or venue, composer, part of, genre, follows, followed by, language of work or name, published in, has characteristic

Television. season, original broadcaster, list of episodes, theme music, original language of film or TV show, screenwriter, voice actor, set during recurring event, media franchise, director, takes place in fictional universe, characters, part of the series, color, narrative location, presenter, set in period, cast member, executive producer, set in environment, distributed by, follows, followed by, list of characters, production company, Kijkwijzer rating, filming location, creator, first appearance, country of origin, has part(s), distribution format, NMHH film rating, inspired by, fabrication method

Transport. terminus location, fare zone, transport network, adjacent station, connecting line, connecting service, terminus, track gauge, located on linear feature, state of use, maintained by, connects with, crosses, owned by, item operated, partially coincident with, located in the administrative territorial entity, destination point, replaces, start point, depicted by, country, powered by, operator, replaced by, is a list of, historic county, has cause, located in protected area, has facility

Video game characters, concepts, and miscellanea. first appearance, narrative role, from narrative universe, present in work, hair color, enemy, subject has role, eye color, product, material, or service produced or provided, legal form, industry, parent organization, residence, creator, family name, given name, armament, is a list of

Video game titles. game mode, CERO rating, PEGI rating, USK rating, compatible with, ESRB rating, software engine, input device, platform, developer, operating system, copyright license, Australian Classification, publisher, designed by, list of characters, distributed by, set in environment, composer, distribution format, soundtrack release, takes place in fictional universe, nominated for, IMDA rating, characters, narrative location, set in period, country of origin, part of the series, genre, BBFC rating, derivative work, copyright status, media franchise, language of work or name, based on, award received, set during recurring event, has characteristic, voice actor, uses

Warships and naval units. vessel class, country of registry, shipping port, location of creation, crew member, conflict, manufacturer, operator, armament, significant event, named after, powered by, different from, allegiance, described by source

Weapons, equipment, and buildings. commissioned by, has facility, historic county, heritage designation, depicted by

World history. capital, official language, replaced by, language used, replaces, allegiance, continent, has cause

Domains without salient relations identified. Baseball; Basketball; Cricket; Hockey; Language; Magazines and print journalism; Military decorations and memorials; Pro wrestling; Recreation.

F Thinking Configurations

Eight of the twenty-one models expose a thinking mode: all four Qwen3.5 sizes and all four Gemma-4 variants. We evaluated all of them, and report the results here rather than in Section 5 because they turn out to characterise decoding behaviour rather than factual recall, and because the two families require different generation protocols, which makes a single side-by-side column misleading.

Protocol. Gemma-4 thinking runs use max_new_tokens = 1,024 with no separate reasoning budget. Qwen3.5 thinking runs cap the reasoning trace at 1,024 tokens—vLLM forces the closing tag at that point—within a total budget of 4,096 tokens. The Qwen configuration therefore changes two things at once relative to the earlier uncapped run: it introduces the reasoning cap *and* quadruples the total budget. The observed change in the closing-tag rate, from 0% to 100%, is almost certainly the cap rather than the budget, but this run does not formally isolate the two. Thinking traces are stripped by the scorer before matching; all figures below are computed over the full 21,775-item set unless stated otherwise.

Qwen3.5: a decoding failure, not a reasoning deficit. Table 7 gives the picture. Over the full item set, thinking is a net loss for every Qwen3.5 size—between 3 and 22 alias-match F1 points below the same model answering directly, and above 6 points in five of the six model–reference combinations. But on the subset of items where the model emits a usable answer, thinking is a consistent *gain* of 1.8 to 7.1 points, in every size and against both ground truths. The two facts are reconciled by the third column: the model fails to produce any usable answer on 15–58% of items, entering a repetition loop that consumes the budget. Median answer length after stripping is 22–91 characters for the 2B, 4B and 9B models, but 9,585 characters for the 0.8B, which essentially never terminates. In the earlier uncapped runs the trace was never closed and 75–90% of items were empty after stripping, driving alias-match F1 down to 0.4–6.9; those numbers measure a chat-template interaction rather than model capability, and we do not report them as such.

		AM F1, all items		clean
Model	% clean	direct	thinking	Δ
Original ground truth				
Qwen3.5 0.8B	42.3	11.65	3.99	+2.56
Qwen3.5 2B	81.8	17.89	11.38	+4.92
Qwen3.5 4B	56.1	26.18	4.12	+4.85
Qwen3.5 9B	85.3	30.53	13.76	+7.07
Augmented ground truth				
Qwen3.5 0.8B	42.3	7.74	3.33	+1.77
Qwen3.5 2B	81.8	12.40	9.53	+3.21
Qwen3.5 4B	56.1	20.42	4.28	+2.18
Qwen3.5 9B	85.3	23.57	13.04	+4.23

Table 7: Qwen3.5 thinking configurations. "% clean" is the share of items on which the model emits a usable answer after the trace is stripped; "direct" and "thinking" are alias-match F1 over all 21,775 items; Δ is the thinking-minus-direct difference *restricted to the clean subset*. Thinking loses over the full set and gains on the subset where it answers.

The clean-subset figures are not a fair headline—they are computed on a self-selected population and are not comparable to any other number in this paper—but they do establish that the deficit is in terminating the answer, not in recalling the fact.

Gemma-4 behaves differently. The failure is Qwen-specific, not a property of thinking runs in general. Under the scorer's own stripping path, Gemma-4 thinking runs produce short, well-formed answers—median 18–28 characters—and essentially no degeneration: at most 0.7% of answers exceed 200 characters, against 9–23% for the uncapped Qwen runs. Their empty-after-stripping

rates are 3.8% (E4B), 6.6% (E2B), 12.5% (31B) and 33.8% (12B). For `gemma-4-12B-it`, the model common to both analyses, thinking raises alias-match F1 from 32.25 to 33.45 against the original ground truth and leaves it essentially unchanged against the augmented one (24.58 to 23.10)—a far smaller effect than the Qwen degeneration, and one that does not survive the change of reference. Its 33.8% empty rate means even this figure is depressed, for a different reason than Qwen's: truncation rather than looping.

What this contributes. The relevance to MUSART is methodological. For a single unchanged model, `Qwen3.5-4B`, alias-match F1 against the original ground truth is 4.0 under the uncapped protocol, 4.1 under the capped protocol scored over all items, and 35.6 under the capped protocol scored over answered items. Three defensible-sounding choices about what counts as the model's answer produce a ninefold spread, without the model or the benchmark changing. The answer-extraction rule is therefore a protocol variable of the same order as the completeness of the reference—the effect quantified in Section 5.2—and a factuality benchmark that leaves it implicit is not reporting a reproducible number. This is the sense in which the thinking axis supports the paper's argument: not as a claim about reasoning, but as a third demonstration that protocol choices move factuality scores as much as models do.

G Evaluation Setup Details and Prompt Format

From triples to questions. Following the LM-as-KB paradigm (Petroni et al., 2019), we convert each (subject, relation) pair into a natural-language question using relation-specific templates: for example, for `heritage designation` the template is "What is [S]'s heritage designation?", where [S] is the subject label. Templates were drafted with an LLM and then manually reviewed and corrected for accuracy and naturalness. Prompts are generated from the structured triples alone—the article text is never shown to the evaluated model—so answers reflect parametric knowledge only. Because many questions admit several valid objects, models are instructed to list each answer on a separate line, and evaluation is set-based.

Prompting and scoring mechanics. Each model answers under five-shot prompting: every test question is preceded by five in-context question–answer exemplars drawn from the same relation and subject category, teaching the expected answer type, surface form, and one-answer-per-line format; exemplar questions are excluded from the evaluation set to prevent leakage. All models are served with vLLM under greedy decoding (temperature 0), with a generation budget of 50 new tokens.

We score model responses with set-based *alias matching* (AM). Every metric (precision, recall, and F1) is reported as a percentage ($\times 100\%$). Each gold object o in the reference set O is defined by a canonical Wikidata label together with an alias set $\mathcal{A}(o) = \{\ell_1, \ldots, \ell_k\}$ of known surface forms, including Wikidata aliases and Wikipedia hyperlinked anchor texts (mean 4.29 surface forms per object, up to 7.82 in *Battles, exercises, and conflicts*). Given a model's line-separated predictions $\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_m\}$, each string $\hat{y}$ is normalised by lowercasing and trimming punctuation. A prediction $\hat{y}$ matches gold object o ($\mathrm{Match}(\hat{y}, o) = 1$) if and only if its normalised form is an *exact match* to at least one normalised alias in $\mathcal{A}(o)$:

$$\mathrm{Match}(\hat{y}, o) = \mathbb{I}\big[\exists \ell \in \mathcal{A}(o) : \mathrm{norm}(\hat{y}) = \mathrm{norm}(\ell)\big]. \quad (1)$$

Rejection of substring false positives. Crucially, alias matching requires full string identity against an alias, *not* substring containment. For example, if the gold entity is *New York City Subway* (whose alias set includes *NYC Subway* and *New York Subway*), returned answers such as "City" or "New" are strictly false positives (0 credit), preventing generic fragments from inflating scores. A generic term such as "Metro" is credited if and only if it is explicitly registered in that specific entity's alias set; otherwise, it receives zero credit.

Each prediction $\hat{y} \in \hat{Y}$ can credit at most one gold object, and each gold object $o \in O$ is credited at most once. Precision is the fraction of predicted answers that match a gold object ($|Y_{\text{correct}}|/|\hat{Y}|$), recall is the fraction of gold objects matched ($|O_{\text{recalled}}|/|O|$), and F1 is their harmonic mean, micro-averaged over all evaluation questions.

Model slate design. Across our twenty-one evaluated open-weight models (0.8B to 70B parameters), *scale* varies within a family and generation: Qwen3.5 at 0.8B/2B/4B/9B, Gemma-3 at 1B/4B/12B/27B, Gemma-4 at E2B/E4B/12B/31B, and the Llama-3 series at 1B, 3B, 8B and 70B. *Generation* varies at roughly matched size: Gemma-2 at 2B and 9B against Gemma-3 and Gemma-4 at comparable scales. *Family* varies at matched scale in the 7–12B band, where Mistral-7B-

You are an AI model designed to answer questions in the following format: “What transport network is the infrastructure [S] part of?”.

Guidelines for Your Response:
- [S] is the subject provided by the user and belongs to the category “Transport”.
- Only include the relevant answer object(s)—without any extra text, explanations, or formatting.
- If multiple objects exist, list each one on a separate line.

Examples of questions and answers:

What transport network is the infrastructure Westminster tube station part of?
London Underground

What transport network is the infrastructure Gare de Lyon part of?
Paris Métro
Réseau Express Régional

What transport network is the infrastructure Alexanderplatz station part of?
Berlin U-Bahn
Berlin S-Bahn

What transport network is the infrastructure Shibuya station part of?
Tokyo Metro
Yamanote Line

What transport network is the infrastructure Flinders Street railway station part of?
Melbourne rail network

What transport network is the infrastructure King’s Cross St. Pancras tube station part of?

Figure 8: Few-shot prompt (no abstain instruction) for the relation `part of`, Transport domain. In-context exemplars are drawn from the same relation and subject category as the test question.

v0.3, Ministral-8B and Mistral-Nemo sit alongside `Llama-3.1-8B`, `gemma-2-9b-it` and `Qwen3.5-9B`. Every score in this paper is computed by a single scoring pipeline over the stored generations of all twenty-one models against both ground truths.

Prompt format. Figure 8 shows the few-shot prompt format used in all evaluations, instantiated for the relation `part of` in the Transport domain. The five in-context exemplars share the test question’s relation and subject category and are excluded from the evaluation set. A variant of this prompt adds an “If you don’t know the answer, respond with ‘I don’t know’” guideline; the single condition reported here omits it, so every question is answered.

Scaling E-Commerce Attribute Extraction with Parallel Decoding

Nikhita Vedula **Dushyanta Dhyani** **Bryan Wang** **Shervin Malmasi**

Amazon.com, Inc.

{veduln, dhyanidd, brywan, malmasi}@amazon.com

Abstract

Customers rely on specific product attributes to compare products and make purchasing decisions, but e-commerce catalogs are messy and unstructured, making it difficult to identify which attributes matter most and extract them at scale. Standard Attribute Value Extraction (AVE) systems treat all attributes equally, producing large, inconsistent attribute sets that do not reflect the factors consumers use to differentiate products. We introduce a two-stage LLM pipeline that first discovers a compact, ranked schema of purchase-discriminative attributes for each product category, then extracts their values from catalog text using a fine-tuned compact LLM (Qwen3-4B) with Hyper-Parallel Decoding (HPD). This pipeline achieves 85% extraction accuracy, on par with the foundational LLM it was distilled from, while reducing inference costs by 92% over foundational LLMs, enabling production-scale use for product discovery and catalog enrichment. The resulting category-level structured representations effectively constitute automatically constructed product knowledge bases, providing consistent, comparable attributes across varied product categories that can ground downstream knowledge-intensive applications.

1 Introduction

Attribute Value Extraction is a core capability for e-commerce systems. Structured product attributes power search, filtering, comparison, recommendation, and product quality workflows (Yang et al., 2022; Brinkmann et al., 2023). However, e-commerce catalogs are highly messy as product information is spread across titles, descriptions, bullets, and semi-structured fields; sellers use inconsistent terminology; and values are often missing, implicit, duplicated, or expressed in non-standard forms (Khandelwal et al., 2023; Yang et al., 2022). Recent work has increasingly explored generative approaches to AVE, where encoder-decoder models and LLMs extract attribute values directly from product text (Blume et al., 2023; Shinzato et al., 2023; Khandelwal et al., 2023). These methods handle heterogeneous inputs and flexible output schemas, but deploying them at e-commerce scale introduces two major challenges. First, defining high-quality schemas manually across thousands of product categories is expensive and difficult to maintain (Huang et al., 2025; Xu et al., 2023). Second, extracting values over massive catalogs is computationally costly, especially when products are added daily and schemas evolve with downstream needs (Chen et al., 2023; Yang et al., 2024; Zhang et al., 2024).

We focus on extracting customer-relevant, purchase-discriminative attributes for each product category. Unlike standard AVE formulations that target broad attribute coverage or extract values for predefined attributes (Yang et al., 2022; Xu et al., 2023), our goal is to identify the attributes most useful for differentiating products within a category, such as wattage for blenders, capacity for storage devices, or noise cancellation for headphones, and ensure all products in the same category share the same schema, making their structured representations directly comparable. Therefore, our pipeline automatically induces a domain ontology (category-level schemas) and populates it at scale, constructing a structured product knowledge base without manual schema engineering.

We propose a scalable two-stage LLM pipeline. First, we automatically discover category-specific schemas from representative catalog examples using a large, foundational LLM (Claude), then standardize semantically similar attributes across categories. In the second stage, we extract values for the discovered schemas from product catalog text using a fine-tuned small language model with Hyper-Parallel Decoding (HPD). HPD exploits the conditional independence of attribute values to decode multiple values in parallel (Glavas et al., 2026b), substantially improving throughput.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 169–177
October 28, 2026 ©2026 Association for Computational Linguistics

We evaluate the system on a large proprietary e-commerce catalog spanning thousands of categories. Our schema discovery stage produces attributes judged correct or relevant in 89.6% of cases, and our fine-tuned value extraction model achieves approximately 85% extraction accuracy, matching the foundational LLM teacher model despite a $>100\times$ reduction in model parameters. The proposed system reduces inference cost by 92%, making repeated catalog-scale processing practical. Our main contributions are:

(i) A two-stage pipeline that automatically discovers compact, category-level attribute schemas and extracts their values at scale, requiring no manual schema engineering.

(ii) A practical, large-scale use case combining knowledge distillation into a compact LLM (Qwen3-4B) with Hyper-Parallel Decoding, processing hundreds of millions of products across multiple marketplaces while achieving 92% cost reduction over a foundational LLM baseline.

(iii) A comprehensive evaluation covering schema quality, extraction accuracy, quantity understanding, determinism, and efficiency, with detailed cost and throughput analysis.

2 Related Work

Product attribute-value extraction. Attribute Value Extraction (AVE) for e-commerce recovers structured product attributes from noisy catalog content to support search, recommendation, comparison, and catalog enrichment (Yang et al., 2022; Brinkmann et al., 2023). Prior work has formulated AVE as sequence tagging (Zheng et al., 2018) or question answering over product context (Wang et al., 2020). These approaches improve extraction for predefined attributes, but typically assume available schemas and can become costly across many attributes and categories.

Generative AVE. Recent work has explored generative AVE, using encoder-decoder or LLM-based models to produce attribute values directly from product text (Shinzato et al., 2023; Khandelwal et al., 2023; Brinkmann et al., 2023). These methods handle heterogeneous inputs and flexible output formats, but production-scale deployment still requires scalable schema construction and efficient extraction over large catalogs. Our work targets both challenges through automatic schema discovery and low-cost generative extraction.

Schema discovery and knowledge base construction. Open-world attribute mining and product schema modeling aim to discover or maintain product attributes with limited human supervision (Xu et al., 2023; Huang et al., 2025). More broadly, automated knowledge base construction encompasses the extraction, integration, and maintenance of structured knowledge from unstructured sources (Weikum et al., 2021), with recent work leveraging LLMs to reshape the classical pipeline of ontology engineering, knowledge extraction, and knowledge fusion (Bian, 2025). In the e-commerce domain, Hongwimol et al. (2026) present a multi-agent LLM framework that induces product types and attribute keys on demand and consolidates them into a globally consistent product knowledge graph, while Peshevski et al. (2025) automate ontology creation and KG population from unstructured product descriptions without predefined schemas. Our work shares the goal of automated product KB construction but differs in emphasis: rather than maximizing broad attribute coverage, we seek compact, purchase-discriminative category-level schemas and prioritize extreme extraction efficiency through parallel decoding, enabling scaling to many millions of products.

Efficient and scalable extraction. Scalability is a central challenge for AVE because real-world products are associated with many attributes. Some QA-style approaches repeatedly process the same product context for different target attributes (Chen et al., 2023). Intra-prompt parallel decoding (Glavas et al., 2026a) addresses this common-context bottleneck by stacking multiple questions into a single prompt and decoding their answers simultaneously through attention mask manipulation. Recent work improves efficiency by caching context representations and using lightweight attribute-context interactions (Yang et al., 2024), while "lifelong" AVE addresses the need to adapt to evolving products, categories, and attributes (Zhang et al., 2024). Our extraction stage is complementary: we specialize a small language model for structured value extraction and use Hyper-Parallel Decoding (HPD) to decode multiple conditionally independent attribute values from the same product context in parallel (Glavas et al., 2026b).

3 Scalable Schema-Guided Attribute Value Generation

An e-commerce catalog contains billions of products, each with a unique set of attributes and values. Extracting these attribute-value pairs for each product in isolation can produce a large number of semantically similar but differently named attributes across the catalog, making them both unmanageable and difficult to use downstream. We propose a two-stage pipeline where we first identify a compact set of key purchase-discriminative attributes per product category, and subsequently extract values corresponding to each attribute for every product per category. Note that the choice of categorization granularity is important: a broad categorization with few categories will result in generic attributes, while being overly granular can result in an untractable attribute set size.

3.1 Problem Statement

Let $\mathcal{P}$ denote a large-scale e-commerce product catalog, categorized into $|\mathcal{T}|$ high-level categories. Each product $p \in \mathcal{P}$ has catalog context $\mathbf{x}_p$ comprising its title, description, and other unstructured product details, and is associated with a category $t(p) \in \mathcal{T}$. We seek to produce, for each product p, a structured representation $\phi(p) = \{(a_i,\ v_i)\}_{i=1}^{N}$ where $\mathcal{A}_{t(p)} = \{a_1, \ldots, a_N\}$ is a schema of N purchase-discriminative attributes defined at the category level, and each v_i is the value of attribute a_i extracted from $\mathbf{x}_p$ if available. This decomposes into two sub-problems:

Attribute Schema Discovery: For each category $t \in \mathcal{T}$, identify a schema $\mathcal{A}_t = \{a_1, \ldots, a_N\}$ of attributes that are most informative for consumer purchase decisions within that category.

Attribute Value Generation: For each product $p \in \mathcal{P}$, given the schema $\mathcal{A}_{t(p)}$ and the catalog context $\mathbf{x}_p$, extract the value v_i of each attribute $a_i \in \mathcal{A}_{t(p)}$ from $\mathbf{x}_p$. The key design constraint is *category-level schema sharing*: all products belonging to the same category t share the same attribute schema $\mathcal{A}_t$, ensuring that $\phi(p)$ and $\phi(p')$ are comparable for any two products p, p' with $t(p) = t(p')$.

3.2 Stage 1: Attribute Schema Discovery

Category-Specific Attributes Specification For each category, we randomly select k representative products that satisfy a minimum threshold of customer engagement metrics, ensuring a balanced selection of quality and diversity. We then use the Claude Sonnet 4 LLM to identify M key attributes for the category from the title and description of each product. The prompts are tuned to also extract each attribute's description, data type (numerical, categorical, single-value, multi-valued, or free-form text), and a list of standardized values the attribute can take for the given category.

Global Attributes Specification While each product has its unique set of attributes (e.g., noise canceling ability for headphones or storage capacity for a hard disk), certain attributes are universally applicable across all products. We supplement the category-specific attributes with the following set of global attributes: *total pack size, total weight, total volume, per product weight, per product volume.*

Attribute Name Standardization While defining a fixed set of M attributes per category constrains the local schema, running Stage 1 across P categories can produce $M \times P$ attributes (which can range from a few to tens of thousands) with semantic redundancy (e.g., *Water Resistance, Waterproof rating, IP Water Rating*). We develop an LLM-based clustering approach to standardize these. We embed all attribute names using a Qwen3-8B embedding model and apply hierarchical agglomerative clustering to group semantically similar attributes. Each cluster is then assigned a canonical name via one of two methods: a) **Heuristic:** replacing all cluster members with the most frequent representative name, provided it satisfies a minimum similarity threshold; or b) **LLM-based:** using an LLM to generate a standardized name while preserving domain-specific context.

3.3 Stage 2: Parallel Attribute Value Generation

Once a standardized attribute set has been generated, the next step is to extract values for those attributes from all catalog products. A straightforward approach is to prompt a foundational LLM with the predefined attributes and product catalog information. While state-of-the-art LLMs can accomplish this with high accuracy, scaling to hundreds of millions of products in a cost-effective way poses a significant challenge. Even at an optimistic throughput of 100K products per hour, processing a complete catalog would take multiple weeks, which does not scale for a real-world e-commerce service.

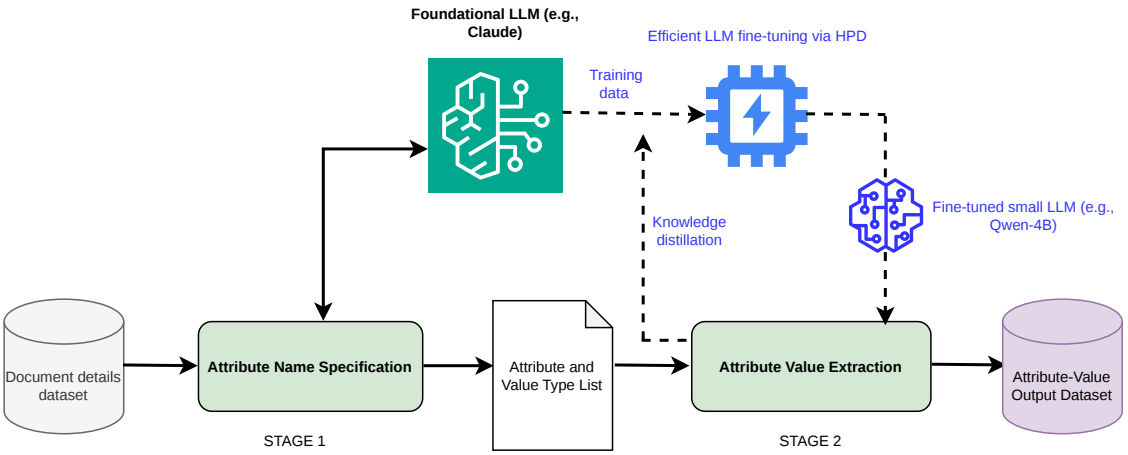


Figure 1: Overview of our two-stage pipeline for scalable schema-guided attribute value generation. Stage 1 discovers a compact schema of N purchase-discriminative attributes per category using a large foundational LLM. Stage 2 extracts attribute values from product catalog text using a fine-tuned compact LLM with Hyper-Parallel Decoding, generating all N values simultaneously.

Given a product context and a fixed schema of N attributes, standard autoregressive decoding generates the output JSON sequentially, requiring approximately $\sum_{i=1}^{N} K_i$ decoding steps, where K_i is the token length of the value for attribute i.

Hyper-Parallel Decoding (HPD): Efficiently Scaling Inference Throughput and Cost In schema-guided AVE, many attribute values are conditionally independent given the same product context. HPD (Glavas et al., 2026b) exploits this structure by decoding values for multiple attributes in parallel, reducing decoding steps from $\sum K_i$ to approximately $\max_i K_i$. Standard autoregressive decoding generates attribute-value pairs sequentially, attending to all previously generated values before producing the next. HPD observes that conditioned on the same product context x_p, the values of different attributes are conditionally independent: extracting a blender's wattage should not causally depend on having first extracted its color. This conditional independence allows HPD to extract values in parallel during decoding.

The mechanism works by constructing a JSON skeleton template with attribute keys and blank value fields, creating position ID "gaps" of size $K_{\max}$ at each value location. Special BOV (beginning-of-value) tokens mark where generation should occur. During the first inference step, the model outputs next-token probabilities for the entire input, and HPD selects N probabilities at the BOV positions to generate the first token of each value simultaneously. Subsequent tokens are appended to the end of the sequence but assigned position IDs that logically place them in the previously created gaps, maintaining key-value cache functionality. A position-based causal attention mask ensures tokens only attend to appropriate context despite being arranged out of order in physical memory. In subsequent steps, only the N new tokens are passed to the model with the cached keys/values, generating the t-th token for all values in parallel until completion. HPD further stacks J documents in a single prompt, decoding $J \times N$ tokens per step; combined with batch inference (batch size b), this yields $b \times J \times N$ tokens per step, achieving large efficiency gains.

HPD requires custom fine-tuning for parallel decoding. BOV tokens are added to the vocabulary, and a "block ID" tensor tracks which inference step each token belongs to (block 0 for the prompt, block 1 and above for generated tokens). Position IDs are manipulated during training to match the inference configuration, and a custom 2D attention mask enforces that prompt tokens only attend to the prompt while generated tokens attend to themselves and lower block IDs. This enables the model to match autoregressive quality while maintaining $\sim$10$\times$ speedup during inference.

Attribute Value Standardization A key challenge in large-scale attribute extraction is ensuring that extracted values are consistent and comparable across products, categories, and marketplaces.

Without standardization, the same attribute can produce highly variable surface forms (e.g., "30 oz", "30 ounces", "30oz", "thirty ounces") that are difficult to use in downstream applications requiring exact matching or aggregation. We address this at multiple levels: Stage 1 defines expected data types and standardized categorical values for each attribute; Stage 2 prompts instruct the model to adhere to type requirements, output numerical values with standard units, select from predefined categorical options, and produce comma-separated lists for multi-valued attributes; and a strictly structured JSON output format further reduces variability across runs.

We additionally experiment with Constrained Hyper-Parallel Decoding at inference time, where token-level constraints force the model to generate only valid data types and standardized categorical values. For categorical attributes, a logit mask is applied at each decoding step to permit only tokens corresponding to valid options from the Stage 1 schema; for numerical attributes, a regex expression enforces the correct number-unit format (e.g., allowing only digit and unit tokens in sequence). In practice, we find that the combination of careful prompt engineering and knowledge distillation from a foundational LLM during fine-tuning already yields high standardization compliance, with constrained decoding affecting fewer than 1% of extracted values while providing a formal guarantee of format adherence.

4 Experiments and Results

4.1 Experimental Setup

Product Catalog and Attribute Schema Inventory. We perform experiments with our pipeline on a large proprietary e-commerce catalog containing more than 30M products spanning a few thousand product categories. Each product is represented by catalog text and metadata, including title, description, and other structured or semi-structured product details. Stage 1 produces category-level key attribute schemas, which are then standardized across categories. The resulting schema inventory contains several thousand unique attributes. Across the evaluated catalog, these attributes correspond to tens of millions of observed attribute values, including numerical, categorical, and free-form values.

Training data generation. To train the Stage 2 extraction model, we construct a supervised dataset of 100K product-schema examples sampled across product categories. For each example, we prompt the Claude Sonnet 4 LLM with the product catalog context and corresponding Stage 1 attribute schema to produce a structured JSON object containing the extracted value for each attribute, or null when absent. This data is used for HPD fine-tuning.

Models and HPD configuration. We fine-tune Qwen3-4B and Qwen3-8B models for purchase-discriminative attribute value generation using Hyper-Parallel Decoding. Each product is paired with a fixed category-level schema, and the model generates one value slot per attribute in structured JSON format. We use up to 20 attributes per product with $K_{\max} = 100$ tokens and 6 input products per prompt. Fine-tuning is performed on an AWS EC2 p4de.24xlarge instance with 8 A100 GPUs.

At inference time, HPD decodes values for all attributes in parallel, rather than generating the JSON output strictly left-to-right. We compare against a foundational LLM baseline accessed via batched API inference, which processes requests asynchronously with variable queue wait times.

Evaluation. We comprehensively evaluate both stages using automated and human evaluation. For Stage 1, we use Claude Sonnet 4.5 as an LLM judge to classify discovered attributes as *correct/relevant*, *irrelevant*, or *too vague/too specific*. For Stage 2, we use the same LLM judge to evaluate extraction quality on 20K products, classifying outputs as: *correct extraction*, *correct null* (attribute absent and model correctly returns null), *incorrect extraction*, *missed extraction* (value present but model returns null), or *ungrounded extraction*. We additionally perform human evaluation on 500 products. Finally, to assess determinism, we run the complete pipeline five times on 50 products from diverse categories and compare outputs across runs.

Attribute Schema Discovery Quality We observe that 89.6% of the discovered key attribute names are classified as "correct" or relevant by the LLM judge with respect to the category, 4.0% are deemed irrelevant, and 6.4% are considered either too vague or too specific for the given category (e.g., "Performance" is overly vague as it lacks discriminative specificity, while "Bluetooth Codec Support" is too specific for a broad "Audio Equipment" category). Table 1 summarizes the schema quality evaluation. This assessment is conducted at the category level across a few thousand categories, covering 20 key attributes per category.

Classification	% of Attributes
Correct / Relevant	89.6%
Too vague or too specific	6.4%
Irrelevant	4.0%

Table 1: Stage 1 attribute schema quality evaluated by an LLM judge across a few thousand product categories (20 attributes per category).

Attribute Value Generation Quality As shown in Table 2, the accuracy of key attribute extraction is very similar for both fine-tuned models, at approximately 85%. Both Qwen3 models fine-tuned with HPD achieve extraction accuracy on par with or slightly exceeding the foundational teacher LLM, demonstrating that knowledge distillation combined with task-specific fine-tuning can fully close the quality gap despite a $>100\times$ reduction in model parameters. Notably, HPD introduces no quality degradation while providing significant speed gains.

Of the remaining approximately 15%, the LLM judge flags 4.6% as *potentially ungrounded* extractions (values not directly verifiable from the explicit input text), 3.5% as missed extractions, and 2% as incorrect extractions. Manual investigation of 50 such cases, corroborated by our 500-product human evaluation, reveals that approximately half of ungrounded cases are reasonable inferences from implicit context (e.g., inferring "plastic" from "BPA-free"), bringing the effective ungrounded extraction rate to approximately 2.3%. Additionally, fewer than 2% involve format violations where the model does not conform to Stage 1 type constraints. The remaining 3-4% include ambiguous cases due to catalog noise, where the LLM judge cannot make a definitive determination, and are excluded from evaluation.

Interestingly, the larger Qwen3-8B model performs slightly worse than Qwen3-4B. Our practical constraints - processing several hundred million products with daily incremental and monthly full catalog refreshes - require sustained high throughput on commodity GPU instances, where models exceeding 8B parameters significantly reduce per-GPU batch capacity. We therefore focus on models at or below 8B parameters and rely on knowledge distillation to close any quality gap.

Determinism and Coverage. To assess reliability, we run both stages five times on 50 products from diverse categories. For both schema discovery and value extraction, approximately 93% of outputs are identical or semantically equivalent across runs, with variations primarily in subjective attributes such as "special features." Overall pipeline coverage is 99.997%: attributes are successfully generated for all products except those lacking catalog information or with incorrect category assignments (0.003% of products).

Human Evaluation We perform human evaluation on 500 products spanning diverse categories, assessing correctness of extracted values against input catalog text for both categorical and numerical attributes. Human judges rate 85% of extractions as correct, consistent with the LLM judge evaluation, with Qwen3-4B again performing slightly better than Qwen3-8B. Overall, the LLM judge and human annotators agree on 91% of generated attribute values. Of the errors, approximately 6% are incorrect extractions; the remainder are primarily data type violations and, for numerical quantity attributes, cases where the attribute information is present only in product images rather than the catalog text provided as input.

Efficiency and Throughput Table 3 compares the cost and throughput of our HPD-based extraction against both standard autoregressive decoding on the same model and the foundational LLM baseline using cloud-hosted batched API inference. HPD achieves an average throughput of $\sim$159K products per hour per instance. Running inference on 30M products using 5 parallel instances takes approximately 36 hours at a total cost of $\sim$\$5,000. Standard autoregressive decoding on the same Qwen3-4B model yields approximately 18K products per hour per instance ($\sim 9\times$ lower throughput), requiring over 330 hours on the same hardware at a cost of $\sim$\$46K.

The foundational LLM baseline, accessed via cloud-hosted batched API inference with 10 parallel batch jobs, achieves a nominal throughput of $\sim$100K products per hour excluding queue wait times. In practice, variable queue delays add significant additional latency, making the effective wall-clock time substantially longer than the reported 300 hours. The total API cost for 30M products is $\sim$\$63,000, representing a 92% cost reduction with HPD. Fine-tuning the Qwen3-4B model requires 8 hours on a single instance with 8 A100 GPUs, at a one-time cost of $\sim$\$350.

Model	Total Correct	Correct Extractions	Correct Null Extractions	Incorrect Extractions	Missed Extractions
Foundational LLM Baseline	84.80%	54.30%	30.50%	1.30%	4.00%
Qwen3-4B (AR)	84.60%	51.80%	32.80%	2.00%	3.80%
Qwen3-4B (HPD)	**85.40%**	**52.40%**	33.00%	**1.70%**	**3.50%**
Qwen3-8B (HPD)	84.40%	49.60%	**34.80%**	2.60%	**3.50%**

Table 2: Stage 2 key attribute value generation quality for fine-tuned models, evaluated by an LLM judge (Claude Sonnet 4.5) on a sample of 20K products. Bold indicates the best result per column (highest for correct metrics, lowest for incorrect/missed).

	Qwen3-4B (HPD)	Qwen3-4B (AR)	Foundational LLM
Throughput (products/hr)	159K	18K	100K†
Parallel instances / jobs	5	5	10
Time for 30M products	36 hrs	333 hrs	300+ hrs†
Inference cost	$5K	$46K	$63K
Cost reduction vs. Foundational LLM	**92%**	27%	—

† Excludes variable queue wait times, which add significant latency.

Table 3: Efficiency comparison for 30M products between HPD inference, standard autoregressive (AR) inference, and foundational LLM batched API inference.

Effect of Attribute Schema Size on Parallelism. HPD decodes all N attribute values simultaneously, generating $J \times N$ tokens per inference step. Since decoding steps remain fixed at $K_{\max}$ regardless of N, throughput in attribute-values per hour scales approximately linearly with schema size. We validate this by varying $N \in \{10, 12, 16, 18, 20\}$ and observe that product throughput remains approximately constant while attribute-values extracted per hour increases proportionally. Meanwhile, the foundational LLM baseline cost scales linearly with N, yielding progressively larger cost reductions from approximately 84% at $N = 10$ to 92% at $N = 20$, making our choice of $N = 20$ near-optimal for both parallelism and category coverage.

5 Practical Use Case Evaluation

We employ our full two-stage pipeline to process product catalogs across two major marketplaces, extracting key attributes for hundreds of millions of products. Large-scale inference is distributed across 4–20 A100 GPU instances, each processing products independently to achieve linear throughput scaling. The pipeline is containerized for reproducibility across instance types.

Given that new products appear in the catalog daily and product content is frequently updated, we plan a tiered refresh strategy. New products are processed on a daily or weekly cadence, while the full catalog is re-processed monthly to capture updates. Schema discovery (Stage 1) is refreshed less frequently as category-level attributes are relatively stable; schemas are versioned per refresh cycle and validated against the previous version before deployment. For Stage 2, outputs that fail JSON parsing or violate data type constraints are caught by automated post-processing validation and re-processed in the subsequent batch.

The extracted attribute-value pairs are published to an internal large-scale knowledge base and made queryable by downstream use cases. These structured representations can enable several applications: (i) product comparison via shared category-level schemas; (ii) catalog enrichment, where extracted values fill gaps in incomplete product listings and augment sparse catalog metadata; (iii) catalog quality improvement by surfacing inconsistencies in product details; and (iv) personalized product understanding through compact structured inputs to recommendation and search systems.

We additionally validate our outputs through live seller interviews across two marketplaces, where sellers evaluate the extracted attributes for quality and usefulness in differentiating products and informing customer purchase decisions. Sellers provide positive feedback, confirming that the extracted attributes reflect factors they consider when positioning their products relative to competitors.

5.1 Sample System Output

Attribute	Data Type	Standardized Values
Brew Type	categorical	Manual, Semi-Automatic, Automatic, Super-Automatic, Pod
Pump Pressure	numerical	(bars)
Heating System	categorical	Thermocoil, Single Boiler, Dual Boiler, Thermoblock
Temperature Control	categorical	PID, Thermostat, Digital
Grinder Type	categorical	Conical Burr, Flat Burr, Blade, None
Tank Capacity	numerical	(oz)
Milk System	categorical	Steam Wand, Auto Frother, None
Build Material	categorical	Stainless Steel, Plastic, Aluminum

Table 4: Stage 1 output: discovered attribute schema for `COFFEE_MAKER`.

We illustrate the end-to-end pipeline output for the product category COFFEE_MAKER. Stage 1 discovers a schema of purchase-discriminative attributes, which we show in Table 4. Given this schema and the catalog text of a specific product (title, description, and bullet points), Stage 2 extracts the following structured attribute-value pairs:

```
{"Brew Type": "Semi-Automatic",
 "Pump Pressure": "15 bars",
 "Heating System": "Thermocoil",
 "Temperature Control": "PID",
 "Grinder Type": "Conical Burr",
 "Tank Capacity": "67 oz",
 "Milk System": "Steam Wand",
 "Build Material": "Stainless Steel"}
```

All products in the COFFEE_MAKER category share this same schema, making their structured representations directly comparable. For example, a budget pod machine would yield {"Brew Type": "Pod", "Heating System": "Thermoblock", "Grinder Type": "None", "Build Material": "Plastic", ...}, enabling direct attribute-level comparison.

6 Conclusion

We presented a scalable two-stage pipeline for extracting purchase-discriminative attributes from large e-commerce catalogs. The first stage automatically discovers compact, category-level schemas using a foundational LLM, eliminating manual schema engineering across thousands of categories. The second stage extracts attribute values at scale using a fine-tuned Qwen3-4B model with HPD, which exploits the conditional independence of attribute values to decode them simultaneously. The system achieves 85% extraction accuracy, on par with the foundational teacher LLM, while reducing inference costs by 92%. Future directions include incorporating multimodal inputs such as product images to capture visually conveyed attributes, and extending the pipeline to multilingual catalogs.

Limitations

Our system extracts attributes exclusively from textual catalog content such as product titles, descriptions, and bullet points. Product information that is conveyed only through images, such as visual design details, color variations, or quantities shown in packaging photos, is not captured by our current approach. Incorporating multimodal inputs to address this gap is left for future work. The current pipeline has been evaluated only on English-language catalogs. Extending the system to non-English marketplaces would require multilingual fine-tuning and potentially different schema discovery prompts, which we have not yet explored. The conditional independence assumption underlying HPD assumes that extracting the value of one attribute should not depend on previously extracted values. While this holds for most attributes in practice, certain attributes may exhibit correlations (e.g., product weight and volume, or material and durability). Our empirical results suggest this does not significantly impact extraction quality, but a deeper investigation into correlated attribute groups is warranted. Finally, we evaluated our fine-tuned models exclusively using the Qwen3 model family in 4B and 8B parameter sizes. Other model architectures or larger model sizes may yield different quality-efficiency tradeoffs, and the optimal model choice may vary across domains or catalog characteristics.

References

Haonan Bian. 2025. Llm-empowered knowledge graph construction: A survey. *arXiv preprint arXiv:2510.20345*.

Ansel Blume, Nasser Zalmout, Heng Ji, and Xian Li. 2023. Generative models for product attribute extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 575–585, Singapore. Association for Computational Linguistics.

Alexander Brinkmann, Roee Shraga, and Christian Bizer. 2023. Extractgpt: Exploring the potential of large language models for product attribute value extraction. *arXiv preprint arXiv:2310.12537*.

Wei-Te Chen, Keiji Shinzato, Naoki Yoshinaga, and Yandi Xia. 2023. Does named entity recognition truly not scale up to real-world product attribute extraction? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 152–159, Singapore. Association for Computational Linguistics.

Theodore Glavas, Nikhita Vedula, Dushyanta Dhyani, Antonios Valkanas, Yilun Zhu, and Shervin Malmasi. 2026a. Intra-prompt parallel decoding for common-context question answering. *Preprint*, arXiv:2609.05707.

Theodore Glavas, Nikhita Vedula, Dushyanta Dhyani, Yilun Zhu, and Shervin Malmasi. 2026b. Breaking the autoregressive chain: Hyper-parallel decoding for efficient llm-based attribute value extraction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 36792–36808.

Pollawat Hongwimol, Haoning Shang, Chutong Wang, Zhichao Wan, Yi Gao, Yuanming Li, Lin Gui, Wenhao Sun, and Cheng Yu. 2026. AutoPKG: An automated framework for dynamic e-commerce product-attribute knowledge graph construction. In *Findings of ACL*.

Yunhan Huang, Klevis Ramo, Andrea Iovine, Melvin Monteiro, Sedat Gokalp, Arjun Bakshi, Hasan Turalic, Arsh Kumar, Jona Neumeier, Ripley Yates, Rejaul Monir, Simon Hartmann, Tushar Manglik, and Mohamed Yakout. 2025. AttributeForge: An Agentic LLM Framework for Automated Product Schema Modeling.

Anant Khandelwal, Happy Mittal, Shreyas Kulkarni, and Deepak Gupta. 2023. Large scale generative multimodal attribute extraction for e-commerce attributes. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 305–312, Toronto, Canada. Association for Computational Linguistics.

Dimitar Peshevski, Riste Stojanov, and Dimitar Trajanov. 2025. Ai agent-driven framework for automated product knowledge graph construction in e-commerce. In *Proceedings of the 1st GOBLIN Workshop on Knowledge Graph Technologies*.

Keiji Shinzato, Naoki Yoshinaga, Yandi Xia, and Wei-Te Chen. 2023. A unified generative approach to product attribute-value identification. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6599–6612, Toronto, Canada. Association for Computational Linguistics.

Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D. Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to extract attribute value from product via question answering: A multi-task approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Gerhard Weikum, Xin Luna Dong, Simon Razniewski, and Fabian Suchanek. 2021. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends in Databases*.

Liyan Xu, Chenwei Zhang, Xian Li, Jingbo Shang, and Jinho D. Choi. 2023. Towards open-world product attribute mining: A lightly-supervised approach. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12223–12239, Toronto, Canada. Association for Computational Linguistics.

Li Yang, Qifan Wang, Jianfeng Chi, Jiahao Liu, Jingang Wang, Fuli Feng, Zenglin Xu, Yi Fang, Lifu Huang, and Dongfang Liu. 2024. Eave: Efficient product attribute value extraction via lightweight sparse-layer interaction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1491–1505, Miami, Florida, USA. Association for Computational Linguistics.

Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Kumar Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. Mave: A product dataset for multi-source attribute value extraction. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1256–1265.

Tao Zhang, Chenwei Zhang, Xian Li, Jingbo Shang, Hoang Nguyen, and Philip Yu. 2024. Stronger, Lighter, Better: Towards Life-Long Attribute Value Extraction for E-Commerce Products. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8631–8643, Bangkok, Thailand. Association for Computational Linguistics.

Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1049–1058.

Evidence-Carrying Claims: Fail-Closed Verification Gates for Language-Model-Assisted Research as Knowledge Base Construction

Robert Sneiderman
Independent
robbysneiderman@gmail.com

Abstract

Language-model-assisted pipelines emit theorems, experiment reports, and literature claims faster than any reviewer pool can check them. This paper argues that the knowledge base such pipelines demand is not a store of extracted facts but a *claim ledger*: a knowledge base whose entries pair each claim with the artifact that verifies it (a machine-checked proof, a rerunnable receipt, or a resolved forecast) and whose gates *fail closed*, blocking an unverified claim from entering the record rather than scoring it after the fact. We ground the argument in three mechanisms already running in one research operation: a twenty-two-gate manuscript-admission stack, a Lean 4 library in which the claim is the artifact, and a sealed-prediction registry with forced resolution. We then state what a community-scale claim ledger would require of its schema, its adjudication protocol, and its incentives. The evidence is a single operation and does not establish generalization; the contribution is the knowledge-base-construction framing and the fail-closed gate architecture.

1 Introduction

An end-to-end research pipeline built around language models can now produce a paper-shaped object, with a plausible theorem, a results table, and a related-work section, in hours (Lu et al., 2024). In a controlled comparison, expert reviewers rated ideas from such pipelines as more novel than ideas from human researchers (Si et al., 2024). The binding constraint on machine-assisted science has moved: generating candidate claims is cheap, and deciding which of them deserve trust is not.

The tools built so far for that decision run downstream of the claim. Fact-extraction and claim-verification datasets score an already-written statement against retrieved evidence (Thorne et al., 2018; Wadden et al., 2020; Guo et al., 2022). Hallucination detectors estimate, probabilistically, whether a generated statement is confabulated (Farquhar et al., 2024; Ji et al., 2023). Both attach a judgment to a claim that is already in circulation. For human-scale literatures this is tolerable: review and replication eventually catch part of what slips through, at a measurable cost (Ioannidis, 2005). For pipelines that emit claims at machine speed, post hoc scoring loses the race by construction. The scored claim has already entered the record, and the score itself is an estimate.

This paper argues for moving the verification boundary upstream and treating what results as a knowledge-base-construction problem. The object to build is a claim ledger: a knowledge base whose atomic entry is a claim bound to a verification artifact and a checker verdict, and whose population step is a gate that fails closed (Figure 1). Section 2 places the proposal against post hoc verification and against the structured scientific-assertion models that already exist. Section 3 defines the entry types and the gate semantics. Section 4 describes three mechanisms that run today, at manuscript, theorem, and forecast granularity, inside our own research operation. Section 5 sketches the community-scale version, and Section 6 states what the evidence here cannot support. Two terms recur. Lean 4 is a proof assistant whose small trusted kernel re-checks every proof term and reports the axioms a proof relies on (de Moura and Ullrich, 2021). A Brier score is the mean squared difference between a probability forecast and the 0/1 outcome: 0 is perfect, and always saying 0.5 scores 0.25 (Brier, 1950).

Our contributions:

1. A framing: the knowledge base that language-model-assisted science needs stores claims bound to verification artifacts, and it is populated by fail-closed gates at generation time, not by extraction from published text.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 178–184
October 28, 2026 ©2026 Association for Computational Linguistics

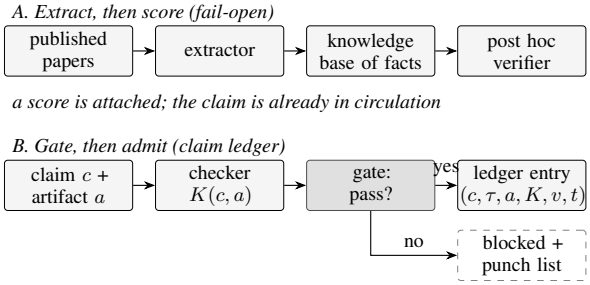


Figure 1: The proposal in one contrast. Extraction-based construction verifies a claim after it exists and scores it; a claim ledger admits a claim only when its checker returns a verdict, and a missing or unrunnable artifact blocks the entry with a typed reason instead of a score.

2. Existence evidence: three working mechanisms from one research operation that instantiate the ledger at three granularities, described concretely enough to reconstruct the design, with three live entries shown in full (Table 1).

3. A sketch of the community-scale infrastructure: an entry schema with checker-typed evidence links, an adjudication protocol for contested or decayed artifacts, and incentive placement at venues.

Non-claims. No controlled evidence is offered that these mechanisms generalize beyond the operation that built them. Each component composes existing ideas: proof kernels, preregistration, and the fail-closed discipline of continuous integration; the claimed contribution is the composition and the knowledge-base framing. This is not a calibration study, and the release contains no dataset, no benchmark, and no deployed community system.

2 Related work

Fact verification datasets (Thorne et al., 2018; Wadden et al., 2020) and the automated fact-checking literature (Guo et al., 2022) verify statements against retrieved text after the fact; the ledger differs in when verification happens (admission rather than post hoc) and in what counts as evidence (a runnable artifact rather than a supporting passage). Hallucination detection (Farquhar et al., 2024; Ji et al., 2023) is complementary: a detector can triage which generated claims to demand artifacts for, but its output is a score, so it cannot serve as a fail-closed gate. Process supervision (Lightman et al., 2024; Uesato et al., 2022) verifies reasoning steps rather than outcomes; it is gate-shaped but applied to model training, whereas the ledger applies the same instinct to publication artifacts. Proof assistants and their libraries (de Moura and Ullrich, 2021; The mathlib Community, 2020) show that the formal layer already operates at scale under fail-closed discipline, since a broken proof blocks integration; we read mathlib as the largest existing claim ledger. The metascience literature diagnosed the reliability problem (Ioannidis, 2005) and built the instruments (Nosek et al., 2018; Tetlock, 2005); the ledger proposal operationalizes those instruments as admission control and gives them a knowledge-base home.

Structured representations of scientific assertions are older than this proposal and supply its vocabulary. Semantic publishing (Shotton, 2009) asked for machine-readable claims inside articles; nanopublications (Groth et al., 2010) fix the smallest publishable unit as an assertion plus its provenance plus its publication information; micropublications (Clark et al., 2014) model claims, evidence, and arguments as a graph; the Open Provenance Model (Moreau et al., 2011) standardizes the derivation records those graphs carry; and research objects (Bechhofer et al., 2013) bundle the data, code, and workflow a claim depends on. The ledger takes the split between assertion and provenance from this line and adds what it leaves open. In each of these models an assertion can be minted with a citation, or nothing, as its support, and the provenance record describes how the assertion was made rather than whether a third party can re-establish it. The ledger types every evidence edge by an executable checker, refuses an entry whose checker cannot run, and admits forecasts whose truth value has a scheduled arrival. That is the whole delta beyond the fail-closed admission mechanism, and we state it narrowly on purpose.

3 Claim ledgers and fail-closed gates

Fix a research operation that emits claims. A *claim ledger* is a knowledge base whose atomic entry is a tuple

$$e = (c,\ \tau,\ a,\ K,\ v,\ t),$$

where c is the claim text, τ is its type, a is a verification artifact, K is a checker (a procedure a third party can run on the pair (c, a)), $v = K(c, a)$ is the verdict, and t is the admission timestamp. Three types cover the claims a research pipeline emits:

- **Formal** ($\tau = \mathsf{formal}$): the artifact is a machine-checkable proof; the checker is a proof kernel;

the verdict is exact and includes the axioms the proof assumes.

- **Empirical** ($\tau = \mathsf{empirical}$): the artifact is a rerun receipt: pinned environment, inputs, script, and output hash; the checker reruns and compares within a stated tolerance.
- **Forecast** ($\tau = \mathsf{forecast}$): the claim concerns a future outcome; the artifact is a probability, a horizon, and a resolution criterion, all sealed before the outcome exists; the checker is the resolution protocol, and the verdict arrives at the horizon.

Checkers are ordered by trust, not by weight. The three checkers answer different questions and do not sit on one evidential scale. A kernel settles a formal statement exactly; a rerun settles whether an artifact reproduces its recorded output; a resolution protocol settles whether a sealed forecast came true. What the ledger does order is how much a reader must take on trust to accept the verdict: nothing beyond the kernel for a proof, the pinned environment and the stated tolerance for a rerun, the honesty of the seal timestamp for a forecast, and the citing author for a free-text citation. An evidence edge carries its checker, so a reader never has to guess which kind of support a claim has, and a proof of a lemma nobody uses does not outrank a rerun of a result everybody uses.

Checkability is not validity. A rerun receipt certifies that the artifact reproduces its output; it does not certify the sentence a paper builds on that output. The ledger handles the gap by scope. The claim text of an empirical entry is the statement the checker settles: this script, on these inputs, yields this table. A wider statement, that the effect generalizes or that the difference is causal, is a separate claim that must carry its own artifact, in practice a sealed forecast of a replication with a resolution criterion, or else enter the visible `WARN` tier described below. Reproducibility is a floor the gate can enforce, not the validity the reader wants, and the ledger records which of the two a given edge supplies.

Gate semantics. A gate is a predicate on entries evaluated at admission. The gate *fails closed* when its default is denial: a claim whose artifact is missing, stale, or unrunnable is blocked, and the burden of producing a checkable artifact sits with the claimant. Post hoc scoring is the fail-open alternative: the claim circulates and carries a score. Fail-closed admission is standard practice for code, where a failing test blocks a merge, and close to nonexistent for scientific claims.

Why this is knowledge base construction. Population, in this design, is not extraction: entries are emitted by the gates themselves at the moment a claim passes, so the ledger is complete for its operation by construction. Curation is the adjudication of disputed or decayed artifacts. Temporal scope is carried by the forecast layer, whose entries have truth values with scheduled arrival times. A minimal entry serializes in a dozen lines; this one is the forecast entry for the present paper:

```
{"claim": "AKBC 2026 accepts this paper",
 "type": "forecast",
 "artifact": {"ref": "seals/akbc-vision",
   "hash": "sha256:4881d859b7e6",
   "p": 0.55},
 "checker": "resolution",
 "verdict": "pass",
 "sealed": "2026-07-24T14:59Z",
 "resolves": "2026-09-15"}
```

4 Three mechanisms in operation

The design above is not hypothetical. Our research operation, which submits mathematics, measurement, and systems papers to open venues, runs a version of each layer. What each mechanism checks is described below; every specific check named exists in the operation's source as of the day of writing, and Appendix B lists the full gate set. The point is existence, not scale.

4.1 A manuscript gate stack that fails closed

Every manuscript leaves the operation through a single gate runner that applies twenty-two gates and emits one verdict from a four-token set; one token permits submission, and the other three block with a punch list. The gates are deterministic wherever possible. A citation gate parses every bibliography entry, resolves it against the arXiv API, Crossref, and DBLP, and returns a typed per-entry verdict (`PASS`, `FAIL_MISMATCH`, `FAIL_NOT_FOUND`, `WARN_UNVERIFIABLE`); any `FAIL` blocks. A metrics gate recounts quantitative claims about our formal library (Section 4.2) against the live source tree, so a stale module or theorem count cannot ship. An exact-support gate requires that a citation attached to a key claim support that exact claim rather than a weaker neighbor, and a hedge gate requires every hedged key step to carry a discharge

Type	Claim	Artifact	Checker	Verdict
formal	The sharp McDiarmid inequality for i.i.d. inputs with constant width (`sharp_mcdiarmid_inequality_iid_const_width`).	Proof term in the public library at commit `5e41e98`.	Lean 4 kernel, run by the repository's continuous integration.	pass; CI green at that commit.
empirical	A deployed health-information assistant's release gate, on a fixed 107-probe regression set, refused 23 probes after hardening versus 55 before, with no released answer failing the sentence-level support check in either run.	Two stored evaluation artifacts (SHA-256 prefixes `e9503721c3e1`, `19f6da02b9d1`) and a 40-line recount script.	Rerun of the recount over the stored artifacts.	pass on 2026-09-06; artifacts are operation-internal, so a public ledger would hold this entry in the `WARN` tier.
forecast	"AKBC 2026 vision track accepts this paper by 2026-09-15."	$p = 0.55$, sealed 2026-07-24T14:59Z, horizon 2026-09-15, criterion: the venue decision.	Resolution protocol: the decision, recorded against the seal.	hit; resolved 2026-09-01 (decision dated 2026-08-31).

Table 1: Three live entries from the operation's ledger, one per type, shown in the schema of Section 3. The empirical row is a claim about a deployed system, not about mathematics; its artifacts are not public, which is exactly the case the `WARN` tier exists for.

obligation; an open obligation blocks. A blind-policy gate compares the byline to the venue's verified anonymity policy and fails closed when no verified policy is on record. Deterministic parses of the built PDF block blank pages, margin bleed, bitmap fonts, template-default strings, and a count printed under two denominators. A final gate refuses the ready verdict unless a sealed prediction of the submission's outcome is already on file (Section 4.3).

Two properties matter for the ledger argument. The stack rejects in practice, and its rejections are typed and actionable: a blocked manuscript receives the failing gate, the offending object, and a one-line fix, not a score. The gate set also accretes by post-mortem: each gate below that traces to an incident was added the day the defect was found, so that class of failure is now unreachable. A fail-closed gate set maintained this way is a knowledge artifact in its own right: it encodes, as executable checks, the ways the operation has been caught being wrong.

What the stack has recorded. The operation's machine-readable submitted set lists 28 manuscripts that left through the stack between June and September 2026. The runner does not yet persist its verdict history, so we cannot report how many of those runs blocked before the final pass, nor a false-positive rate; that log is the first fix on our list. What is on record is the incident register, 36 entries across the operation, six of which concern manuscripts and produced seven gates that did not exist before the incident: a half-blank page that passed an eyeball check and a real-name byline on a venue assumed to be blind (both from one incident; G13 and G14 in Appendix B), content clipped at the page edge on every page (G16), a submit packet that handed the operator raw LaTeX to paste (G17), a template-default footer on nine pages (G18), one count published under two denominators (G21), and a paper whose numbers had been read from a scratch directory that no longer existed (G22). The stack has no override flag; the remaining bypass is a person uploading a PDF that never ran it.

4.2 Machine-checked claims: the artifact is the claim

The strongest ledger entries are those in which claim and artifact coincide. A theorem in a proof assistant is its own verification artifact: the statement is the claim, the proof term is the evidence, the kernel is the checker, and the axiom report states exactly what was assumed (de Moura and Ullrich, 2021). Nothing about such an entry is probabilistic, and checking it requires no trust in its author.

As an existence proof that this layer scales to research mathematics, our operation maintains FormalSLT, an open-source Lean 4 library of statistical learning theory built on mathlib (The math-

lib Community, 2020).[1] At that commit the library holds 358 source files, 184,769 lines, and 6,389 theorem and lemma statements, with no unfinished-proof markers, and its continuous integration re-checks every file on every push. The library is a claim ledger in the sense of Section 3: looking a theorem up means re-checking it, and because every proof names the lemmas it uses, the kernel itself enforces the evidence edges between entries. Formalization is expensive, so this layer is reserved for claims whose downstream weight justifies it; the ledger assumes a mixture of entry types.

4.3 Sealed predictions: the temporal layer

Forecast entries give the ledger its temporal dimension: they are claims admitted before their truth value exists. Our operation seals a probability, a horizon, and a resolution criterion for submission outcomes and internal milestones before the outcome is known; because the criterion is fixed at seal time, the forecaster cannot later choose what counts as success, and a resolution that does not post-date its seal is refused. At the time of writing the registry holds 150 forecast rows. The 52 venue-outcome forecasts that were sealed through the registry's seal tool and have since resolved score a Brier of 0.213, against 0.25 for a constant forecast of 0.5. A broader tally that also scores component-level rows resolved on trust rather than through the seal tool gives 0.146 over 134 components; we do not count it as calibration evidence, and we cite the honest number once, as evidence that a working operation can sustain sealing and forced resolution for months, and for nothing more. Preregistration (Nosek et al., 2018) and forecasting-judgment research (Tetlock, 2005) supply the underlying instruments. The addition here is placement: seals live in the same ledger as proofs and rerun receipts, linked to the claims they concern, and the manuscript gate stack refuses to pass a submission with no seal on file. The forecast row of Table 1 is the seal for this paper.

5 Community-scale claim ledgers

The mechanisms above serve one operation. The vision is the shared version, and it decomposes into three design problems the knowledge-base community is equipped for.

Schema. The minimal entry of Section 3 is thin on purpose; a shared ledger needs more. Claims become first-class nodes; evidence links are typed by checker, verdict, and date, and ordered by the trust hierarchy of Section 3. The assertion, provenance, and publication-information split that nanopublications standardize (Groth et al., 2010) carries over directly, and to it the ledger must add the scope of each claim, the evidence dependencies between entries, support and refutation links, and the adjudication history below. Since artifacts are re-checked and verdicts can change, an entry is append-only history, not a mutable cell. The hard schema problem is claim identity: recognizing that two papers state the same claim is exact for formal entries and open for informal ones, where restatement, changed scope, and partial overlap defeat entity resolution over surface strings. Provenance, temporal scoping, and alignment remain the classical knowledge-graph concerns (Hogan et al., 2021).

Adjudication. Someone must run the checkers, and disputes need a protocol. A failed rerun must be classified: artifact rot (an environment that no longer builds) is not refutation, and the protocol must separate them. Verdicts downgrade rather than delete: an entry whose artifact has decayed drops in the trust order but keeps its history. Empirical entries need a re-verification cadence; formal entries need re-checking only when their assumptions change. The adjudication trail is part of the ledger, so a reader can audit the process that admitted a claim.

Incentives. Gates cost claimants effort, so placement decides adoption. Venues are the natural first gate operators: they already run fail-closed checks of weaker kinds (format, anonymity, plagiarism), and a portal is a chokepoint where a check cannot be skipped. A track could require one typed artifact per named contribution and check it mechanically at the portal. The ledger also creates a new citable object: resolutions and refutations are recorded with their adjudicator, so verification work becomes visible and creditable. A graded on-ramp keeps the gate honest about its own coverage: claims without a cheap checker enter a visible `WARN` tier instead of being silently excluded.

[1] `https://github.com/Robby955/FormalSLT`, MIT license. Counts were measured on a fresh clone of the `main` branch at commit `5e41e98` on 2026-09-06 with the commands in Appendix A; the metrics gate of Section 4.1 re-verifies such counts before any manuscript ships.

6 Limitations

Three boundaries bound the argument. The first is evidential: all evidence here comes from a single operation, and the selection effects run deep. The gates were built by the operation they gate, so passing them may reflect shared blind spots rather than correctness; the gate runner keeps no verdict history, so the block rate and the false-positive rate of the stack are unmeasured; the 52 resolved forecasts are not independent samples, and their resolution criteria were written by the forecaster being scored. The Brier number supports a feasibility claim and nothing stronger. The second is a coverage bias: fail-closed admission favors claims with cheap checkers, namely formal mathematics, rerunnable computation, and resolvable forecasts. Much of science, including qualitative and interpretive work and experiments too expensive to rerun, has no cheap checker, and a naive gate would entrench whatever happens to be easy to verify; the `WARN` tier is a patch for this, not a solution, and Section 3 already concedes that a rerun certifies reproducibility rather than the inference drawn from it. The third is that the community-scale problems are unsolved: claim identity across restatements, artifact rot at scale, and adjudication under adversarial pressure. The concrete next step is small: a pilot at a single workshop track, requiring one typed artifact per named contribution and checking it mechanically at the portal. The step the vision cannot skip is the adjudication protocol, on which the rest depends.

References

Sean Bechhofer, Iain Buchan, David De Roure, Paolo Missier, John Ainsworth, Jiten Bhagat, Philip Couch, Don Cruickshank, Mark Delderfield, Ian Dunlop, Matthew Gamble, Danius Michaelides, Stuart Owen, David Newman, Shoaib Sufi, and Carole Goble. 2013. Why linked data is not enough for scientists. *Future Generation Computer Systems*, 29(2):599–611.

Glenn W. Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3.

Tim Clark, Paolo N. Ciccarese, and Carole A. Goble. 2014. Micropublications: A semantic model for claims, evidence, arguments and annotations in biomedical communications. *Journal of Biomedical Semantics*, 5(1):28.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *Automated Deduction (CADE 28)*, volume 12699 of *Lecture Notes in Computer Science*, pages 625–635. Springer.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630.

Paul Groth, Andrew Gibson, and Jan Velterop. 2010. The anatomy of a nanopublication. *Information Services and Use*, 30(1–2):51–56.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, and 1 others. 2021. Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37.

John P. A. Ioannidis. 2005. Why most published research findings are false. *PLoS Medicine*, 2(8):e124.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations (ICLR)*. ArXiv:2305.20050.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. 2011. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756.

Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. 2018. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606.

David Shotton. 2009. Semantic publishing: The coming revolution in scientific journal publishing. *Learned Publishing*, 22(2):85–94.

Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. 2024. Can LLMs generate novel research ideas? a large-scale human study with 100+ NLP researchers. *arXiv preprint arXiv:2409.04109*.

Philip E. Tetlock. 2005. *Expert Political Judgment: How Good Is It? How Can We Know?* Princeton University Press, Princeton, NJ.

The mathlib Community. 2020. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pages 367–381. ArXiv:1910.09336.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 809–819. ArXiv:1803.05355.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550. ArXiv:2004.14974.

A Reproducing the library counts

The counts in Section 4.2 were produced on 2026-09-06 by the following commands against a fresh clone at commit `5e41e98`; the last command searches for the unfinished-proof keyword outside the module docstrings that state its absence.

```
git clone --depth 1 \
  https://github.com/Robby955/FormalSLT
cd FormalSLT
find FormalSLT -name '*.lean' | wc -l
find FormalSLT -name '*.lean' \
  -exec cat {} \; | wc -l
find FormalSLT -name '*.lean' | xargs \
  grep -c '^theorem\|^lemma' | awk -F: \
  '{s+=$2} END {print s}'
grep -rn '\bsorry\b' FormalSLT \
  --include='*.lean' | grep -v -i \
  'no `sorry\|no sorry' | wc -l
```

B The manuscript gate stack

The gate runner applies the gates below to a registered paper and emits one of four verdicts: `SUBMIT_READY`, `PANEL_PENDING`, `NEEDS_FIX`, or `MISSING_BUNDLE`. Only the first permits submission. "Deterministic" gates are pure parses or recounts with no model call; "model-assisted" gates call a language or vision model and degrade to a warning, never to a silent pass, when the model is unavailable.

G1. Bundle present: source, built PDF, review prompt, and submit packet all exist. Deterministic.

G2. Venue format: document class, page cap, anonymization grade, and required sections match the venue's verified policy. Deterministic.

G3. Voice: punctuation and self-referential-phrase limits on the source. Deterministic.

G4. PDF freshness: the built PDF is newer than its source. Deterministic.

G5. Sealed prediction: a forecast of the submission's outcome is on file (Section 4.3). Deterministic.

G6. Disclosure log: the venue's disclosure requirements have a log entry. Deterministic.

G7. Bound panel verdict: an adversarial review panel's receipt whose hashes recompute against the registered source, PDF, and prompt; a stale receipt cannot colour a newer build. Model-assisted.

G8. Non-checks section: the review prompt names what reviewers must not propose. Deterministic.

G9. Verified metrics: quantitative claims about the formal library match a live recount of the source tree. Deterministic.

G10. Exact support and hedges: each key-claim citation supports the exact claim, and each hedged key step carries a discharge obligation. Model-assisted audit with a deterministic block on open obligations.

G11. Formal style: changed proof files pass the library's lint. Deterministic.

G12. Review flags: no open review flag remains on the paper or venue. Deterministic.

G13. Layout fill: no mostly blank page, by per-page character count. Deterministic.

G14. Blind policy: the byline matches the venue's verified anonymity policy; fails closed with no policy on record. Deterministic.

G15. Figure format: no bitmap font and no raster image below 250 dpi at its placed size. Deterministic.

G16. Margin overflow: no overfull line beyond 5 pt in the build log and no ink in the outer page band on any page. Deterministic.

G17. Packet abstract: the submit packet carries a clean plaintext abstract and a one-line summary within the portal's limits. Deterministic.

G18. Template residue: no template-default string survives in the built PDF. Deterministic.

G19. Visual review: rendered pages pass a multimodal check for overflow, clipping, and broken layout. Model-assisted.

G20. Byline identity: a real-name submission carries the canonical author, affiliation, and address tokens. Deterministic.

G21. Numeric consistency: no "k/n ($p\%$)" pair in the PDF disagrees with itself. Deterministic.

G22. Results provenance: the paper directory references no ephemeral scratch path, and every absolute input path resolves on disk. Deterministic.

Evidence-Guided Schema Normalization for Temporal Tabular Reasoning

Ashish Thanga[1*], **Vibhu Dixit**[1*], **Abhilash Shankarampeta**[2], **Vivek Gupta**[1†]
[1]Arizona State University, [2]UC San Diego
athanga@asu.edu, vdixit5@asu.edu, ashankarampeta@ucsd.edu, vgupt140@asu.edu

Abstract

Temporal reasoning over evolving semi-structured tables poses a challenge to current QA systems. We propose an approach that recasts the task as automated knowledge base construction: (1) prompting an LLM to synthesize a 3NF-compliant relational schema from Wikipedia infobox timelines, (2) populating the schema to obtain a queryable database, and (3) generating and executing SQL queries against it, with QA accuracy serving as an extrinsic evaluation of the constructed knowledge base. In a controlled grid of three schema generators crossed with six query models, the schema source accounts for 79.5% of the exact match (EM) variance against 1.6% for the query model: replacing the schema, and the prompt scaffolding derived from it, shifts EM by 14.7 to 20.0 points, whereas replacing the query model under a fixed schema shifts it by 4.4 to 12.1. From this evidence, we distill three candidate schema-design principles: balanced normalization, semantic naming, and consistent temporal anchoring, framed as correlational hypotheses. Our best configuration (Gemini 2.5 Flash schemas + Gemini-2.0-Flash queries) reaches 80.39 EM, 11.5 points above the strongest reported baseline (68.89 EM); an open-weights configuration reaches 79.52.

1 Introduction

Large Language Models (LLMs) excel at understanding tabular data (Liu et al., 2024; Fang et al., 2024), but struggle with temporal reasoning over evolving semi-structured data. The TRANSIENTTABLES dataset (Shankarampeta et al., 2025) highlights this limitation by asking questions about Wikipedia infoboxes sampled over time. Current approaches treat these temporal infobox timelines as text, forcing LLMs to perform implicit temporal reasoning across multiple table snapshots. Despite recent advances in prompting strategies, state-of-the-art models using the best prompting technique achieve only an exact match (EM) score of 68.89 on TRANSIENTTABLES with Gemini-2.5-Pro (Shankarampeta et al., 2025).

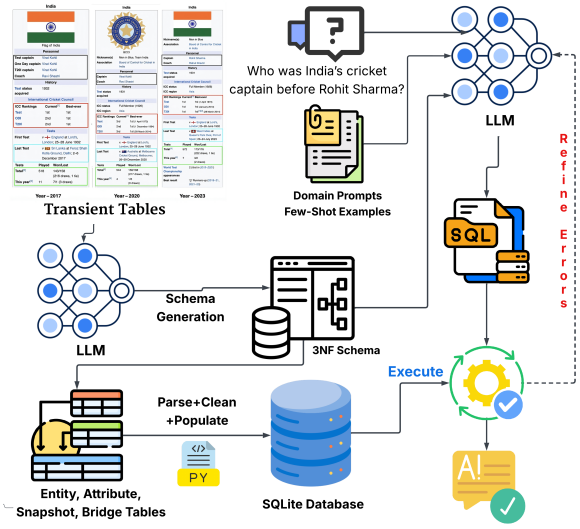


Figure 1: Our three-stage pipeline, read left to right. **Stage 1, Schema Generation**: an LLM turns infobox timelines into a 3NF schema. **Stage 2, Schema Population**: scripts clean and insert each JSON snapshot, one per `snapshot_id`. **Stage 3, SQL Generation & Execution**: a schema-guided prompt turns a question into SQL, executed against the database. Stages 1–2 run once per domain, offline. Worked example: Appendix A.1.

The TRANSIENTTABLES task: Given a series of Wikipedia infobox snapshots for an entity (e.g., a cricket team) sampled over time, answer natural language questions that require temporal reasoning. For example, *"Who was India's cricket captain before Rohit Sharma?"* requires identifying temporal ordering across snapshots, while *"How many Tests did India play between 2020 and 2023?"* requires aggregating time-bounded statistics.

We reframe temporal table reasoning as a text-to-SQL problem, an approach we call EVINORM (**Evi**dence-guided schema **Norm**alization). Un-

*Equal Contribution
†Corresponding Author

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 185–223
October 28, 2026 ©2026 Association for Computational Linguistics

like standard text-to-SQL that assumes pre-existing static schemas, we address the novel challenge of synthesizing schemas from evolving semi-structured data. Rather than asking LLMs to reason over infobox timelines serialized as JSON strings, we dynamically generate normalized database schemas and leverage the LLMs' ability to generate SQL. This offers three advantages: (1) explicit temporal structure via timestamps and foreign keys, (2) reduced ambiguity through schema constraints and data normalization, and (3) access to SQL's temporal operators that directly map to the reasoning required (date arithmetic, window functions, and temporal joins) (Liu et al., 2025). Unlike direct text-based reasoning, which often fails on complex semi-structured data (Kulkarni et al., 2025; Vo et al., 2022; Deng et al., 2025; Wu and Hooi, 2025), every answer is produced by an executed, fully traceable query, enabling systematic debugging through prompt or schema refinement.

On TRANSIENTTABLES, schema-guided SQL generation improves substantially over the strongest reported baseline (§5). More importantly, our controlled comparison isolates *where* the improvement comes from: decomposing the variance of a 3×6 generator $\times$ query-model grid attributes 79.5% of the EM variance to the schema source and 1.6% to the query model. The unit varied is a schema together with the prompt scaffolding derived from it, not relational structure in isolation.

Contributions. Our main contributions are:

1. **Task framing and pipeline.** EVINORM, a three-stage pipeline (schema synthesis, database population, schema-guided SQL generation) and, to our knowledge, the first systematic study of LLM-driven schema *synthesis* as automated knowledge base construction from evolving semi-structured data, with temporal QA as the extrinsic evaluation (§3).
2. **Controlled comparison.** Across 3 schema generators $\times$ 6 query models on TRANSIENTTABLES, a variance decomposition attributes 79.5% of EM variance to the schema source against 1.6% to the query model, and well-normalized schemas transfer across query models. The finding does not depend on proprietary models: an open-weights configuration (Llama-3.3-70B for both schema synthesis and querying) reaches 79.52 EM, within 0.9 EM of the best cell in the grid (§5).
3. **Design principles.** Three candidate schema-design principles (balanced normalization, semantic naming, consistent temporal anchoring), stated as correlational hypotheses rather than established causes, supported by join-depth and fragmentation analysis and by an Allen-interval analysis delimiting the temporal scope of the approach (§6, Appendix A.2).
4. **Error analysis.** A stratified analysis showing that 70% of residual failures stem from data understanding rather than SQL or schema generation (§6.2).

2 Related Work

Our work sits at the intersection of tabular reasoning, temporal question answering, text-to-SQL generation, and symbolic reasoning with LLMs. It also connects to automated knowledge base construction: systems such as DBpedia (Auer et al., 2007), YAGO2 (Hoffart et al., 2013), and Wikidata (Vrandečić and Krötzsch, 2014) harvest structured, temporally scoped knowledge from Wikipedia infoboxes into fixed ontologies, whereas we synthesize a task-specific relational schema with an LLM and treat downstream QA as an extrinsic evaluation of the constructed knowledge base. A natural alternative is to skip synthesis and query the curated base directly, which would be cheaper to deploy where applicable. We do not report that comparison. It is not like-for-like here, since TRANSIENTTABLES answers are defined relative to specific infobox revisions whereas Wikidata statements and their `start time`/`end time` qualifiers are curated on a separate editorial track; and it trades schema synthesis for ontology mapping rather than removing it, since infobox fields must still be aligned to properties that may not exist for long-tail entities. Quantifying how much of TRANSIENTTABLES can be answered directly from Wikidata is left to future work. We frame temporal table reasoning as schema *synthesis* followed by schema-guided SQL generation, and position our contribution relative to each line of work below.

Tabular Reasoning. Research on semi-structured tables spans inference and question answering (Gupta et al., 2020), fact verification (Chen et al., 2020), conversational summarization and exploration (Zhang et al., 2020), and table retrieval and augmentation (Zhang and Balog, 2020). Neural encoders such as TAPAS (Herzig et al., 2020), TaBERT (Yin et al., 2020), and

TABBIE (Iida et al., 2021) learn joint table–text representations, while Table2vec (Zhang et al., 2019) and TabGCN (Pramanick and Bhattacharya, 2021) explore alternative table encodings. Closer to our work, symbolic approaches such as BINDER (Cheng et al., 2023), DATER (Ye et al., 2023), and Chain-of-Table (Wang et al., 2024) interleave language models with executable programs over tables, but all operate on a *given*, fixed table or schema. Our setting differs fundamentally: no schema exists, and the structure itself must be synthesized from evolving input before any reasoning can occur.

Temporal Question Answering. Datasets such as TIME-SENSITIVEQA (Chen et al., 2021), TORQUE (Ning et al., 2020), TEMPQA-WD (Neelam et al., 2022), and CRONQUESTIONS (Saxena et al., 2021) capture temporal dependencies in text and curated knowledge graphs; TempTabQA (Gupta et al., 2023) and TRAM (Wang and Zhao, 2024) extend temporal evaluation to tables, though without explicit schema normalization. All of these operate over a single table snapshot or a static knowledge graph. TRANSIENTTABLES (Shankarampeta et al., 2025) is, to our knowledge, the only public benchmark whose instances are *sequences* of table snapshots; this state-change signal is precisely what our schema-synthesis approach exploits (§4).

Text-to-SQL and Schema Design. Text-to-SQL has progressed from schema-linking architectures (Guo et al., 2019; Wang et al., 2020) and constrained decoding (Scholak et al., 2021) to LLM-era benchmarks targeting robustness and scale (Yu et al., 2019; Gan et al., 2021; Lei et al., 2025; Liu et al., 2025); temporal SQL over versioned databases has a long history in the database community (Vo et al., 2022; Snodgrass, 1999). This literature almost universally assumes a pre-existing, human-curated schema. Most directly related, Kohita (2025) systematically studies the effect of normalization on SQL generation over *given*, static schemas, finding that normalized (2NF/3NF) schemas introduce challenges such as base-table selection and join-prediction errors. Our work differs on three axes. First, the *setting*: no schema is given, so an LLM must synthesize it from raw, evolving JSON timelines, and this synthesis step is what we systematically study. Second, the *direction of the finding*: in the temporal regime we observe the converse and more nuanced trade-off, where balanced normalization helps, because it is precisely what makes snapshot joins expressible, while over-normalization hurts (§6.1). Third, our temporal-anchoring principle (a uniform `snapshot_id` key enabling equality joins across time-varying tables) has no analog in static text-to-SQL.

Symbolic Reasoning with LLMs. NL2FOL (Lalwani et al., 2025), LOGIC-LM (Pan et al., 2023), and LINC (Olausson et al., 2023) demonstrate that mapping language into formal representations improves inference robustness, and LLM-symbolic integration has been applied to temporal tabular reasoning (Kulkarni et al., 2025; Kulkarni and Srikumar, 2025; Deng et al., 2025; Wu and Hooi, 2025). We instantiate this program with relational algebra as the formal system and show that the quality of the *constructed* symbolic representation accounts for more of the variance in downstream accuracy than the capacity of the model querying it.

3 Methodology

Task Definition. An instance consists of an entity e with a timeline $S_e = \{(t_1, J_1), \ldots, (t_k, J_k)\}$ of k (typically 8–12) timestamped snapshots, where t_i is the timestamp of a Wikipedia revision and J_i is that revision's infobox serialized as a JSON object mapping attribute names to values, and a natural-language question q whose answer requires reasoning across multiple snapshots: temporal ordering, concurrency, aggregation over bounded intervals, or duration. Successive J_i are neither identical nor schema-aligned: attributes appear, disappear, are renamed, and change value between snapshots, and it is precisely this state-change signal that the question set targets. Given (S_e, q), the system must produce the answer a.

EVINORM transforms this temporal table reasoning task into a text-to-SQL problem through three stages: (1) dynamic schema generation from temporal infoboxes, (2) database population with cleaned data, and (3) schema-guided SQL generation with few-shot prompting.

3.1 Normalized Schema Generation

Given the snapshots S_e, we generate a normalized relational schema using LLMs: the model receives 2-3 example JSON timelines from the same domain and is prompted for a Third Normal Form (3NF) schema. It produces four table types:

- **Entity tables**: Core entities with unique identifiers (e.g., `Countries`, `Leaders`, `Players`)
- **Attribute tables**: Domain-specific lookup tables (e.g., `LeaderRoles`, `MatchFormats`)
- **Snapshot tables**: Temporal data with `snapshot_id` as timestamps, containing time-varying attributes
- **Bridge tables**: Many-to-many relationships linking entities, attributes, and time periods (e.g., `SnapshotLeaders`)

Normalization rationale. 3NF reduces redundancy across temporal snapshots by factoring out repeated information. Schemas are explicitly prompted for 3NF and manually validated for atomicity of values, complete dependency on primary keys, and absence of transitive dependencies. We do not formally prove 3NF compliance; Appendix A.5 gives the validation procedure and examples.

3.2 Schema Tables (database) population

Automated Python scripts populate SQLite databases from LLM-generated schemas, handling date format variations, text normalization, and referential integrity validation (more details in Appendix A.6).

3.3 Schema-Guided SQL Generation

Few-shot Examples Construction. We build one exemplar per question type. TRANSIENTTABLES instantiates ten to fifteen recurring question templates in each domain, and we take a single question from each, so the exemplar set covers the domain's question taxonomy uniformly instead of over-representing any one pattern. Given a question Q with expected answer A, we prompt the model with the schema, Q, A, and database access; it proposes an SQL query, executes it, compares the result to A, and refines until exact match. The resulting "gold queries" span before/after ordering, concurrent roles, temporal aggregation, tenure duration (`JULIANDAY()` arithmetic), and temporal extrema.

Few-shot Provenance. The question-answer pairs used to construct gold queries are drawn exclusively from the TRANSIENTTABLES development split, one per question template; no test-set question appears as a few-shot exemplar in any domain. Execution-based refinement accesses only the populated database, never test-set answer keys. Because TRANSIENTTABLES questions instantiate recurring temporal templates by design, our exemplars reflect those templates, as do the IRE+CoT baseline's CoT exemplars; the comparison is few-shot SQL against few-shot CoT, not against zero-shot.

Domain-specific Prompting. Domain-specific prompts include: (1) complete schema with constraints, (2) natural language descriptions of key relationships, (3) 8-10 SQL templates, (4) 10-15 few-shot examples, and (5) critical rules (e.g., "Never hardcode entity IDs; always use subqueries", "Use `DISTINCT` for queries spanning multiple snapshots"). Unlike generic text-to-SQL prompting (Lei et al., 2025), this encodes temporal semantics (`snapshot_id` ordering, `JULIANDAY()` arithmetic) and domain constraints directly, narrowing the search space from all well-formed SQL to temporally valid queries. Appendix A.1 walks a single question through all three stages, from infobox snapshots to executed SQL, and Appendix F.1 lists the complete synthesized Countries schema.

Query Generation and Evaluation. At test time, given question Q from TRANSIENTTABLES, we concatenate the domain prompt with Q and query the SQL generation LLM. The model generates an SQL query, which we execute on the corresponding SQLite database. We compare the retrieved result to the expected answer using Exact Match (EM), token-level F1, Rouge-1, and Rouge-L scores.

Domain-Generality of the Pipeline. The pipeline contains no benchmark-specific machinery; Appendix A.3 sets out the cost of adapting it to a new semi-structured source.

4 Experiments

Benchmark: We evaluate on TRANSIENTTABLES (Shankarampeta et al., 2025): 3,971 questions derived from over 14,000 table revisions spanning 1,238 entities, grouped into 10 structurally distinct entity domains (countries, economies, government agencies, and seven sports domains), with each entity represented by 8–12 infobox snapshots. The number 10 refers to domains, not entities: each domain contains on the order of a hundred distinct entities, and every reported score is a domain-level average over that domain's evaluation questions. We use this single benchmark deliberately: to our knowledge, it is the only publicly available benchmark of *multi-snapshot* temporal semi-structured tables. Existing temporal QA datasets

such as TempTabQA (Gupta et al., 2023), TIME-SENSITIVEQA (Chen et al., 2021), CRONQUESTIONS (Saxena et al., 2021), and TEMPQA-WD (Neelam et al., 2022) contain single-snapshot tables or static knowledge graphs; repurposing them would either discard the state-change signal our method targets or require synthetic snapshot generation.

Model selection: We select models to vary the capability tier and openness rather than to survey the frontier. The open-weights side spans an order of magnitude within one family (Llama-3.1-8B-Instruct, Llama-3.3-70B-Instruct) plus a second family at the small end (Qwen-2.5-7B-Instruct), separating scale effects from family effects; the proprietary side spans two tiers of one family (Gemini-2.0-Flash, Gemini-2.5-Pro) and a compact model from another (GPT-4o-mini). We use instruction-tuned checkpoints throughout because every stage is prompted and never fine-tuned, and base checkpoints do not reliably respect the output-format constraints the prompts impose. Schema generators are drawn from the same pool, so for two of the three, the same model also appears as a query model.

Baseline: We compare against Information-Retrieval-Extraction with Chain-of-Thought (IRE+CoT), the strongest of the four prompting strategies reported by Shankarampeta et al. (2025) and the published state of the art on this benchmark. It is also the closest structural match to our pipeline: it decomposes the task into stages (retrieve snapshots, extract attributes, reason) and supplies few-shot exemplars, so the comparison is few-shot SQL against few-shot CoT rather than against a zero-shot control. We reuse the published per-model numbers and evaluate our pipeline with the same six query models.

Normalized Schema Generation LLMs: Schemas are generated by Gemini 2.5 Flash (Comanici et al., 2025), Llama 3.3-70B-Instruct (Grattafiori et al., 2024), and Llama 3.1-8B-Instruct. Each generator sees 2–3 example timelines per domain and emits one 3NF schema per domain in a single prompted call, giving 10 schemas per generator, each reused unchanged for every entity and question in its domain. We neither re-sample schemas per entity nor measure run-to-run variability of the synthesis step, so the effect sizes we report are between-generator rather than between-run (see Limitations). We additionally synthesize schemas with GPT-5.4-mini as an out-of-family robustness check. GPT-5.4-mini is used for schema synthesis only and never for query generation; the resulting schemas are queried by three of the six models listed below (§5, Appendix B) and are reported separately from the main grid.

SQL Query Generation LLMs: We evaluate six query models: Gemini-2.0-Flash, Qwen-2.5-7B-Instruct (Bai et al., 2023), Llama-3.1-8B-Instruct, Llama-3.3-70B-Instruct, GPT-4o-mini (Hurst et al., 2024), and Gemini-2.5-Pro.

Decoding: All stages use temperature $= 0.1$, top-p $= 0.9$, and a 2048-token generation limit. Database population and few-shot exemplar construction follow §3.

5 Results

Table 1 presents our main findings comparing different schema generation and SQL query generation model combinations against the Information-Retrieval-Extraction (IRE) with Chain of Thought (CoT) baseline from Shankarampeta et al. 2025.

Schema generation helps. Our best configuration (Gemini 2.5 Flash schema, Gemini-2.0-Flash queries) reaches 80.39 EM / 82.1 F1, 11.5 EM points (16.7% relative) above the strongest reported baseline (Gemini-2.5-Pro with IRE+CoT: 68.89 EM / 75.3 F1). Sixteen of the eighteen configurations outscore the IRE+CoT result for the same query model; both exceptions occur with Gemini-2.5-Pro, the one model whose baseline is itself strong.

The schema is the lever, but no single generator dominates. Averaged over the six query models, Llama-3.3-70B schemas are the strongest (77.21 EM), ahead of Gemini 2.5 Flash (74.00) and well ahead of Llama-3.1-8B (62.31); 70B schemas beat Flash schemas for four of the six query models. Flash schemas nonetheless yield the single best cell in the grid (80.39 EM) and the widest spread across query models (12.1 EM points, against 6.8 for 70B and 4.4 for 8B). The conclusion we draw is therefore not that one generator writes better schemas in general, but that the schema is the effective lever: replacing the schema under a fixed query model moves EM by 14.7 to 20.0 points, against 4.4 to 12.1 for the reverse.

	Baseline		Schema Generation LLMs					
	IRE+CoT		Gemini 2.5 Flash		Llama-3.3-70B-Instruct		Llama-3.1-8B-Instruct	
SQL/CoT LLMs	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Gemini-2.0-Flash	48.98	55.93	80.39	82.11	72.70	73.33	60.37	60.32
Qwen-2.5-7B-Instruct	30.40	29.22	77.84	78.86	75.30	75.07	60.91	60.32
Llama-3.1-8B-Instruct	29.83	37.95	78.08	79.49	78.70	77.85	61.78	60.66
Llama-3.3-70B-Instruct	41.08	54.91	69.86	75.54	79.52	79.91	64.79	63.66
GPT-4o-mini	44.53	52.27	69.56	66.80	78.54	78.70	62.88	63.15
Gemini-2.5-Pro	68.89	75.3	68.25	67.18	78.51	78.20	63.13	63.07

Table 1: Exact Match (EM) and F1 scores on TRANSIENTTABLES across schema generators (columns) and SQL query generators (rows).

Source	**df**	**Sum Sq.**	**% of total**	**F**
Schema generator	2	737.9	79.5	21.08*
Query model	5	15.2	1.6	0.17
Residual (interaction)	10	175.0	18.9	
Total	17	928.2	100.0	

Table 2: Two-way analysis of variance without replication over the 18 EM scores in Table 1, with schema generator and query model as the two factors. *df*: degrees of freedom. *Sum Sq.*: sum of squared deviations attributable to that source. F: the source's mean square (Sum Sq. divided by df) over the residual mean square, so $F \gg 1$ marks a source explaining far more variation than noise. With one observation per cell, the schema $\times$ query-model interaction cannot be separated from error and serves as the residual. $^*p < 0.001$; the query-model effect is not significant ($p = 0.97$).

What the comparison holds fixed. Within each column the prompt template, few-shot strategy, and decoding parameters are identical, so the comparison is internally controlled. It is not a comparison of relational structure alone, however: the relationship descriptions, SQL templates, and exemplars are derived per schema, so the unit varied is a schema *together with the scaffolding it induces*, and the evidence is correspondingly associational. Limitations states both caveats in full, with the perturbation experiment that would resolve them.

How much variance does the schema account for? Table 2 decomposes the variance of the eighteen EM scores with a two-way analysis of variance, treating schema generator and query model as factors. The schema source accounts for 79.5% of the total sum of squares ($F(2, 10) = 21.1$, $p < 0.001$); the query model accounts for 1.6% and is indistinguishable from noise ($F(5, 10) = 0.17$, $p = 0.97$). The remaining 18.9% is the schema $\times$ query-model interaction, inseparable from error in a one-observation-per-cell design but more than ten times the query-model main effect, with a visible candidate source: two of the three columns peak at a query model from the generator's own family, an affinity we record as a pattern worth testing rather than an established effect. One qualification bounds the headline share: it is carried substantially by the weakest generator. With Llama-3.1-8B schemas excluded, the schema and query-model factors contribute comparably (14.9% and 17.1% of the variance in that subgrid). What the grid supports is therefore that a poor schema imposes a ceiling no query model lifts, rather than that schema choice dominates at every level of schema quality.

Statistical reliability. We adopt the baseline figures reported by Shankarampeta et al. (2025) rather than reproducing that evaluation, so per-question predictions are not available for both systems, and our reliability checks accordingly operate at the level of the configuration grid. Table 2 provides the main one: the schema effect is significant across the eighteen configurations ($F(2, 10) = 21.1$, $p < 0.001$), while the query-model effect is not ($p = 0.97$). The ordering is also stable across the grid, with both Gemini 2.5 Flash and Llama-3.3-70B schemas outscoring Llama-3.1-8B schemas for *all six* query models (sign test, $p = 0.016$ one-sided). Small differences between adjacent cells are correspondingly less informative, and we interpret them with caution.

A newer, out-of-family schema generator behaves like the weakest. The generators in Table 1 come from two model families, so the ranking could be an artifact of that choice. We therefore used GPT-5.4-mini to synthesize a fourth set of schemas, changing nothing else, and queried them with three of the six query models (Appendix Table 11). Although it is the most recent generator, its schemas behave like the weakest, landing within 2.7 EM of the corresponding Llama-3.1-8B schema for all three and 5.2 to 19.6 points below Gemini 2.5 Flash. One generator crossed with three query models is a probe rather than a systematic sweep: it disconfirms the reading that a more recent generator yields better schemas, but cannot say which properties matter.

High cross-model schema portability. The Llama-3.3-70B-Instruct schema is the most portable of the three, holding F1 above 73 across all six query models and varying by only 6.8 EM points between its best and worst querier against 12.1 for Flash: well-normalized schemas capture temporal structure in a model-agnostic way. The baseline, by contrast, swings from 29.22 to 75.3 F1, with our approach falling within the 60.3-82.1 band. Llama-3.3-70B, as both schema generator and query model, reaches 79.52 EM / 79.91 F1, beating the strongest proprietary baseline by 10.6 EM and landing within 0.9 EM of the best cell: the benefits are a property of the framework, not of any proprietary model, and the pipeline runs end to end on open weights.

6 Analysis

6.1 Schema Design Principles

What makes a synthesized schema effective? We quantify structural properties of the generated schemas and relate them to downstream QA accuracy. For each domain, we compute four fragmentation metrics over the populated database: *completeness* (C, the fraction of base identifiers that participate in each fragment), *losslessness* (R, whether identifiers are reconstructible from unions of fragments), *referential consistency* (RC, whether all foreign keys resolve), and *disjointness* (D, the absence of duplicated fact units). Together with join depth and table counts, these metrics reveal concrete patterns underlying the performance differences:

Fragmentation and Join Complexity: Gemini 2.5 Flash generates the most compact schemas (5.5 tables/domain against 7.7 for Llama-3.1-8B and 6.2 for Llama-3.3-70B; Table 3, right) and the shallowest joins (1.089 average depth; left). Excessive fragmentation, notably in data-rich domains such as Economy (18 tables with Llama-3.1-8B against 2 with Flash), and high join depth (Government Agencies: 2.03-3.65) both track increased SQL generation errors, consistent with our balanced-normalization principle.

Fragmentation Quality: Tables 16-18 reveal that all schema generators maintain high referential consistency ($RC = 1.0$) and losslessness ($R \geq 0.96$ for most domains), but differ in completeness. Flash schemas achieve $C \geq 0.92$ in 6 of 10 domains, with the remaining deviations attributable to domain semantics (optional annotations, sparsely populated roles) rather than schema defects, while maintaining simpler structures that facilitate SQL generation.

Case Study: Balanced vs. Over-Normalized Schemas. Gemini-2.5-Pro schemas enter our structural analysis but were not swept across the query grid, so this is evidence about query complexity, not a measured accuracy gap. For *"Who was the coach when Balaji Pai was T20I captain?"* the Flash schema answers in 6 lines over 4 tables by joining role-specific tables on the shared snapshot key, while Pro routes both roles through generic `People/Roles` tables, forcing 15 lines over 6 tables including a self-join (Appendix E.1). Over-normalization is not wrong relationally, it is costly *generatively*: every additional join is another chance for the query model to err.

What these metrics do and do not certify. C, R, RC, and D are *intrinsic* measures, computed on the populated database independently of any question: they establish that the artifact is well formed and that no base identifier is lost under fragmentation. They are silent on semantic fidelity, saying nothing about whether each fact reaches the right relation and column, whether values survive cleaning undistorted, or whether a row is traceable to the revision that produced it. A schema could score perfectly while mis-assigning a role the benchmark never probes, and downstream accuracy, our only other check, rewards representations suited to this question distribution. Auditing fact-level mapping accuracy and provenance against sampled snapshots is the intrinsic evaluation that this line of work most needs (Limitations).

6.2 Error Analysis

Our best-performing variant (Gemini-2.5-Flash (schema) + Gemini-2.0-Flash (query)) answers 80.39% of test questions correctly, leaving approximately 19.6% of the test set as errors. To characterize failure modes, we analyzed a stratified random sample of 50 error cases drawn proportionally across the 10 domains. Table 4 reveals a surprising finding: **70% of errors stem from data understanding issues, not SQL generation or schema design.** See Appendix E for the erroneous examples.

Data Understanding Bottleneck: The primary failure mode involves: wrong calculations (tenure: 1148 vs 1113 days), entity variant mismatches

Domain	Avg. outer-join depth per query				Tables per domain			
	Flash	Pro	70B	8B	Flash	Pro	70B	8B
country	2.12	2.00	2.67	2.40	5	5	7	7
cricket_team	2.31	2.67	2.14	2.82	6	7	8	8
cricketer	0.00	0.00	0.00	0.00	3	4	3	7
economy	0.48	0.67	0.67	0.51	2	2	4	18
table_tennis_player	0.05	1.75	0.00	1.40	4	5	12	5
gov_agencies	2.03	1.20	3.65	2.03	7	3	6	5
equestrian	1.30	1.00	1.00	1.80	5	8	4	4
field_hockey	0.31	0.30	3.00	0.31	9	6	8	8
golfer	1.06	0.56	0.56	1.06	4	4	3	10
cyclist	1.23	1.89	1.89	1.43	10	9	7	5
Average	**1.089**	**1.204**	**1.558**	**1.376**	**5.5**	**5.3**	**6.2**	**7.7**

Table 3: Structural properties of the generated schemas (Flash = Gemini-2.5-Flash, Pro = Gemini-2.5-Pro, 70B = Llama-3.3-70B-Instruct, 8B = Llama-3.1-8B-Instruct). Left: average outer-join depth per executed query, from the outermost SELECT only. Right: relational tables generated per domain.

Issue (#Samples)	Error Category	Count
Data Quality (35)	Wrong Calculations	15
	Empty Results	12
	Wrong Entity Mapping	5
	Precision/Format Issues	3
SQL Generation (12)	Aggregate Function Misuse	6
	Syntax Errors	3
	Schema Column Errors	2
	Non-SQL Responses	1
Schema Generation (3)	Missing Data Handling	2
	Schema Misunderstanding	1

Table 4: Error distribution from 50 failure cases. Data quality dominates over SQL generation and schema issues.

("Paul Kihara Kariuki" vs "P.K. Kariuki"), and mapping errors (wrong captain retrieved).

SQL Generation Patterns: Aggregate function misuse dominates SQL errors (50%), following a consistent pattern: using `MIN()/MAX()/SUM()` in `ORDER BY` without proper `GROUP BY` clauses. This affects primarily temporal queries across the Country (3 cases) and Cricket Team (2 cases) domains.

Schema Robustness: Only 6% of errors trace to schema generation, validating our defensive parsing approach; the few failures involve missing columns, suggesting incomplete domain coverage during extraction. Rare does not mean harmless, and we return to that asymmetry in §7.

6.3 Cost and Efficiency

Schema synthesis is a one-time offline cost per domain: a single LLM call over 2-3 example timelines, after which the schema is reused for every question in that domain and adds no per-query latency. Few-shot exemplar construction is likewise offline and bounded by a single pass over the development questions. At inference time, the pipeline reduces to a single SQL generation call plus local SQLite execution (milliseconds). Notably, our best configuration relies on smaller, cheaper models (Gemini 2.5 Flash for schemas, Gemini-2.0-Flash for queries) than the strongest baseline (Gemini-2.5-Pro with IRE+CoT), yet scores 11.5 EM points higher, so the accuracy gains do not come at the price of a heavier stack.

Average prompt and completion tokens per SQL-generation call are nearly flat across schema sources (3752 to 4230 total, a 13% range) while the best EM attainable with each schema spans 15.6 points, so accuracy differences are not bought with inference budget; Appendix C gives the breakdown.

7 Discussion

Scope of the claims. Our design principles are hypotheses supported by controlled comparisons on one benchmark, grounded where possible in database theory: snapshot-anchored tables mirror SQL:2011 system-versioned temporal tables (Snodgrass, 1999), and semantic naming accords with schema-ambiguity findings in static text-to-SQL (Kohita, 2025).

Reading behavior off closed models. Two of our four schema generators are proprietary systems whose training is undisclosed, so explaining *why* their schemas differ risks reverse-engineering a black box. We keep our claims to what is observable: the schemas are inspectable artifacts, and join depth, table count, naming style, and temporal anchoring are properties of those artifacts rather than inferences about model internals. The central finding does not rest on closed models in any

case. The schema effect is just as clear among the open-weights generators (Llama-3.3-70B schemas at 77.21 EM on average against Llama-3.1-8B at 62.31), and the best open-weights configuration lands within 0.9 EM of the best overall, so both pipeline and conclusion are reproducible without proprietary access.

What would falsify the principles. Because generator identity, naming style, and table granularity co-vary in our data, the principles are hypotheses with a concrete test: perturb one fixed schema, one factor at a time, preserving information content, and read off the EM deltas. Limitations give the design; we regard it as the most valuable single addition to this work.

Fragility is real but asymmetric. A schema error cascades through every downstream query, a single point of failure that retrieval-based approaches do not share. Empirically such errors are rare (6% of failures; §6.2), and current LLMs prove strikingly reliable at producing referentially consistent structure ($RC = 1.0$ across all generators). But rare-and-catastrophic is a different risk profile from frequent-and-recoverable, which makes automated schema verification a high-value direction for future work (Appendix A.4).

Implications for knowledge base construction. The variance asymmetry suggests a division of labor for LLM-era knowledge base construction: invest model capability in *constructing* the representation, then exploit it with inexpensive query models. Well-normalized schemas transfer across all six query models we test, so the recipe is not tied to any proprietary system.

8 Conclusion

We presented EVINORM, a SQL-based approach to temporal reasoning over evolving semi-structured tables that synthesizes normalized schemas rather than assuming them. Across three schema generators and six query models on TRANSIENTTABLES, **the choice of schema source accounts for a far larger share of the variance in QA accuracy than the choice of query model**: 79.5% of the EM variance across the grid against 1.6%, or 14.7 to 20.0 points of swing from replacing the schema against 4.4 to 12.1 points from replacing the query model, though the margin narrows once the weakest generator is set aside (§5). The unit being varied is a schema together with the prompt scaffolding derived from it, and the evidence is associational, since generator identity, naming style, and table granularity co-vary. From this evidence, we distill three candidate design principles for temporal QA systems: balanced normalization outperforms excessive fragmentation; semantic naming reduces token-level confusion relative to generic abstractions; and consistent temporal anchoring enables pattern transfer across domains. We frame these as empirically supported hypotheses, grounded in database theory, in which snapshot-anchored tables mirror SQL:2011 system-versioned temporal tables, rather than as established universals: information-preserving schema perturbations would be needed to determine which property does the work. For automated knowledge base construction more broadly, our results suggest a division of labor: invest model capability in constructing the structured representation once, then exploit it with inexpensive query models.

Limitations

Attribution of the schema effect: Our few-shot exemplars, relationship descriptions, and SQL templates are derived from each schema, so replacing a schema replaces the entire prompt scaffolding built on it. The controlled comparison in Table 1 therefore measures the effect of a schema *together with its induced scaffolding*, under a fixed construction procedure, and cannot separate the contribution of relational structure from that of the surrounding prompt. Statements about the schema being the dominant lever should be read with that unit of comparison in mind. A zero-shot condition and a shared-few-shot condition, crossed with the three generators, would separate the two; we did not run them.

Correlational design principles: Normalization depth, naming style, temporal anchoring, and generator identity co-vary across the schemas we compare, so the three design principles are hypotheses consistent with our measurements rather than isolated causes. Establishing causality requires information-preserving perturbations of a single schema, varying one property at a time: renaming tables and columns to opaque identifiers while holding structure constant (semantic naming); merging or splitting relations while preserving the same functional dependencies (balanced normalization); and replacing the uniform `snapshot_id` with per-table date columns carrying identical information

(temporal anchoring). Each perturbation preserves the knowledge base's information content and changes only the property under test. Until such an ablation exists, the principles should be treated as design heuristics with empirical support on one benchmark, not as established results.

Schema-generation variability: Each generator produces one schema per domain in a single call, decoded at temperature 0.1, and that schema is reused for every entity and question in the domain. We therefore report between-generator differences and do not measure how much a generator's schema varies across independent runs of generation. Because we argue that the schema matters a great deal, run-to-run variance is exactly the quantity a follow-up should report: repeated synthesis per domain, with the resulting spread in downstream EM, would show whether the gaps in Table 1 are stable properties of a generator or partly draw from a wide distribution.

No intrinsic evaluation of semantic fidelity: Our structural metrics (completeness, losslessness, referential consistency, disjointness) certify that the constructed knowledge base is well-formed, but not that it faithfully preserves the source. We do not measure fact-level mapping accuracy, value fidelity under cleaning, or provenance from individual snapshots; downstream QA accuracy serves as a proxy that rewards representations suited to this benchmark's question distribution (§6.1).

Single benchmark evaluation: Our empirical validation is confined to the TRANSIENTTABLES benchmark. To our knowledge, it is the only publicly available benchmark of multi-snapshot temporal semi-structured tables; repurposing single-snapshot or static-KG temporal QA datasets would eliminate the state-change signal our method exploits. Nevertheless, validation across additional sources (e.g., corporate databases, temporal knowledge graphs, or other domain-specific temporal QA tasks) would strengthen generalizability claims. We also do not compare against querying an existing curated temporal knowledge base, such as Wikidata or YAGO2, directly; an alternative that would be cheaper to deploy where it applies, measuring how much of TRANSIENTTABLES is answerable from those sources is left to future work (§2).

A comparatively clean data source: Wikipedia infoboxes are curated, actively maintained, and already reasonably regular, which makes them a favorable setting for schema synthesis. Real deployments, particularly in industry, involve missing fields, inconsistent or undocumented schemas, contradictory records, and structures that change more often and less predictably than an infobox does. Our population layer handles a catalog of concrete inconsistencies (Appendix A.6), but that catalog was assembled against this source. How far synthesis quality degrades as the source gets noisier is an open question our experiments cannot answer, and the 70% of residual errors we attribute to data understanding (§6.2) suggests the degradation could be steep.

Single-domain restriction: Our current implementation limits queries to a single entity domain (e.g., only cricket teams or only countries). Cross-domain queries like "Who was India's cricket captain when GDP growth exceeded 7%?" would require joining schemas from different domains, introducing significant computational and modeling complexity.

Schema generation dependency: Performance is highly sensitive to schema quality. A single normalization error, such as failing to create a bridge table for a many-to-many relationship, cascades through all queries using that schema. This creates a single point of failure that is absent in retrieval-based approaches.

No handling of uncertainty: Our approach generates deterministic SQL queries that return exact answers. However, many temporal questions involve uncertainty (e.g., "approximately when did X begin?"). SQL's boolean logic cannot naturally express probabilistic or fuzzy temporal boundaries, forcing binary decisions where gradual transitions would be more appropriate.

Wikipedia-specific data assumptions: Our data cleaning pipeline is tuned for Wikipedia infobox inconsistencies (date format variations, nested arrays, key naming variations). Applying this approach to other semi-structured sources (corporate databases, government records, scientific datasets) would require substantial re-engineering of the cleaning logic.

Ethical considerations

This study uses only publicly available Wikipedia infobox timelines from the TransientTables bench-

mark and involves no human subjects, private data, or sensitive attributes; all entities correspond to public figures or institutions. Our SQL-based framework for temporal reasoning produces verifiable, traceable outputs that minimize subjective or value-laden interpretations, while the use of normalized schemas constrains model generations and mitigates bias. Although large language models (e.g., Gemini and Llama variants) may inherit pretraining biases, our system's reliance on deterministic SQL execution limits the propagation of such biases. Potential downstream impacts include misinterpretation or overreliance on automated schema designs when applied to high-stakes domains such as healthcare or finance. To mitigate these risks, we recommend human validation, transparent documentation of schema transformations, and maintenance of audit logs for all generated queries. All experiments were conducted on open data in accordance with the ACL Ethics Policy, and any released code or derived data will be distributed solely for research purposes under terms that encourage transparency and responsible reuse.

References

James F. Allen. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*, ISWC'07/ASWC'07, page 722–735, Berlin, Heidelberg. Springer-Verlag.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Irwin Deng, Kushagra Dixit, Dan Roth, and Vivek Gupta. 2025. Enhancing temporal understanding in LLMs for semi-structured tables. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4951–4970, Albuquerque, New Mexico. Association for Computational Linguistics.

Xi Fang, Weijie Xu, Fiona Anting Tan, Ziqing Hu, Jiani Zhang, Yanjun Qi, Srinivasan H. Sengamedu, and Christos Faloutsos. 2024. Large language models (LLMs) on tabular data: Prediction, generation, and understanding - a survey. *Transactions on Machine Learning Research*.

Yujian Gan, Xinyun Chen, and Matthew Purver. 2021. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8926–8931, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Vivek Gupta, Pranshu Kandoi, Mahek Vora, Shuo Zhang, Yujie He, Ridho Reinanda, and Vivek Srikumar. 2023. TempTabQA: Temporal question answering for semi-structured tables. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2431–2453, Singapore. Association for Computational Linguistics.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456, Online. Association for Computational Linguistics.

Ryosuke Kohita. 2025. Exploring database normalization effects on sql generation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, page 5788–5796, New York, NY, USA. Association for Computing Machinery.

Atharv Kulkarni, Kushagra Dixit, Vivek Srikumar, Dan Roth, and Vivek Gupta. 2025. LLM-symbolic integration for robust temporal tabular reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19914–19940, Vienna, Austria. Association for Computational Linguistics.

Atharv Kulkarni and Vivek Srikumar. 2025. Reinforcing code generation: Improving text-to-sql with execution-based learning. *arXiv preprint arXiv:2506.06093*.

Abhinav Lalwani, Tasha Kim, Lovish Chopra, Christopher Hahn, Zhijing Jin, and Mrinmaya Sachan. 2025. Autoformalizing natural language to first-order logic: A case study in logical fallacy detection. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 132–147, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin SU, ZHAOQING SUO, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2025. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *The Thirteenth International Conference on Learning Representations*.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024. Rethinking tabular data understanding with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 450–482, Mexico City, Mexico. Association for Computational Linguistics.

Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. 2025. A survey of text-to-sql in the era of llms: Where are we, and where are we going? *IEEE Transactions on Knowledge and Data Engineering*, 37(10):5735–5754.

Sumit Neelam, Udit Sharma, Hima Karanam, Shajith Ikbal, Pavan Kapanipathi, Ibrahim Abdelaziz, Nandana Mihindukulasooriya, Young-Suk Lee, Santosh Srivastava, Cezar Pendus, Saswati Dana, Dinesh Garg, Achille Fokoue, Shrivatsa Bhargav, Dinesh Khandelwal, Srinivas Ravishankar, Sairam Gurajada, Maria Chang, Rosario Uceda-Sosa, Salim Roukos, Alexander Gray, Guilherme Lima, Ryan Riegel, Francois Luus, and L Venkata Subramaniam. 2022. Tempqa-wd: A benchmark for generalizable and interpretable temporal question answering over knowledge bases. *arXiv preprint arXiv:2201.05793*.

Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. 2020. TORQUE: A reading comprehension dataset of temporal ordering questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1158–1172, Online. Association for Computational Linguistics.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Aniket Pramanick and Indrajit Bhattacharya. 2021. Joint learning of representations for web-tables, entities and types using graph convolutional network.

In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1197–1206, Online. Association for Computational Linguistics.

Apoorv Saxena, Soumen Chakrabarti, and Partha Talukdar. 2021. Question answering over temporal knowledge graphs. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6663–6676, Online. Association for Computational Linguistics.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Abhilash Shankarampeta, Harsh Mahajan, Tushar Kataria, Dan Roth, and Vivek Gupta. 2025. TRANSIENTTABLES: Evaluating LLMs' reasoning on temporally evolving semi-structured tables. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6526–6544, Albuquerque, New Mexico. Association for Computational Linguistics.

Richard Thomas Snodgrass. 1999. *Developing time-oriented database applications in SQL*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ngoc Phuoc An Vo, Octavian Popescu, Irene Manotas, and Vadim Sheinin. 2022. Tackling temporal questions in natural language interface to databases. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 179–187, Abu Dhabi, UAE. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Yuqing Wang and Yun Zhao. 2024. TRAM: Benchmarking temporal reasoning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6389–6415, Bangkok, Thailand. Association for Computational Linguistics.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations*.

Jiaying Wu and Bryan Hooi. 2025. Chain-of-timeline: Enhancing LLM zero-shot temporal reasoning with SQL-style timeline formalization. In *Workshop on Reasoning and Planning for Large Language Models*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 174–184, New York, NY, USA. Association for Computing Machinery.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1962–1979, Hong Kong, China. Association for Computational Linguistics.

Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 1029–1032, New York, NY, USA. Association for Computing Machinery.

Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(2).

Shuo Zhang, Zhuyun Dai, Krisztian Balog, and Jamie Callan. 2020. Summarizing and exploring tabular data in conversational search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1537–1540, New York, NY, USA. Association for Computing Machinery.

A Appendix

A.1 A Worked Example, End to End

This section follows one question through all three stages of EVINORM, using the `cricket_team` domain and the schema synthesized by Gemini 2.5 Flash. The question is *"Who was the coach when Balaji Pai was T20I captain?"*, a concurrency question: it asks which value one attribute held at the time another attribute held a given value, so it cannot be answered from any single snapshot in isolation.

Input: an infobox timeline. The pipeline receives S_e, a sequence of timestamped infobox snapshots for one entity. Schematically, with values elided and keys abridged:

```
2015-03-12: {"coach": ..., "captain (T20I)":
    ...,
            "T20I rank": ..., "T20I record":
                ...}
2016-07-04: {"coach": ..., "captain (T20I)":
    ...,
            "T20I rank": ..., "T20I record":
                ...}
2018-01-22: {"coach": ..., "captain (T20I)":
    ...,
            "T20I rank": ..., "T20I record":
                ...}
```

The same keys recur across timestamps while their values change, and keys are occasionally added or dropped between revisions. This is the state-change signal the schema must capture.

Stage 1: schema synthesis. A single prompted call over 2–3 such timelines returns a 3NF schema for the whole domain. The four tables relevant to this question are reproduced below (the full schema is listed in Appendix F.2):

```
Coaches (coach_id PK, coach_name)
Players (player_id PK, player_name)
CaptainRoles (captain_role_id PK, role_title)
TeamSnapshots (snapshot_id PK, team_id FK,
               coach_id FK, t20i_rank, ...)
TeamCaptaincy (snapshot_id FK, player_id FK,
               captain_role_id FK)
```

Three design decisions matter here. Time-varying attributes live in `TeamSnapshots`, keyed by `snapshot_id`; the many-to-many relation between players and captaincy roles at a point in time is factored into the bridge table `TeamCaptaincy`; and both tables are anchored on the *same* `snapshot_id` vocabulary, which is what makes "at the same time as" expressible as an equality join rather than an interval comparison.

Stage 2: population. Automated scripts parse the `CREATE` statements, clean each snapshot's values (date normalization, entity-name normalization, type coercion, null handling), and insert rows, rejecting any row that violates a declared foreign key. Each infobox revision becomes one `TeamSnapshots` row plus zero or more `TeamCaptaincy` rows.

Stage 3: SQL generation and execution. The domain prompt supplies the full schema, natural-language descriptions of the relationships, 8–10 SQL templates, 10–15 execution-validated exemplars, and the critical rules listed in §3. The query model then emits:

```
SELECT DISTINCT c.coach_name
FROM Coaches c
  JOIN TeamSnapshots ts ON c.coach_id = ts.
      coach_id
  JOIN TeamCaptaincy tc ON ts.snapshot_id = tc.
      snapshot_id
  JOIN Players p ON tc.player_id = p.player_id
  JOIN CaptainRoles cr
      ON tc.captain_role_id = cr.
          captain_role_id
WHERE cr.role_title = 'T20I Captain'
  AND p.player_name = 'Balaji Pai'
  AND ts.team_id = (SELECT team_id FROM
      CricketTeams
                  WHERE team_name =
                  'Gibraltar national cricket
                      team');
```

The temporal condition is carried entirely by `ts.snapshot_id = tc.snapshot_id`: coach and captain are required to be recorded in the same revision. The query is executed against the populated SQLite database and its result is compared to the reference answer under Exact Match, token-level F1, Rouge-1, and Rouge-L. Appendix E.1 shows what the same question costs under a more heavily normalized schema, where both roles route through generic `People`/`Roles` tables and the query needs a self-join over personnel.

A.2 Temporal Coverage: An Allen-Interval View

Allen's interval algebra (Allen, 1983) defines 13 basic relations between time intervals and provides a natural yardstick for the temporal scope of our approach. Our few-shot SQL templates compile into predicates over snapshot ranges that natively express 11 of the 13 relations: *before*/*after* map to strict inequalities on snapshot bounds (e.g., `s1.end < s2.start`), *meets*/*met-by* to boundary equality (`s1.end = s2.start`), and *overlaps*, *during*, *starts*, *finishes* and their inverses to conjunctions

of <, =, > over interval endpoints. The remaining relations *equals* on closed intervals and strict containment are handled via window functions or LEAD/LAG, which our templates employ for tenure and succession questions. This mapping makes the theoretical scope of the approach explicit: deterministic SQL over snapshot-anchored schemas covers the full qualitative interval calculus, whereas *metric* vagueness ("around 2020", "early career") lies outside Boolean semantics (see Limitations).

A.3 Domain-Generality of the Pipeline

The pipeline contains no benchmark-specific machinery. Schema synthesis consumes only 2–3 example timelines and no Wikipedia-specific heuristics; few-shot exemplars are constructed automatically via execution-based validation rather than manual curation, making them reproducible for any new domain at the cost of a small development set; and the population layer's defensive strategies (null normalization, type-conversion fallbacks, constraint-violation handling) are standard for any semi-structured source. The main adaptation cost for a new data source is the set of composite-field parsers described in Appendix A.6.

A.4 Future Works

Cross-domain temporal reasoning: Extending our approach to support queries spanning multiple entity domains (e.g., "What was India's GDP when Virat Kohli became captain?") requires developing schema integration techniques and multi-database join strategies while maintaining temporal consistency across heterogeneous schemas.

Generalization beyond Wikipedia: Adapting our schema generation and data cleaning pipeline to corporate databases, government records, and scientific datasets would require domain-agnostic normalization strategies and automated schema quality assessment tools.

Automated schema optimization: Developing methods to automatically evaluate and refine schema designs based on query performance patterns could reduce the dependency on initial schema quality and enable adaptive normalization strategies.

A.5 3NF Validation

We implement automated Python scripts to create and populate SQLite databases from the LLM-generated schemas. The pipeline includes: (1) parsing SQL CREATE statements to instantiate tables, (2) cleaning JSON data to handle inconsistencies in Wikipedia infoboxes (e.g., date format variations, text normalization, nested structure parsing), (3) executing INSERT statements with proper type conversion, and (4) validating referential integrity constraints. Semi-structured JSON infoboxes exhibit high variability across domains, for example, handling inconsistencies such as "Prime Minister" vs. "PM", malformed arrays, and missing values, necessitating defensive parsing and cleaning strategies

Given a set of JSON-formatted Wikipedia infobox snapshots for an entity (typically 8-12 temporal samples), we generate a normalized relational schema using LLMs. We provide the model with 2-3 example JSON timelines from the same domain and explicitly request a Third Normal Form (3NF) schema, in line with standard database normalization principles.

For example, rather than storing "Prime Minister: Angela Merkel" in 20 consecutive snapshots, we normalize to three tables: `Leaders(leader_id, leader_name)`, `LeaderRoles(role_id, role_title)`, and `SnapshotLeaders(snapshot_id, leader_id, role_id)`. This structure explicitly represents temporal state changes through foreign key relationships, enabling SQL's temporal operators (date comparisons, temporal joins, and window functions) to express complex temporal reasoning patterns.

Ensuring 3NF Compliance. We employ a three-step validation process:

(1) Explicit prompting: Our schema generation prompt explicitly instructs: *"Generate a 3NF schema where: (a) all attributes are atomic (no multi-valued fields), (b) all non-key attributes are fully functionally dependent on the primary key, and (c) no transitive dependencies exist between non-key attributes."*

(2) Manual structural inspection: Each generator emits one schema per domain, so ten schemas per generator, all of which we inspected; within them we audited 50 randomly sampled tables per generator in detail against the 3NF criteria:

- *Atomicity*: No composite values (e.g., "Name (Role)" split into separate columns)
- *Full dependency*: All attributes depend on the entire primary key, not a subset

- *No transitive dependencies*: No attributes $A \rightarrow B \rightarrow C$ chains where C depends on A through B

(3) Downstream performance validation: Although we do not formally prove 3NF algorithmically, this multifaceted validation, combining explicit prompting, structural analysis, use of foreign key constraints, and empirical SQL generation performance, provides strong evidence of normalization quality.

A.6 Database Population and Error Handling

We implement automated Python scripts to create and populate SQLite databases from the LLM-generated schemas. The pipeline handles multiple error categories that arise when mapping semi-structured Wikipedia data to normalized relational schemas:

A.6.1 Data Type Conversion Errors

Wikipedia infoboxes contain numeric values with formatting inconsistencies (e.g., "1,234", "1.0", "∼100"). Direct conversion to SQL integer/float types fails. We implement a safe integer parser that:

- Strips commas and whitespace
- Converts to float first (handles "1.0" → 1)
- Returns `None` safely on failure, preventing insertion crashes

A.6.2 Missing and Invalid Data Normalization

Infoboxes use multiple representations for missing data ("n/a", "–", "vacant", "–", blank). We normalize null values to standardize these variants to SQL `NULL` before parsing, ensuring consistent handling.

A.6.3 Database Constraint Violations

Duplicate entries violate UNIQUE constraints during insertion. We implement defensive insertion:

```
try:
    cursor.execute("INSERT INTO Leaders
        ...", values)
except sqlite3.IntegrityError:
    # Fallback: retrieve existing row ID
    cursor.execute("SELECT id FROM
        Leaders
                    WHERE name=?", (name
                        ,))
    return cursor.fetchone()[0]
```

This prevents crashes while maintaining referential integrity.

A.6.4 Complex JSON Structure Parsing

Wikipedia infoboxes contain domain-specific composite fields that resist standard parsing:

- Combined statistics: "100s/50s": "2/22" (centuries/half-centuries)
- Performance metrics: "best bowling": "6/35" (wickets/runs)
- Special characters: "catches/stumpings": "31/–"
- Dynamic fields: `proyears1`, `proteam1`, ... with embedded ranges

We implement domain-specific parsers that:

1. Normalize null values.
2. Use regex-based extractors for composite fields.
3. Handle dynamic field names with pattern matching.

Rationale. JSON's semi-structured nature allows field structures to vary across domains and even within records. This unpredictability makes direct schema mapping prone to errors. Our robust parsing strategies like defensive fallbacks, domain-specific extractors, and standardization layers handle this variability.

B Domain-wise metrics for Query generation models

This section presents domain-wise performance metrics for various query-generation models evaluated across different generated schemas. Tables 5 and 6 detail the performance of the query generation models using the schema generated by Gemini 2.5 Flash. Tables 7 and 8 present the evaluation of models utilizing the schema generated by Llama-3.3-70B-Instruct. Tables 9 and 10 report the results for query generation models when evaluated against the schema produced by Llama-3.1-8B-Instruct. Finally, Table 11 reports the out-of-family robustness check discussed in §5, in which schemas are synthesized by GPT-5.4-mini and queried by three of the six query models.

C Token Usage for SQL Query Generation

Table 12 reports average prompt and completion tokens per SQL-generation call (per-model, per-domain counts in Appendix C). Inference cost is

Domain	Gemini-2.0-Flash				Llama-3.3-70B-Instruct				GPT-4o-mini			
	EM	F1	R-1	R-L	EM	F1	R-1	R-L	EM	F1	R-1	R-L
country	80.00	84.30	70.84	70.26	66.50	78.44	48.67	48.13	81.50	63.78	67.59	66.38
cricket_team	82.70	80.00	67.99	66.58	75.61	84.50	65.00	60.27	79.00	55.72	61.80	60.66
cricketer	78.70	82.60	85.79	85.79	64.00	68.00	58.30	58.91	72.00	87.80	87.80	87.80
economy	84.30	81.40	46.87	46.87	62.57	73.12	54.12	54.12	82.10	81.50	81.50	81.50
table_tennis_player	76.70	78.30	49.60	48.30	80.21	78.10	67.23	68.91	65.00	60.67	61.00	61.00
gov_agencies	81.30	82.00	73.05	72.22	68.51	79.51	58.19	55.23	45.00	47.00	47.00	43.00
equestrian	85.30	75.70	55.05	55.05	78.22	79.81	65.76	66.74	66.00	68.00	68.00	68.00
field_hockey	79.27	81.77	81.82	81.73	72.14	74.66	59.01	59.20	77.00	77.00	77.00	72.37
golfer	81.44	89.77	60.82	60.82	68.35	71.24	61.54	62.98	62.00	62.00	61.00	61.00
cyclist	74.23	85.21	69.02	69.02	62.45	68.01	60.91	62.10	66.00	64.50	64.50	64.50
Avg	**80.39**	**82.11**	**66.09**	**65.66**	**69.81**	**75.54**	**59.87**	**59.66**	**69.56**	**66.80**	**67.72**	**66.62**

Table 5: Query Generation Models: Performance Comparison of Query Generation Models (Gemini-2.0-Flash, Llama-3.3-70B-Instruct, GPT-4o-Mini) across various domains. The models' query generation quality is evaluated using Exact Match (EM), F1-score (F1), R-1 (Rouge-1), and R-L (Rouge-L) metrics. The schema used for the queries was generated using Gemini 2.5 Flash.

Domain	Gemini-2.5-Pro				Qwen-2.5-7B-Instruct				Llama-3.1-8B-Instruct			
	EM	F1	R-1	R-L	EM	F1	R-1	R-L	EM	F1	R-1	R-L
country	66.00	78.43	78.43	75.23	78.14	81.24	65.41	65.18	76.24	80.11	66.54	64.78
cricket_team	87.00	78.34	78.34	75.30	79.92	77.00	62.13	61.35	78.31	77.10	61.98	61.23
cricketer	60.50	60.50	60.50	60.50	76.50	79.50	80.95	80.89	78.20	80.54	78.41	77.69
economy	80.50	60.00	60.00	60.00	81.00	77.93	42.17	42.08	80.15	78.46	50.35	52.71
table_tennis_player	50.00	48.83	48.83	48.83	74.50	75.50	44.42	43.46	76.40	77.23	47.56	46.79
gov_agencies	63.50	61.29	70.50	66.61	78.50	79.50	67.09	66.55	79.48	80.10	63.90	64.02
Equestrian	61.50	69.30	69.30	68.07	82.50	73.12	50.11	50.01	81.24	75.47	52.43	54.18
field_hockey	76.00	77.17	78.00	78.00	76.91	78.20	76.54	76.28	77.82	79.46	79.23	78.11
golfer	63.50	63.50	63.50	63.50	78.33	85.43	55.19	55.02	79.18	84.22	56.79	57.71
cyclist	74.00	74.46	75.00	75.00	72.09	81.23	63.63	63.59	73.81	82.18	61.36	60.16
Avg	**68.25**	**67.18**	**68.24**	**67.10**	**77.84**	**78.87**	**60.76**	**60.44**	**78.08**	**79.49**	**61.86**	**61.74**

Table 6: Performance Comparison of three Query Generation Models (Gemini-2.5-Pro, Qwen-2.5-7B-Instruct, and Llama-3.1-8B-Instruct) across diverse domains. The models' query generation quality is evaluated using Exact Match (EM), F1-score (F1), R-1 (Rouge-1), and R-L (Rouge-L) metrics. Llama-3.1-8B-Instruct demonstrated the highest average performance with an F1 score of 79.49 and an EM score of 78.08. The schema used for the queries was generated using Gemini 2.5 Flash.

Domain	Llama-3.1-8B-Instruct				Qwen 2.5 7B Instruct				Llama-3.3-70B-Instruct			
	EM	F1	R-1	R-L	EM	F1	R-1	R-L	EM	F1	R-1	R-L
country	82.25	79.22	65.71	68.24	78.34	76.22	65.93	67.12	81.17	81.80	65.18	62.86
cricket_team	81.97	80.03	66.29	67.12	76.99	74.56	63.88	65.45	77.52	79.28	62.70	62.52
gov_agencies	80.89	81.10	63.98	66.52	73.11	74.55	61.99	62.99	80.05	81.79	65.32	66.95
economy	81.99	78.89	64.91	67.77	76.45	74.22	63.22	65.11	78.80	77.57	66.37	64.28
table_tennis_player	80.45	76.99	62.88	65.55	74.28	75.10	62.37	63.49	78.80	78.09	63.29	65.97
cricketer	81.56	78.78	64.55	67.33	77.87	75.91	64.99	66.88	80.72	79.14	64.29	63.01
Equestrian	73.66	75.19	64.23	66.90	75.55	76.88	64.21	65.52	78.97	81.72	66.31	66.03
field_hockey	78.28	79.05	62.77	65.41	72.99	74.01	61.22	62.55	79.05	81.41	62.74	63.89
golfer	74.01	75.99	64.87	67.55	74.88	76.22	63.45	64.88	79.55	80.23	64.81	64.58
cyclist	71.99	73.23	61.99	64.55	72.54	73.01	60.85	62.11	80.55	78.07	62.80	62.29
Avg	**78.71**	**77.85**	**64.22**	**66.69**	**75.30**	**75.07**	**63.21**	**64.61**	**79.52**	**79.91**	**64.38**	**64.24**

Table 7: Performance Evaluation of three Query Generation Models (Llama-3.1-8B-Instruct, Qwen-2.5-7B-Instruct, and Llama-3.3-70B-Instruct) across diverse domains. The models' query generation quality is assessed using Exact Match (EM), F1-score (F1), R-1 (Rouge-1), and R-L (Rouge-L) metrics. Llama-3.3-70B-Instruct achieved the highest overall average scores, leading with an average F1 of 79.91 and an average EM of 79.52. The schema used for these queries was generated by Llama-3.3-70B-Instruct.

Domain	GPT-4o-mini				Gemini-2.5-pro				Gemini-2.0-Flash			
	EM	**F1**	**R-1**	**R-L**	**EM**	**F1**	**R-1**	**R-L**	**EM**	**F1**	**R-1**	**R-L**
country	79.44	80.95	64.69	64.10	79.88	77.00	64.37	60.81	71.91	74.15	59.57	60.33
cricket_team	80.07	79.28	64.37	64.72	76.59	78.86	62.22	62.90	73.33	72.33	57.53	57.65
cricketer	80.06	77.26	65.90	60.58	80.04	77.96	66.06	61.75	75.55	75.05	58.61	60.31
economy	77.53	76.60	65.35	61.71	81.87	79.79	63.38	66.90	72.14	70.26	61.23	59.77
table_tennis_player	77.06	79.42	61.36	64.01	78.37	77.74	63.64	64.29	71.83	72.39	61.27	58.92
gov_agencies	75.64	78.04	60.67	66.20	77.40	80.10	60.84	65.78	74.23	75.97	58.58	58.77
Equestrian	78.02	78.46	64.39	65.06	78.07	75.42	64.22	63.82	72.14	74.58	57.86	61.15
field_hockey	77.98	77.19	60.11	64.16	77.79	80.17	62.40	62.13	73.56	74.15	58.69	59.76
golfer	77.98	81.25	63.90	61.11	76.28	76.23	62.92	64.77	70.91	72.39	59.89	59.61
cyclist	81.61	78.51	61.07	64.87	78.72	78.72	66.13	63.57	71.45	72.06	57.01	61.94
Avg	**78.54**	**78.70**	**63.18**	**63.65**	**78.50**	**78.20**	**63.62**	**63.67**	**72.71**	**73.33**	**59.02**	**59.82**

Table 8: Query Generation Model Performance for GPT-4o-mini, Gemini-2.5-Pro, and Gemini-2.0-Flash evaluated across multiple domains. Both GPT-4o-mini and Gemini-2.5-Pro demonstrated strong, competitive results, with less than 1 point separating their average Exact Match (EM) and F1 scores. Gemini-2.0-Flash showed a lower average performance across all four metrics (EM, F1, R-1, R-L). The schema for generating these queries was derived from Llama-3.3-70B-Instruct.

Domain	Llama-3.1-8B-Instruct				Qwen-2.5-7B-Instruct				Gemini-2.5-pro			
	EM	**F1**	**R-1**	**R-L**	**EM**	**F1**	**R-1**	**R-L**	**EM**	**F1**	**R-1**	**R-L**
country	64.82	62.70	52.92	50.01	62.85	62.05	57.08	54.78	61.27	64.93	52.81	50.45
cricket_team	59.15	64.89	44.34	47.91	58.82	64.47	49.19	52.42	65.84	60.11	49.33	54.19
gov_agencies	62.43	57.01	46.87	43.12	59.80	64.44	49.54	48.32	60.55	63.89	53.94	52.37
economy	57.98	61.21	43.55	45.68	56.01	60.80	46.70	49.83	62.41	61.56	54.91	51.58
table_tennis_player	64.50	58.12	47.10	46.05	62.57	57.10	50.78	50.18	66.50	59.13	51.02	49.88
cricketer	60.33	63.78	45.22	44.76	63.08	56.13	49.30	46.17	59.90	66.25	50.77	53.60
Equestrian	65.00	59.05	44.99	47.32	65.27	58.89	47.53	52.10	64.38	65.01	49.56	54.51
field_hockey	61.71	62.11	46.60	43.88	59.94	62.44	49.48	48.55	61.98	60.72	52.05	50.04
golfer	58.02	60.45	43.00	45.19	57.22	60.23	45.49	49.44	63.29	66.91	54.40	53.08
cyclist	63.88	57.30	47.75	44.44	63.51	56.63	51.99	49.11	65.17	62.15	50.18	51.79
Avg	**61.78**	**60.66**	**46.23**	**45.84**	**60.91**	**60.32**	**49.71**	**50.09**	**63.13**	**63.07**	**51.90**	**52.15**

Table 9: Performance comparison of Llama-3.1-8B-Instruct, Qwen-2.5-7B-Instruct, and Gemini-2.5-Pro using a schema generated by Llama-3.1-8B-Instruct. The models' average scores across all metrics (EM, F1, R-1, R-L) are significantly lower than in previous tables, suggesting a more challenging evaluation set. Gemini-2.5-Pro was the top performer, achieving the highest average F1 score of 63.07.

Domain	Llama-3.3-70B-Instruct				GPT-4o-mini				Gemini-2.0-Flash			
	EM	**F1**	**R-1**	**R-L**	**EM**	**F1**	**R-1**	**R-L**	**EM**	**F1**	**R-1**	**R-L**
country	67.82	65.70	59.92	57.01	60.03	64.07	53.53	52.66	60.59	58.12	49.16	52.18
cricket_team	62.15	67.89	51.34	54.91	63.50	59.45	49.72	54.88	62.81	61.03	50.57	51.54
cricketer	65.43	60.01	53.87	50.12	66.82	65.59	51.34	50.93	57.34	60.05	53.54	50.25
economy	60.98	64.21	50.55	52.68	62.80	61.08	54.09	53.31	59.22	62.17	51.05	52.78
table_tennis_player	67.50	61.12	54.10	53.05	59.76	63.03	52.50	49.07	61.47	57.88	50.14	48.29
gov_agencies	63.33	66.78	52.22	51.76	64.55	66.01	50.49	54.74	63.65	59.51	50.45	49.81
Equestrian	68.00	62.05	51.99	54.32	61.73	59.54	53.11	51.22	58.70	61.70	53.65	53.04
field_hockey	64.71	65.11	53.60	50.88	65.33	64.20	51.67	53.86	60.24	63.48	52.17	53.36
golfer	61.02	63.45	50.00	52.19	60.29	61.88	49.11	52.92	62.09	58.46	53.02	49.93
cyclist	66.88	60.30	54.75	51.44	63.96	66.70	54.67	50.69	57.61	60.85	53.46	50.70
Avg	**64.78**	**63.66**	**53.23**	**52.86**	**62.88**	**63.16**	**52.02**	**52.43**	**60.37**	**60.33**	**51.72**	**51.19**

Table 10: Query Generation Model Performance for Llama-3.3-70B-Instruct, GPT-4o-Mini, and Gemini-2.0-Flash. The evaluation uses the challenging schema generated by the Small Language Model (Llama-3.1-8B-Instruct). Despite the resultant drop in average scores, the larger Llama-3.3-70B-Instruct model successfully crossed previous performance baselines, leading the group with an average EM of 64.78 and F1 score of 63.66.

Domain	GPT-4o-mini				Llama-3.3-70B-Instruct				Qwen-2.5-7B-Instruct			
	EM	F1	R-1	R-L	EM	F1	R-1	R-L	EM	F1	R-1	R-L
country	68.50	79.03	80.38	76.85	64.50	79.00	79.88	76.35	61.50	73.66	77.88	74.60
cricket_team	52.50	60.22	60.25	56.76	65.00	73.06	73.25	69.76	53.00	60.22	60.25	57.01
cricketer	74.50	74.50	74.50	74.50	75.00	75.00	75.00	75.00	73.50	73.83	74.00	74.00
economy	47.00	51.50	51.50	49.46	55.00	60.50	62.00	58.50	44.50	49.00	49.00	47.00
table_tennis_player	51.50	54.21	57.50	57.50	53.00	55.81	57.00	57.00	43.00	46.06	47.50	47.50
gov_agencies	51.50	76.59	83.87	79.87	53.50	77.17	83.30	79.19	43.00	64.15	70.10	66.25
equestrian	78.50	78.83	79.00	79.00	84.00	84.33	84.50	84.50	69.50	69.83	70.00	70.00
field_hockey	62.00	82.00	85.00	82.00	64.00	82.00	85.00	82.00	55.00	73.00	76.00	73.00
golfer	73.00	75.00	75.00	75.00	75.00	76.00	76.00	76.00	65.00	67.00	67.00	67.00
cyclist	63.00	66.00	66.00	66.00	66.00	68.00	68.00	68.00	58.00	61.00	61.00	61.00
Avg	**62.20**	**69.79**	**72.30**	**67.49**	**65.50**	**73.04**	**74.76**	**72.63**	**55.60**	**63.78**	**65.27**	**63.74**

Table 11: Out-of-family schema check: performance of three query generation models (GPT-4o-mini, Llama-3.3-70B-Instruct, Qwen-2.5-7B-Instruct) across the ten domains, evaluated with Exact Match (EM), F1, ROUGE-1 (R-1), and ROUGE-L (R-L). GPT-5.4-mini is used only to synthesize the database schema that these queries are written against; it generates no queries itself. This is a partial sweep: only three of the six query models in Table 1 were run against this schema. Average EM nonetheless lands within 2.7 points of the Llama-3.1-8B schemas for every query model tested, despite the schema generator being a substantially more recent model.

Schema source	In	Out	Total	Best EM
Gemini 2.5 Flash	4040	116	4156	80.39
Llama-3.3-70B	4119	111	4230	79.52
Llama-3.1-8B	3598	154	3752	64.79

Table 12: Average input and output tokens per SQL-generation call by schema source, averaged over six query models and ten domains, alongside the best EM achieved with that schema. A 13% spread in token cost accompanies a 15.6-point spread in accuracy.

nearly flat across schema sources, spanning 3752 to 4230 total tokens, a 13% range, while the best EM attainable with each schema spans 15.6 points: accuracy differences are not bought with inference budget. The two components move in opposite directions. Llama-3.1-8B schemas serialize compactly, giving the shortest prompts but the longest queries (154 output tokens against 111 to 116), consistent with their heavier fragmentation (7.7 tables per domain against 5.5 for Flash; Table 3): more tables to join means more SQL to write, and more surface for the errors catalogd in §6.2.

This section reports the per-call token accounting summarized in §6.3. For every combination of schema source, query model, and domain, we log the prompt (In) and completion (Out) tokens of the single SQL-generation call that each test question requires; values are averaged over all questions in the domain. The prompt consists of the serialized schema, the few-shot exemplars, and the question, so its length is governed largely by how verbose the synthesized schema is: within a fixed schema source, the spread across domains (roughly 2.1K tokens for Equestrian to 9.0K for Government Agencies) is far wider than the spread across query models. Tables 13, 14, and 15 report these counts for schemas produced by Gemini 2.5 Flash, Llama-3.3-70B-Instruct, and Llama-3.1-8B-Instruct respectively. Schema synthesis and few-shot exemplar construction are excluded because they are one-time offline costs amortized over every question in a domain.

D Domain-wise Fragmentation Metrics for Different Schema Generation Models

Fragmentation Metrics: Completeness (C) measures the fraction of base dimension or fact identifiers participating in each fragment. Losslessness (R) evaluates whether all identifiers are reconstructible from fragment unions. Referential Consistency (RC) verifies that all foreign-key references point to valid parent tuples. Disjointness (D) checks whether fragments introduce duplicate fact units.

Table 16 details the fragment metrics for the schema generated using Gemini 2.5 Flash. Table 17 presents the fragment metrics for the schema created by Llama-3.1-8B-Instruct. Finally, Table 18 reports the evaluation of fragmentation metrics for the schema generated by Llama-3.3-70B-Instruct.

Domain	Gemini-2.0-Flash		Llama-3.3-70B-Instruct		GPT-4o-mini		Gemini-2.5-Pro		Qwen-2.5-7B-Instruct		Llama-3.1-8B-Instruct	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
country	3155	100	3571	113	3214	102	3603	114	3286	104	3518	111
cricket_team	4450	119	4991	133	4435	119	5057	135	4388	117	4582	122
cricketer	4220	146	4774	165	4279	148	4736	163	4369	151	4782	165
economy	3327	79	3020	98	3427	101	3061	102	3010	101	3389	98
table_tennis_player	2857	59	2537	99	2546	103	2804	103	2863	102	2474	99
gov_agencies	8029	103	8997	116	8785	113	8956	115	7977	103	8937	115
equestrian	2354	108	2287	105	2299	106	2147	105	2068	106	2296	105
field_hockey	4095	93	3997	100	3709	104	3542	107	3654	107	4063	100
golfer	3419	153	3386	152	3390	152	3171	142	3029	136	3040	136
cyclist	3916	132	4304	125	3804	112	3766	110	3884	114	4386	128
Avg	**3982.2**	**109.2**	**4186.4**	**120.6**	**3988.8**	**116.0**	**4084.3**	**119.6**	**3852.8**	**114.1**	**4146.7**	**117.9**

Table 13: Average input and output token counts per SQL-generation call across six query models and ten domains, using Gemini 2.5 Flash as the schema generator.

Domain	Gemini-2.0-Flash		Llama-3.3-70B-Instruct		GPT-4o-mini		Gemini-2.5-Pro		Qwen-2.5-7B-Instruct		Llama-3.1-8B-Instruct	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
country	3475	110	3603	112	3576	103	3710	107	3517	103	3239	108
cricket_team	4864	130	4802	125	4693	126	4966	124	5175	139	4966	129
cricketer	4636	160	4751	171	4618	168	4501	168	4565	171	4704	149
economy	3286	78	3253	82	3181	73	3515	75	3104	83	3103	79
table_tennis_player	2737	57	2559	57	2890	60	2679	53	2610	60	2605	53
gov_agencies	8697	112	8334	112	9027	105	8325	110	8913	119	8163	114
equestrian	2264	104	2299	109	2188	99	2271	105	2201	106	2410	104
field_hockey	3905	89	3918	94	4132	84	3823	85	4146	84	3635	85
golfer	3352	150	3271	142	3177	144	3378	147	3204	151	3437	156
cyclist	4184	122	4163	126	4354	117	4093	130	3969	114	4012	121
Avg	**4140.0**	**111.2**	**4095.3**	**113.0**	**4183.6**	**107.9**	**4126.1**	**110.4**	**4140.4**	**113.0**	**4027.4**	**109.8**

Table 14: Average input and output token counts per SQL-generation call across six query models and ten domains, using Llama-3.3-70B-Instruct as the schema generator.

Domain	Gemini-2.0-Flash		Llama-3.3-70B-Instruct		GPT-4o-mini		Gemini-2.5-Pro		Qwen-2.5-7B-Instruct		Llama-3.1-8B-Instruct	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
country	3422	139	3187	140	3376	133	3198	136	3344	131	3486	146
cricket_team	4679	120	4452	112	4633	117	4971	126	4711	123	4652	121
cricketer	3370	142	3219	133	3186	148	3530	134	3299	133	3514	137
economy	3180	150	3025	154	3260	153	3272	156	3081	157	3079	153
table_tennis_player	3716	131	3835	128	3915	129	3877	136	3503	139	3743	132
gov_agencies	3490	180	3564	179	3431	178	3314	172	3368	173	3409	191
equestrian	2264	201	2264	212	2234	206	2357	208	2141	215	2197	206
field_hockey	3905	164	3802	172	3635	163	4120	163	4124	160	4120	162
golfer	3966	170	4072	174	3985	166	4182	161	3741	174	3719	163
cyclist	4184	137	3895	129	3904	141	4452	144	4411	128	3930	130
Avg	**3617.6**	**153.4**	**3531.5**	**153.3**	**3555.9**	**153.4**	**3727.3**	**153.6**	**3572.3**	**153.3**	**3584.9**	**154.1**

Table 15: Average input and output token counts per SQL-generation call across six query models and ten domains, using Llama-3.1-8B-Instruct as the schema generator. Output length is uniformly higher here than under the other two schema sources, tracking this generator's heavier table fragmentation (Table 3).

Domain	Fragmentation Metrics				Remarks
	C	R	RC	D	
Country	1.000	1.000	1.000	1.000	All countries participate fully across snapshot and leadership fragments.
Cricket Team	0.992	1.000	1.000	1.000	A few snapshots lack captain assignments; captaincy is modeled as an optional extension.
Cricketer	0.929	1.000	1.000	1.000	Some players do not appear in statistical snapshots due to partial format participation.
Cyclist	0.444	1.000	1.000	1.000	Optional annotation fragments (nicknames, medals) cover only a subset of riders.
Economy	1.000	1.000	1.000	1.000	All countries are fully represented across economic snapshots.
Equestrian	0.939	1.000	1.000	1.000	Not all riders earn medals; lower completeness reflects realistic outcomes.
Field Hockey	0.083	1.000	1.000	1.000	Managerial roles are rare career transitions and therefore sparsely populated.
Golfer	0.620	1.000	1.000	1.000	Awards are earned by only a subset of golfers, leading to reduced completeness.
Government Agency	0.125	1.000	1.000	1.000	Deputy-minister data is sparsely available in the original source timelines.
Table Tennis Player	0.960	0.960	1.000	0.040	Some snapshots contain no medal records; multiple medals per snapshot yield low disjointness.

Table 16: Fragment metrics for schema made using Gemini-2.5. Values below 1.0 arise from domain semantics such as optional annotation fragments, realistic sparsity in achievements or leadership roles, and intentional one-to-many relationships, rather than schema or reconstruction errors.

Domain	Fragmentation Metrics				Remarks
	C	R	RC	D	
Country	0.863	1.000	1.000	1.000	HDI and Gini indicators are unavailable for a subset of countries, reflecting missing socio-economic coverage rather than key loss.
Cricket Team	0.974	1.000	1.000	1.000	One team lacks an associated coach entry; coaching data is optional and incomplete for a small subset.
Cricketer	1.000	1.000	1.000	1.000	All cricketers, years, formats, and statistics are fully covered across fragments.
Cyclist	1.000	1.000	1.000	1.000	All rider, ride, nickname, and snapshot identifiers participate completely in fragments.
Economy	0.000	1.000	1.000	1.000	The `EconomyValues` fragment contains no temporal records as it is empty; however, other economic fragments jointly reconstruct all time snapshots.
Equestrian	0.702	0.000	1.000	1.000	Only a subset of athletes participate in medal templates, causing both incompleteness and failure of identifier reconstruction.
Field Hockey	1.000	1.000	1.000	1.000	Medal templates and athlete associations form a complete and lossless decomposition.
Golfer	0.000	1.000	1.000	1.000	Tournament fragment is empty, reducing completeness, but golfer identifiers remain recoverable via other fragments.
Government Agency	0.786	1.000	1.000	1.000	Not all agencies have minister assignments; chief mappings provide full identifier coverage.
Table Tennis Player	1.000	1.000	1.000	1.000	All player identifiers participate fully across medal and yearly snapshot fragments.

Table 17: Fragment metrics for schema made using llama-3.1-8B. Across domains, deviations from perfect completeness arise primarily from optional or sparsely populated fragments (e.g., awards, indicators, leadership roles) and empty indicator tables, while losslessness is generally preserved through fragment unions. All schemas maintain strict referential consistency and disjointness, indicating correct key preservation and absence of duplicate fact units despite domain-specific sparsity.

Domain	Fragmentation Metrics				Remarks
	C	R	RC	D	
Country	1.00	1.00	1.00	1.00	All snapshots contain GDP, HDI, and Gini fragments; joins fully reconstruct original records.
Cricket Team	1.00	1.00	1.00	1.00	Rank and record fragments fully cover all team snapshots without duplication or FK violations.
Cricketer	1.00	1.00	1.00	1.00	Snapshot and career statistics form a perfectly complete and lossless temporal decomposition.
Cyclist	0.11	1.00	1.00	1.00	Amateur-career data is sparse by design; however, all rider identifiers remain recoverable via fragment union.
Economy	1.00	1.00	1.00	1.00	Each economic snapshot is associated with indicator values, preserving temporal coverage and reconstruction.
Equestrian	1.00	1.00	1.00	1.00	Horse and rider result fragments jointly preserve all competition outcomes without duplication.
Field Hockey	0.76	1.00	1.00	1.00	Medal participation is partial across athletes, reducing completeness but preserving entity reconstruction.
Golfer	1.00	1.00	1.00	1.00	Timestamp and award fragments fully cover golfer entities and maintain lossless reconstruction.
Government Agency	0.79	1.00	1.00	0.36	Multiple officials linked to the same snapshot introduce intra-fragment duplication, reducing disjointness.
Table Tennis Player	0.95	0.62	1.00	0.68	Club and equipment data are missing for some snapshots; rank data introduces multiple rows per snapshot key.

Table 18: Fragmentation Metrics Evaluation (LLaMA 3.3 70B Instruct Schema). Most domains achieve perfect referential consistency and lossless reconstruction. Deviations from full completeness arise primarily from optional or sparsely populated fragments, while reduced disjointness reflects intentional one-to-many relationships such as officials, rankings, or equipment histories rather than schema errors.

E Error Analysis Examples

Example 1:
Question: In which years was the Panama's GDP nominal per capita the lowest?
SQL Query:

```
SELECT
  strftime('%Y', snapshot_id)
FROM
  Snapshots
WHERE
  country_id = (
    SELECT
      country_id
    FROM
      Countries
    WHERE
      country_name = 'Panama'
  )
  AND gdp_nominal_per_capita = (
    SELECT
      MIN(gdp_nominal_per_capita)
    FROM
      Snapshots
    WHERE
      country_id = (
        SELECT
          country_id
        FROM
          Countries
        WHERE
          country_name = 'Panama'
      )
  );
```

Answer: 2016, **Expected:** 2012

Example 2:
Question: Which person had the shortest tenure as the Monarch of the Barbados?(Assume that tenure for a position starts when the person appears in the timeline in that role and ends when either the timeline ends, or someone else replaces him.)
SQL Query:

```
SELECT l.leader_name
FROM Leaders l
JOIN SnapshotLeaders sl ON l.leader_id = sl.
    leader_id
JOIN LeaderRoles lr ON sl.role_id = lr.role_id
JOIN Snapshots s ON sl.snapshot_id = s.
    snapshot_id
WHERE lr.role_title = 'Monarch'
AND s.country_id = (SELECT country_id
       FROM Countries WHERE country_name = '
           Barbados')
ORDER BY (
       JULIANDAY(
           (
               SELECT
                   MIN(sl2.snapshot_id)
               FROM
                   SnapshotLeaders sl2
                   JOIN Leaders l2 ON sl2.
                       leader_id = l2.leader_id
                   JOIN LeaderRoles lr2 ON sl2.
                       role_id = lr2.role_id
                   JOIN Snapshots s2 ON sl2.
                       snapshot_id = s2.
                       snapshot_id
               WHERE
                   lr2.role_title = 'Monarch'
                   AND s2.country_id = (
                       SELECT
                           country_id
                       FROM
                           Countries
                       WHERE
                           country_name = '
                               Barbados'
                   )
                   AND l2.leader_name != l.
                       leader_name
                   AND sl2.snapshot_id > (
                       SELECT
                           MAX(sl3.snapshot_id)
                       FROM
                           SnapshotLeaders sl3
                           JOIN Leaders l3 ON
                               sl3.leader_id =
                               l3.leader_id
                           JOIN LeaderRoles lr3
                               ON sl3.role_id =
                               lr3.role_id
                           JOIN Snapshots s3 ON
                               sl3.snapshot_id =
                                s3.snapshot_id
                       WHERE
                           l3.leader_name = l.
                               leader_name
                           AND lr3.role_title =
```

```
                        'Monarch'
                    AND s3.country_id = (
                        SELECT
                            country_id
                        FROM
                            Countries
                        WHERE
                            country_name =
                                '
                                    Barbados'
                    )
                )
            )
        ) - JULIANDAY(MIN(sl.snapshot_id))
    ) ASC
LIMIT 1;
```

Answer: , **Expected:** Elizabeth II

Example 3:

Question: How many different event types did Wilhelm Melliger participate in?

SQL Query:

```
SELECT
    COUNT(DISTINCT discipline)
FROM
    Riders
WHERE
    full_name LIKE '%Wilhelm%'
    AND full_name LIKE '%Melliger%';
```

Answer: 1, **Expected:** 2

E.1 Examples of why 2.5 Pro isn't working better than 2.5 Flash

Example 1:

Question: Who was the coach when Balaji Pai was T20I captain?

Flash Schema Query (6 lines, 4 tables):

```
SELECT DISTINCT
    c.coach_name
FROM
    Coaches c
    JOIN TeamSnapshots ts ON c.coach_id = ts.
        coach_id
    JOIN TeamCaptaincy tc ON ts.snapshot_id =
        tc.snapshot_id
    JOIN Players p ON tc.player_id = p.
        player_id
    JOIN CaptainRoles cr ON tc.captain_role_id
        = cr.captain_role_id
WHERE
    cr.role_title = 'T20I Captain'
    AND ts.team_id = (
        SELECT
            team_id
        FROM
            CricketTeams
        WHERE
            team_name = 'Gibraltar national
                cricket team'
    )
    AND p.player_name = 'Balaji Pai';
```

Pro Schema Query (15 lines, 6 tables, self-join):

```
SELECT DISTINCT
    p.person_name
FROM
    People p
    JOIN TeamPersonnel tp ON p.person_id = tp.
        person_id
    JOIN Roles r ON tp.role_id = r.role_id
    JOIN TeamSnapshots ts ON tp.snapshot_id =
        ts.snapshot_id
    JOIN Teams t ON ts.team_id = t.team_id
    JOIN TeamPersonnel tp2 ON ts.snapshot_id =
        tp2.snapshot_id
    JOIN People p2 ON tp2.person_id = p2.
        person_id
    JOIN Roles r2 ON tp2.role_id = r2.role_id
WHERE
    r.role_name = 'Coach'
    AND r2.role_name = 'T20I Captain'
    AND p2.person_name = 'Balaji Pai'
    AND t.team_name = 'Gibraltar national
        cricket team';
```

Error Type: Schema Design

What Flash did right: Uses role-specific tables and direct relationships, allowing easier mappings of relationships, whereas the Pro schema requires a self-join on personnel data due to generic table structure (People, Roles).

Example 2:

Question: What is the win-loss ratio of the T20I for Gibraltar in 2019?

Flash Schema Query (4 lines, 1 table):

```
SELECT
    CAST(ts.t20i_wins_this_year AS REAL) /
        NULLIF(ts.t20i_losses_this_year, 0) AS
         ratio
FROM
    TeamSnapshots ts
WHERE
    ts.team_id = (
        SELECT
            team_id
        FROM
            CricketTeams
        WHERE
            team_name = 'Gibraltar national
                cricket team'
    )
    AND ts.snapshot_id LIKE '2019%';
```

Pro Schema Query (8 lines, 4 tables):

```
SELECT
    CAST(ps.wins AS REAL) / NULLIF(ps.losses,
        0) AS ratio
FROM
    Teams t
    JOIN TeamSnapshots ts ON t.team_id = ts.
        team_id
    JOIN PerformanceStats ps ON ts.snapshot_id
        = ps.snapshot_id
    JOIN Formats f ON ps.format_id = f.
        format_id
WHERE
```

```
    t.team_name = 'Gibraltar national cricket
        team'
    AND f.format_name = 'T20I'
    AND ts.snapshot_ts LIKE '2019%';
```

Error Type: Over-normalization
What Flash did right: Consolidated stats in the main table with minimal joins, whereas Pro split performance data into PerformanceStats and uses a separate Formats lookup, increasing unnecessary joins and increasing prompt complexity.

F Database Schema for all domains: Created by Gemini 2.5 Flash

F.1 Country

```
Countries (
    country_id INTEGER PRIMARY KEY,
    country_name TEXT UNIQUE NOT NULL
)
Leaders (
    leader_id INTEGER PRIMARY KEY ,
    leader_name TEXT UNIQUE NOT NULL
)
LeaderRoles (
    role_id INTEGER PRIMARY KEY ,
    role_title TEXT UNIQUE NOT NULL
)
Snapshots (
    snapshot_id TEXT PRIMARY KEY,
    country_id INTEGER NOT NULL,
    gdp_ppp INTEGER,
    gdp_ppp_rank INTEGER,
    gdp_ppp_per_capita REAL,
    gdp_ppp_per_capita_rank INTEGER,
    gdp_nominal INTEGER,
    gdp_nominal_rank INTEGER,
    gdp_nominal_per_capita REAL,
    gdp_nominal_per_capita_rank INTEGER,
    gini REAL,
    gini_rank INTEGER,
    hdi REAL,
    hdi_rank INTEGER,
    FOREIGN KEY (country_id)
        REFERENCES Countries(country_id)
)
SnapshotLeaders (
    snapshot_id TEXT NOT NULL,
    leader_id INTEGER NOT NULL,
    role_id INTEGER NOT NULL,
    PRIMARY KEY (snapshot_id,
    leader_id, role_id),
    FOREIGN KEY (snapshot_id)
        REFERENCES Snapshots(snapshot_id),
    FOREIGN KEY (leader_id)
        REFERENCES Leaders(leader_id),
    FOREIGN KEY (role_id)
        REFERENCES LeaderRoles(role_id)
)
```

F.2 Cricket team

```
CricketTeams (
        team_id INTEGER PRIMARY KEY ,
        team_name TEXT UNIQUE NOT NULL
    )
Coaches (
        coach_id INTEGER PRIMARY KEY,
        coach_name TEXT UNIQUE NOT NULL
    )
Players (
        player_id INTEGER PRIMARY KEY,
        player_name TEXT UNIQUE NOT NULL
    )
CaptainRoles (
        captain_role_id INTEGER PRIMARY KEY,
        role_title TEXT UNIQUE NOT NULL
    )
TeamSnapshots (
        snapshot_id TEXT PRIMARY KEY,
        team_id INTEGER NOT NULL,
        coach_id INTEGER,
        test_rank TEXT,
        odi_rank TEXT,
        t20i_rank TEXT,
        num_tests INTEGER,
        test_wins INTEGER,
        test_losses INTEGER,
        test_draws INTEGER,
        test_wins_this_year INTEGER,
        test_losses_this_year INTEGER,
        test_draws_this_year INTEGER,
        num_odis INTEGER,
        odi_wins INTEGER,
        odi_losses INTEGER,
        odi_ties INTEGER,
        odi_no_results INTEGER,
        odi_wins_this_year INTEGER,
        odi_losses_this_year INTEGER,
        odi_no_results_this_year INTEGER,
        wc_apps INTEGER,
        wcq_apps INTEGER,
        wcq_best TEXT,
        num_t20is INTEGER,
        t20i_wins INTEGER,
        t20i_losses INTEGER,
        t20i_ties INTEGER,
        t20i_no_results INTEGER,
        t20i_wins_this_year INTEGER,
        t20i_losses_this_year INTEGER,
        t20i_no_results_this_year INTEGER,
        wt20_apps INTEGER,
        wt20q_apps INTEGER,
        wt20q_best TEXT,
        FOREIGN KEY (team_id) REFERENCES
            CricketTeams(team_id),
        FOREIGN KEY (coach_id)
            REFERENCES Coaches(coach_id)
    )
TeamCaptaincy (
        snapshot_id TEXT NOT NULL,
        player_id INTEGER NOT NULL,
        captain_role_id INTEGER NOT NULL,
        PRIMARY KEY
        (snapshot_id, player_id,
            captain_role_id),
        FOREIGN KEY (snapshot_id)
        REFERENCES TeamSnapshots(snapshot_id),
        FOREIGN KEY (player_id)
        REFERENCES Players(player_id),
        FOREIGN KEY (captain_role_id)
        REFERENCES CaptainRoles(captain_role_id
            )
    )
```

F.3 cricketer

```
Players (
        player_id INTEGER PRIMARY KEY,
        player_name TEXT UNIQUE NOT NULL
    )
MatchFormats (
        format_id INTEGER PRIMARY KEY,
        format_name TEXT UNIQUE NOT NULL
    )
PlayerStatsSnapshots (
        snapshot_id TEXT NOT NULL,
        player_id INTEGER NOT NULL,
        format_id INTEGER NOT NULL,
        matches INTEGER,
        runs INTEGER,
        bat_avg REAL,
        hundreds INTEGER,
        fifties INTEGER,
        top_score INTEGER,
        deliveries INTEGER,
        wickets INTEGER,
        bowl_avg REAL,
        fivefor INTEGER,
        tenfor INTEGER,
        best_bowling_wickets INTEGER,
        best_bowling_runs INTEGER,
        catches INTEGER,
        stumpings INTEGER,
        PRIMARY KEY (snapshot_id, player_id,
            format_id),
        FOREIGN KEY (player_id)
            REFERENCES Players(player_id),
        FOREIGN KEY (format_id)
            REFERENCES MatchFormats(format_id)
    )
```

F.4 Economy

```
Countries (
        country_id INTEGER PRIMARY KEY,
        country_name TEXT UNIQUE NOT NULL
    )
EconomicSnapshots (
        snapshot_id TEXT NOT NULL,
        country_id INTEGER NOT NULL,
        gdp REAL,
        growth REAL,
        per_capita REAL,
        inflation REAL,
        poverty REAL,
        labor REAL,
        occupations REAL,
        unemployment REAL,
        exports REAL,
        imports REAL,
        debt REAL,
        revenue REAL,
        expenses REAL,
        aid REAL,
        edbr INTEGER,
        current_account REAL,
        balance REAL,
        reserves REAL,
        population INTEGER,
        PRIMARY KEY (snapshot_id, country_id),
        FOREIGN KEY (country_id)
            REFERENCES Countries(country_id)
    )
```

F.5 Equestrian

```
Athletes (
    athlete_id INTEGER PRIMARY KEY,
    full_name TEXT NOT NULL UNIQUE,
    birth_date TEXT,
    birth_place TEXT,
    nationality TEXT,
    discipline TEXT
)
Snapshots (
    snapshot_id INTEGER PRIMARY KEY,
    athlete_id INTEGER NOT NULL,
    snapshot_timestamp TEXT NOT NULL,
    height_ft INTEGER,
    height_in INTEGER,
    weight_lb INTEGER,
    FOREIGN KEY (athlete_id)
        REFERENCES Athletes(athlete_id),
    UNIQUE (athlete_id, snapshot_timestamp)
)
Sports (
    sport_id INTEGER PRIMARY KEY,
    sport_name TEXT NOT NULL UNIQUE
)
Competitions (
    competition_id INTEGER PRIMARY KEY,
    competition_name TEXT NOT NULL UNIQUE
)
Medal_Types (
    medal_type_id INTEGER PRIMARY KEY,
    medal_name TEXT NOT NULL UNIQUE
)
Medals (
    medal_id INTEGER PRIMARY KEY,
    athlete_id INTEGER NOT NULL,
    snapshot_id INTEGER NOT NULL,
    medal_type_id INTEGER NOT NULL,
    sport_id INTEGER,
    competition_id INTEGER,
    year INTEGER NOT NULL,
    city TEXT NOT NULL,
    event_name TEXT NOT NULL,
    format TEXT,
    FOREIGN KEY (athlete_id)
        REFERENCES Athletes(athlete_id),
    FOREIGN KEY (snapshot_id)
        REFERENCES Snapshots(snapshot_id),
    FOREIGN KEY (medal_type_id)
    REFERENCES Medal_Types(medal_type_id),
    FOREIGN KEY (sport_id)
        REFERENCES Sports(sport_id),
    FOREIGN KEY (competition_id)
        REFERENCES Competitions(competition_id)
)
```

F.6 Gov agencies

```
Agencies (
        agency_id INTEGER PRIMARY KEY,
        agency_name TEXT UNIQUE NOT NULL
    )
Persons (
        person_id INTEGER PRIMARY KEY,
        person_name TEXT UNIQUE NOT NULL
    )
Positions (
        position_id INTEGER PRIMARY KEY,
        position_title TEXT UNIQUE NOT NULL
```

```
    )
AgencySnapshots (
        snapshot_id TEXT PRIMARY KEY,
        agency_id INTEGER NOT NULL,
        employees INTEGER,
        budget REAL,
        FOREIGN KEY (agency_id)
            REFERENCES Agencies(agency_id)
    )
AgencyMinisters (
        snapshot_id TEXT NOT NULL,
        agency_id INTEGER NOT NULL,
        person_id INTEGER NOT NULL,
        position_id INTEGER NOT NULL,
        PRIMARY KEY (snapshot_id,
            agency_id, person_id, position_id),
        FOREIGN KEY (snapshot_id)
            REFERENCES AgencySnapshots(
                snapshot_id),
        FOREIGN KEY (agency_id)
            REFERENCES Agencies(agency_id),
        FOREIGN KEY (person_id)
            REFERENCES Persons(person_id),
        FOREIGN KEY (position_id)
            REFERENCES Positions(position_id)
    )
AgencyDeputyMinisters (
        snapshot_id TEXT NOT NULL,
        agency_id INTEGER NOT NULL,
        person_id INTEGER NOT NULL,
        position_id INTEGER NOT NULL,
        PRIMARY KEY (snapshot_id,
            agency_id, person_id, position_id),
        FOREIGN KEY (snapshot_id)
            REFERENCES AgencySnapshots(
                snapshot_id),
        FOREIGN KEY (agency_id)
            REFERENCES Agencies(agency_id),
        FOREIGN KEY (person_id)
            REFERENCES Persons(person_id),
        FOREIGN KEY (position_id)
            REFERENCES Positions(position_id)
    )
AgencyChiefs (
        snapshot_id TEXT NOT NULL,
        agency_id INTEGER NOT NULL,
        person_id INTEGER NOT NULL,
        position_id INTEGER NOT NULL,
        PRIMARY KEY (snapshot_id,
            agency_id, person_id, position_id),
        FOREIGN KEY (snapshot_id)
            REFERENCES AgencySnapshots(
                snapshot_id),
        FOREIGN KEY (agency_id)
            REFERENCES Agencies(agency_id),
        FOREIGN KEY (person_id)
            REFERENCES Persons(person_id),
        FOREIGN KEY (position_id)
            REFERENCES Positions(position_id)
    )
```

F.7 Table tennis player

```
Players (
    player_id INTEGER PRIMARY KEY,
    player_filename TEXT,
    name TEXT UNIQUE NOT NULL,
    nationality TEXT,
    birth_place TEXT,
    birth_date DATE,
    height REAL,
    weight REAL,
    native_name_lang TEXT
)

PlayerSnapshots (
    snapshot_id INTEGER PRIMARY KEY,
    player_id INTEGER NOT NULL,
    snapshot_timestamp DATETIME NOT NULL,
    hrank TEXT,
    crank TEXT,
    FOREIGN KEY (player_id)
        REFERENCES Players(player_id)
)
Competitions (
    competition_id INTEGER PRIMARY KEY,
    competition_name TEXT UNIQUE NOT NULL
)
Medals (
    medal_id INTEGER PRIMARY KEY ,
    snapshot_id INTEGER NOT NULL,
    competition_id INTEGER NOT NULL,
    medal_type TEXT NOT NULL,
    event_year INTEGER,
    event_location TEXT,
    event_format TEXT,
    partner TEXT,
    sport TEXT,
    country_representation TEXT,
    FOREIGN KEY (snapshot_id)
        REFERENCES PlayerSnapshots(snapshot_id)
            ,
    FOREIGN KEY (competition_id)
        REFERENCES Competitions(competition_id)
)
```

F.8 Golfer

```
Players (
    PlayerID INTEGER PRIMARY KEY,
    FullName TEXT(255) NOT NULL,
    BirthDate DATE,
    BirthPlace TEXT(255),
    Nationality TEXT(10),
    College TEXT(255),
    UNIQUE(FullName)
)
PlayerSnapshots (
    SnapshotID INTEGER PRIMARY KEY,
    PlayerID INTEGER,
    SnapshotTimestamp TIMESTAMP NOT NULL,
    Name TEXT,
    Height VARCHAR(50),
    Weight VARCHAR(50),
    Residence TEXT,
    YearPro INTEGER,
    RetiredYear INTEGER,
    ProWins INTEGER,
    PgaWins INTEGER,
    EuroWins INTEGER,
    AusWins INTEGER,
    ChampWins INTEGER,
    MajorWins INTEGER,
    OtherWins TEXT,
    Masters TEXT,
    USOpen TEXT,
    TheOpen TEXT,
    PGAChampionship TEXT,
```

```
    WghofID TEXT,
    WghofYear INTEGER,
    FOREIGN KEY (PlayerID)
        REFERENCES Players(PlayerID)
)
Tours (
    TourID INTEGER PRIMARY KEY,
    TourName TEXT UNIQUE NOT NULL
)
PlayerTours (
    SnapshotID INTEGER,
    TourID INTEGER,
    PRIMARY KEY (SnapshotID, TourID),
    FOREIGN KEY (SnapshotID)
        REFERENCES PlayerSnapshots(SnapshotID),
    FOREIGN KEY (TourID)
        REFERENCES Tours(TourID)
)
Awards (
    AwardID INTEGER PRIMARY KEY,
    AwardName TEXT UNIQUE NOT NULL
)
PlayerAwards (
    SnapshotID INTEGER,
    AwardID INTEGER,
    YearWon TEXT,
    PRIMARY KEY (SnapshotID, AwardID, YearWon),
    FOREIGN KEY (SnapshotID)
        REFERENCES PlayerSnapshots(SnapshotID),
    FOREIGN KEY (AwardID)
        REFERENCES Awards(AwardID)
)
```

F.9 Field hockey

```
Players (
    player_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    fullname TEXT,
    birth_name TEXT,
    nickname TEXT,
    birth_date TEXT,
    birth_place TEXT,
    death_date TEXT,
    death_place TEXT,
    honorific_prefix TEXT,
    honorific_suffix TEXT,
    UNIQUE (name, birth_date)
)
PlayerSnapshots (
    snapshot_id INTEGER PRIMARY KEY ,
    player_id INTEGER NOT NULL,
    snapshot_timestamp TEXT NOT NULL,
    height TEXT,
    weight TEXT,
    position TEXT,
    allegiance TEXT,
    branch TEXT,
    serviceyears TEXT,
    unit TEXT,
    `rank` TEXT,
    embed TEXT,
    death_place TEXT,
    FOREIGN KEY (player_id)
     REFERENCES Players(player_id)
)
Clubs (
    club_id INTEGER PRIMARY KEY ,
    club_name TEXT UNIQUE NOT NULL
)
PlayerClubs (
    player_club_id INTEGER PRIMARY KEY ,
    snapshot_id INTEGER NOT NULL,
    club_id INTEGER NOT NULL,
    years TEXT,
    FOREIGN KEY (snapshot_id)
        REFERENCES PlayerSnapshots(snapshot_id)
            ,
    FOREIGN KEY (club_id)
        REFERENCES Clubs(club_id)
)
NationalTeams (
    national_team_id INTEGER PRIMARY KEY,
    team_name TEXT UNIQUE NOT NULL
)
PlayerNationalTeams (
    player_national_team_id INTEGER PRIMARY KEY
        ,
    snapshot_id INTEGER NOT NULL,
    national_team_id INTEGER NOT NULL,
    years TEXT,
    caps INTEGER,
    goals INTEGER,
    FOREIGN KEY (snapshot_id)
        REFERENCES PlayerSnapshots(snapshot_id)
            ,
    FOREIGN KEY (national_team_id)
        REFERENCES
            NationalTeams(national_team_id)
)
Sports (
    sport_id INTEGER PRIMARY KEY,
    sport_name TEXT UNIQUE NOT NULL
)
Countries (
    country_id INTEGER PRIMARY KEY,
    country_code TEXT UNIQUE NOT NULL
)
Competitions (
    competition_id INTEGER PRIMARY KEY,
    competition_name TEXT UNIQUE NOT NULL
)
Medals (
    medal_id INTEGER PRIMARY KEY,
    medal_name TEXT UNIQUE NOT NULL
)
PlayerAchievements (
    achievement_id INTEGER PRIMARY KEY,
    player_id INTEGER NOT NULL,
    sport_id INTEGER,
    country_id INTEGER,
    competition_id INTEGER NOT NULL,
    medal_id INTEGER NOT NULL,
    year INTEGER,
    location TEXT,
    event_format TEXT,
    FOREIGN KEY (player_id)
        REFERENCES Players(player_id),
    FOREIGN KEY (sport_id)
        REFERENCES Sports(sport_id),
    FOREIGN KEY (country_id)
        REFERENCES Countries(country_id),
    FOREIGN KEY (competition_id)
        REFERENCES Competitions(competition_id)
            ,
    FOREIGN KEY (medal_id)
        REFERENCES Medals(medal_id)
)
```

F.10 Cyclist

```
Players (
    player_id INTEGER PRIMARY KEY ,
    ridername TEXT,
    fullname TEXT,
    nickname TEXT,
    dateofbirth TEXT,
    height REAL,
    weight REAL
)
Teams (
    team_id INTEGER PRIMARY KEY ,
    team_name TEXT UNIQUE
)
Countries (
    country_id INTEGER PRIMARY KEY ,
    country_name TEXT UNIQUE
)
RiderTypes (
    ridertype_id INTEGER PRIMARY KEY ,
    ridertype_name TEXT UNIQUE
)
Disciplines (
    discipline_id INTEGER PRIMARY KEY ,
    discipline_name TEXT UNIQUE
)
Roles (
    role_id INTEGER PRIMARY KEY ,
    role_name TEXT UNIQUE
)
MajorWins (
    win_id INTEGER PRIMARY KEY,
    win_description TEXT UNIQUE
)
PlayerMajorWins (
    player_id INTEGER,
    win_id INTEGER,
    PRIMARY KEY (player_id, win_id),
    FOREIGN KEY (player_id)
        REFERENCES Players(player_id),
    FOREIGN KEY (win_id)
        REFERENCES MajorWins(win_id)
)
Snapshots (
    snapshot_id INTEGER PRIMARY KEY,
    player_id INTEGER,
    snapshot_timestamp TEXT,
    team_id INTEGER,
    country_id INTEGER,
    ridertype_id INTEGER,
    discipline_id INTEGER,
    role_id INTEGER,
    FOREIGN KEY (player_id)
        REFERENCES Players(player_id),
    FOREIGN KEY (team_id)
        REFERENCES Teams(team_id),
    FOREIGN KEY (country_id)
        REFERENCES Countries(country_id),
    FOREIGN KEY (ridertype_id)
        REFERENCES RiderTypes(ridertype_id),
    FOREIGN KEY (discipline_id)
        REFERENCES Disciplines(discipline_id),
    FOREIGN KEY (role_id)
        REFERENCES Roles(role_id)
)
```

G Database Schema for all domains: Created by Llama-3.1-8B-Instruct

G.1 Country

```
Countries (
  CountryID INT PRIMARY KEY,
  CommonName VARCHAR(255) NOT NULL,
  ISOCode VARCHAR(3) NOT NULL
)

 Leaders (
  LeaderID INT PRIMARY KEY,
  Title VARCHAR(255) NOT NULL,
  Name VARCHAR(255) NOT NULL
)

CountryLeaders (
  CountryID INT NOT NULL,
  LeaderID INT NOT NULL,
  Timestamp DATE NOT NULL,
  PRIMARY KEY (CountryID,
  LeaderID, Timestamp),
  FOREIGN KEY (CountryID)
    REFERENCES Countries(CountryID),
  FOREIGN KEY (LeaderID)
    REFERENCES Leaders(LeaderID)
)

 Economy (
  EconomyID INT PRIMARY KEY,
  Timestamp DATE NOT NULL,
  CountryID INT NOT NULL,
  FOREIGN KEY (CountryID)
    REFERENCES Countries(CountryID)
)

 GDP (
  GDPID INT PRIMARY KEY,
  EconomyID INT NOT NULL,
  Year INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  Rank INT,
  FOREIGN KEY (EconomyID)
    REFERENCES Economy(EconomyID)
)

HDI (
  HDIID INT PRIMARY KEY,
  EconomyID INT NOT NULL,
  Year INT NOT NULL,
  Value DECIMAL(5, 3) NOT NULL,
  Rank INT,
  FOREIGN KEY (EconomyID)
    REFERENCES Economy(EconomyID)
)

Gini (
  GiniID INT PRIMARY KEY,
  EconomyID INT NOT NULL,
  Year INT NOT NULL,
  Value DECIMAL(5, 2) NOT NULL,
  Rank INT,
  FOREIGN KEY (EconomyID)
    REFERENCES Economy(EconomyID)
)
```

G.2 Cricket team

```
cricket_teams (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL
)

coaches (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL
)

captains (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL
)

team_captains (
  team_id INTEGER NOT NULL,
  captain_id INTEGER NOT NULL,
  PRIMARY KEY (team_id, captain_id),
  FOREIGN KEY (team_id)
    REFERENCES cricket_teams (id),
  FOREIGN KEY (captain_id)
    REFERENCES captains (id)
)

team_coaches (
  team_id INTEGER NOT NULL,
  coach_id INTEGER NOT NULL,
  PRIMARY KEY (team_id, coach_id),
  FOREIGN KEY (team_id)
    REFERENCES cricket_teams (id),
  FOREIGN KEY (coach_id)
    REFERENCES coaches (id)
)

rankings (
  id SERIAL PRIMARY KEY,
  team_id INTEGER NOT NULL,
  rank VARCHAR(255) NOT NULL,
  type VARCHAR(255) NOT NULL,
  snapshot DATE NOT NULL,
  FOREIGN KEY (team_id)
    REFERENCES cricket_teams (id)
)

records (
  id SERIAL PRIMARY KEY,
  team_id INTEGER NOT NULL,
  record VARCHAR(255) NOT NULL,
  type VARCHAR(255) NOT NULL,
  snapshot DATE NOT NULL,
  FOREIGN KEY (team_id)
    REFERENCES cricket_teams (id)
)

matches (
  id SERIAL PRIMARY KEY,
  team_id INTEGER NOT NULL,
  record VARCHAR(255) NOT NULL,
  type VARCHAR(255) NOT NULL,
  snapshot DATE NOT NULL,
  FOREIGN KEY (team_id)
    REFERENCES cricket_teams (id)
)
```

G.3 Cricketer

```
Cricketers (
  CricketerID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL
)

Years (
  YearID INT PRIMARY KEY,
  Year DATE NOT NULL
)

Cricketer_Year (
  CricketerID INT,
  YearID INT,
  PRIMARY KEY (CricketerID, YearID),
  FOREIGN KEY (CricketerID)
    REFERENCES Cricketers(CricketerID),
  FOREIGN KEY (YearID)
    REFERENCES Years(YearID)
)

CricketFormats (
  FormatID INT PRIMARY KEY,
  Format VARCHAR(50) NOT NULL
)

 Cricketer_Format (
  CricketerID INT,
  YearID INT,
  FormatID INT,
  PRIMARY KEY (CricketerID,
    YearID, FormatID),
  FOREIGN KEY (CricketerID)
    REFERENCES Cricketers(CricketerID),
  FOREIGN KEY (YearID)
    REFERENCES Years(YearID),
  FOREIGN KEY (FormatID)
    REFERENCES CricketFormats(FormatID)
)

CricketStats (
  StatID INT PRIMARY KEY,
  StatName VARCHAR(50) NOT NULL
)

Cricketer_Stat (
  CricketerID INT,
  YearID INT,
  FormatID INT,
  StatID INT,
  Value VARCHAR(255),
  PRIMARY KEY (CricketerID,
    YearID, FormatID, StatID),
  FOREIGN KEY (CricketerID)
  REFERENCES Cricketers(CricketerID),
  FOREIGN KEY (YearID)
    REFERENCES Years(YearID),
  FOREIGN KEY (FormatID)
    REFERENCES CricketFormats(FormatID),
  FOREIGN KEY (StatID)
    REFERENCES CricketStats(StatID)
)
```

G.4 Cyclist

```
Riders (
  RiderID INT PRIMARY KEY,
  RiderName VARCHAR(255) NOT NULL,
  FullName VARCHAR(255) NOT NULL,
  DateOfBirth DATE NOT NULL,
  Country VARCHAR(3) NOT NULL
)
```

```
Teams (
  TeamID INT PRIMARY KEY,
  TeamName VARCHAR(255) NOT NULL
)

Rides (
  RideID INT PRIMARY KEY,
  RiderID INT NOT NULL,
  TeamID INT NOT NULL,
  Discipline VARCHAR(10) NOT NULL,
  Role VARCHAR(10) NOT NULL,
  Ridertype VARCHAR(20) NOT NULL,
  ProYears VARCHAR(20) NOT NULL,
  ProTeams VARCHAR(255) NOT NULL,
  MajorWins VARCHAR(255) NOT NULL,
  FOREIGN KEY (RiderID)
    REFERENCES Riders(RiderID),
  FOREIGN KEY (TeamID)
    REFERENCES Teams(TeamID)
)

Snapshots (
  SnapshotID INT PRIMARY KEY,
  RideID INT NOT NULL,
  Timestamp DATETIME NOT NULL,
  Height VARCHAR(10) NOT NULL,
  Weight VARCHAR(10) NOT NULL,
  FOREIGN KEY (RideID)
    REFERENCES Rides(RideID)
)

Nicknames (
  NicknameID INT PRIMARY KEY,
  RiderID INT NOT NULL,
  Nickname VARCHAR(255) NOT NULL,
  FOREIGN KEY (RiderID)
  REFERENCES Riders(RiderID)
)
```

G.5 Economy

```
Countries (
  CountryID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL
)

TimeSnapshots (
  TimeSnapshotID INT PRIMARY KEY,
  Timestamp DATE NOT NULL
)

EconomyMetrics (
  EconomyMetricID INT PRIMARY KEY,
  MetricName VARCHAR(255) NOT NULL,
  MetricType VARCHAR(255) NOT NULL
)

EconomyValues (
  EconomyValueID INT PRIMARY KEY,
  EconomyMetricID INT NOT NULL,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (EconomyMetricID)
  REFERENCES EconomyMetrics(EconomyMetricID),
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
);

Exports (
  ExportID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Imports (
  ImportID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Debt (
  DebtID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Revenue (
  RevenueID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Expenses (
  ExpenseID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Aid (
  AidID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Reserves (
  ReserveID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Balance (
  BalanceID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Population (
  PopulationID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
```

```
)

 Labor (
  LaborID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Occupations (
  OccupationID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

Unemployment (
  UnemploymentID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

CurrentAccount (
  CurrentAccountID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)

HDI (
  HDIID INT PRIMARY KEY,
  TimeSnapshotID INT NOT NULL,
  Value DECIMAL(10, 2) NOT NULL,
  FOREIGN KEY (TimeSnapshotID)
  REFERENCES TimeSnapshots(TimeSnapshotID)
)
```

G.6 Equestrian

```
Athletes (
  AthleteID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  Fullname VARCHAR(255),
  Nationality VARCHAR(255) NOT NULL,
  Discipline VARCHAR(255) NOT NULL,
  BirthPlace VARCHAR(255),
  BirthDate DATE NOT NULL
)

MedalTemplates (
  MedalTemplateID INT PRIMARY KEY,
  Sport VARCHAR(255) NOT NULL,
  Country VARCHAR(255),
  Competition VARCHAR(255),
  MedalType VARCHAR(255) NOT NULL,
  Year INT NOT NULL
)

AthleteMedalTemplates (
  AthleteID INT NOT NULL,
  MedalTemplateID INT NOT NULL,
  PRIMARY KEY (AthleteID, MedalTemplateID),
  FOREIGN KEY (AthleteID)
  REFERENCES Athletes(AthleteID),
  FOREIGN KEY (MedalTemplateID)
  REFERENCES MedalTemplates(MedalTemplateID)
)

Timestamps (
  TimestampID INT PRIMARY KEY,
  Timestamp DATETIME NOT NULL UNIQUE
)
```

G.7 Field Hockey

```
yearly_snapshots (
  id INT PRIMARY KEY AUTO_INCREMENT,
  timestamp DATE
)

athletes (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255),
  full_name VARCHAR(255),
  nationality VARCHAR(255),
  birth_place VARCHAR(255),
  birth_date DATE,
  weight DECIMAL(5, 2),
  height DECIMAL(5, 2)
)

countries (
  id INT PRIMARY KEY AUTO_INCREMENT,
  code VARCHAR(255),
  name VARCHAR(255)
)

sports (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255)
)

medal_templates (
  id INT PRIMARY KEY AUTO_INCREMENT,
  country_code VARCHAR(255),
  medal_type VARCHAR(255),
  competition VARCHAR(255),
  year INT
)

athlete_medals (
  id INT PRIMARY KEY AUTO_INCREMENT,
  athlete_id INT,
  medal_template_id INT,
  FOREIGN KEY (athlete_id)
  REFERENCES athletes (id),
  FOREIGN KEY (medal_template_id)
  REFERENCES medal_templates (id)
)

national_teams (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255),
  country_code VARCHAR(255)
)

athlete_national_teams (
  id INT PRIMARY KEY AUTO_INCREMENT,
  athlete_id INT,
  national_team_id INT,
  years VARCHAR(255),
  caps INT,
  goals INT,
```

```
  FOREIGN KEY (athlete_id)
  REFERENCES athletes (id),
  FOREIGN KEY (national_team_id)
  REFERENCES national_teams (id)
)
```

G.8 golfer

```
Golfers (
  GolferID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  Fullname VARCHAR(255) NOT NULL,
  Birthdate DATE NOT NULL,
  Birthplace VARCHAR(255) NOT NULL,
  Height VARCHAR(255) NOT NULL,
  Nationality VARCHAR(255) NOT NULL,
  Residence VARCHAR(255) NOT NULL
)

Timestamps (
  TimestampID INT PRIMARY KEY,
  Timestamp DATETIME NOT NULL
)

GolferTimestamps (
  GolferID INT NOT NULL,
  TimestampID INT NOT NULL,
  PRIMARY KEY (GolferID, TimestampID),
  FOREIGN KEY (GolferID)
  REFERENCES Golfers(GolferID),
  FOREIGN KEY (TimestampID)
  REFERENCES Timestamps(TimestampID)
)

Awards (
  AwardID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  Description VARCHAR(255) NOT NULL
)

GolferAwards (
  GolferID INT NOT NULL,
  AwardID INT NOT NULL,
  TimestampID INT NOT NULL,
  PRIMARY KEY (GolferID, AwardID,
  TimestampID),
  FOREIGN KEY (GolferID)
  REFERENCES Golfers(GolferID),
  FOREIGN KEY (AwardID)
  REFERENCES Awards(AwardID),
  FOREIGN KEY (TimestampID)
  REFERENCES Timestamps(TimestampID)
);

 Tournaments (
  TournamentID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  Type VARCHAR(255) NOT NULL
)

GolferTournaments (
  GolferID INT NOT NULL,
  TournamentID INT NOT NULL,
  TimestampID INT NOT NULL,
  PRIMARY KEY (GolferID, TournamentID,
      TimestampID),
  FOREIGN KEY (GolferID)
  REFERENCES Golfers(GolferID),
  FOREIGN KEY (TournamentID)
  REFERENCES Tournaments(TournamentID),
  FOREIGN KEY (TimestampID)
  REFERENCES Timestamps(TimestampID)
)

TournamentResults (
  TournamentResultID INT PRIMARY KEY,
  GolferID INT NOT NULL,
  TournamentID INT NOT NULL,
  TimestampID INT NOT NULL,
  Result VARCHAR(255) NOT NULL,
  FOREIGN KEY (GolferID)
  REFERENCES Golfers(GolferID),
  FOREIGN KEY (TournamentID)
  REFERENCES Tournaments(TournamentID),
  FOREIGN KEY (TimestampID)
  REFERENCES Timestamps(TimestampID)
)

Children (
  ChildID INT PRIMARY KEY,
  GolferID INT NOT NULL,
  Name VARCHAR(255) NOT NULL,
  FOREIGN KEY (GolferID)
  REFERENCES Golfers(GolferID)
)

Spouses (
  SpouseID INT PRIMARY KEY,
  GolferID INT NOT NULL,
  Name VARCHAR(255) NOT NULL,
  FOREIGN KEY (GolferID)
  REFERENCES Golfers(GolferID)
)
```

G.9 Gov Agencies

```
Government_Agencies (
  id SERIAL PRIMARY KEY,
  agency_name VARCHAR(255) NOT NULL,
  timestamp DATE NOT NULL
)

Ministers (
  id SERIAL PRIMARY KEY,
  government_agency_id INTEGER NOT NULL,
  minister_number INTEGER NOT NULL,
  name VARCHAR(255) NOT NULL,
  position VARCHAR(255) NOT NULL,
  FOREIGN KEY (government_agency_id)
  REFERENCES Government_Agencies(id)
)

Chiefs (
  id SERIAL PRIMARY KEY,
  government_agency_id INTEGER NOT NULL,
  chief_number INTEGER NOT NULL,
  name VARCHAR(255) NOT NULL,
  position VARCHAR(255) NOT NULL,
  FOREIGN KEY (government_agency_id)
  REFERENCES Government_Agencies(id)
);


Government_Agency_Minister_Mappings (
  government_agency_id INTEGER NOT NULL,
  minister_id INTEGER NOT NULL,
  PRIMARY KEY (government_agency_id,
      minister_id),
```

```
  FOREIGN KEY (government_agency_id)
  REFERENCES Government_Agencies(id),
  FOREIGN KEY (minister_id)
  REFERENCES Ministers(id)
)

Government_Agency_Chief_Mappings (
  government_agency_id INTEGER NOT NULL,
  chief_id INTEGER NOT NULL,
  PRIMARY KEY (government_agency_id, chief_id),

  FOREIGN KEY (government_agency_id)
  REFERENCES Government_Agencies(id),
  FOREIGN KEY (chief_id)
  REFERENCES Chiefs(id)
)
```

G.10 Table Tennis Player

```
players (
  player_id INT PRIMARY KEY,
  name VARCHAR(255),
  fullname VARCHAR(255),
  nationality VARCHAR(255),
  birth_place VARCHAR(255),
  birth_date DATE,
  height VARCHAR(255),
  weight VARCHAR(255),
  residence VARCHAR(255)
)

medal_templates (
  medal_template_id INT PRIMARY KEY,
  sport VARCHAR(255),
  country VARCHAR(255),
  competition VARCHAR(255),
  medal_type VARCHAR(255),
  details VARCHAR(255)
)

player_medal_templates (
  player_id INT,
  medal_template_id INT,
  FOREIGN KEY (player_id)
  REFERENCES players (player_id),
  FOREIGN KEY (medal_template_id)
  REFERENCES medal_templates
  (medal_template_id)
)

yearly_snapshots (
  snapshot_id INT PRIMARY KEY,
  year INT,
  timestamp TIMESTAMP
)

player_yearly_snapshots (
  player_id INT,
  snapshot_id INT,
  FOREIGN KEY (player_id)
  REFERENCES players (player_id),
  FOREIGN KEY (snapshot_id)
  REFERENCES yearly_snapshots (snapshot_id)
)
```

H Database Schema for all domains: Created using Llama-3.3-70B-Instruct

H.1 Country

```
Countries (
    id INT AUTO_INCREMENT,
    common_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
)

Leaders (
       id INT AUTO_INCREMENT,
       name VARCHAR(255) NOT NULL,
       title VARCHAR(255),
       PRIMARY KEY (id)
)

Snapshots (
    id INT AUTO_INCREMENT,
    timestamp DATE NOT NULL,
    country_id INT,
    PRIMARY KEY (id),
    FOREIGN KEY (country_id)
    REFERENCES Countries(id)
)

Leader_Snapshots (
    snapshot_id INT,
    leader_id INT,
    leader_order INT,
    PRIMARY KEY (snapshot_id, leader_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Snapshots(id),
    FOREIGN KEY (leader_id)
    REFERENCES Leaders(id)
)

GDP (
    id INT AUTO_INCREMENT,
    snapshot_id INT,
    nominal DECIMAL(20,2),
    ppp DECIMAL(20,2),
    nominal_per_capita DECIMAL(20,2),
    ppp_per_capita DECIMAL(20,2),
    nominal_rank INT,
    ppp_rank INT,
    nominal_per_capita_rank INT,
    ppp_per_capita_rank INT,
    PRIMARY KEY (id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Snapshots(id)
)

HDI (
    id INT AUTO_INCREMENT,
    snapshot_id INT,
    rank INT,
    score DECIMAL(5,3),
    PRIMARY KEY (id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Snapshots(id)
)

Gini (
    id INT AUTO_INCREMENT,
    snapshot_id INT,
    coefficient DECIMAL(5,2),
    rank INT,
```

```
    PRIMARY KEY (id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Snapshots(id)
)
```

H.2 Cricket Team

```
Teams (
    team_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL
)

Coaches (
    coach_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL
)

Captains (
    captain_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL
)

Team_Snapshots (
    snapshot_id SERIAL PRIMARY KEY,
    team_id INTEGER NOT NULL,
    snapshot_date DATE NOT NULL,
    FOREIGN KEY (team_id)
    REFERENCES Teams(team_id)
)

Ranks (
    rank_id SERIAL PRIMARY KEY,
    team_id INTEGER NOT NULL,
    snapshot_id INTEGER NOT NULL,
    test_rank VARCHAR(10),
    odi_rank VARCHAR(10),
    t20i_rank VARCHAR(10),
    FOREIGN KEY (team_id)
    REFERENCES Teams(team_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Team_Snapshots(snapshot_id)
)

Records (
    record_id SERIAL PRIMARY KEY,
    team_id INTEGER NOT NULL,
    snapshot_id INTEGER NOT NULL,
    num_tests INTEGER,
    test_record VARCHAR(50),
    num_odis INTEGER,
    odi_record VARCHAR(50),
    num_t20is INTEGER,
    t20i_record VARCHAR(50),
    FOREIGN KEY (team_id)
    REFERENCES Teams(team_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Team_Snapshots(snapshot_id)
)

Coaching_History (
    coaching_history_id SERIAL PRIMARY KEY,
    team_id INTEGER NOT NULL,
    coach_id INTEGER NOT NULL,
    snapshot_id INTEGER NOT NULL,
    FOREIGN KEY (team_id)
    REFERENCES Teams(team_id),
    FOREIGN KEY (coach_id)
    REFERENCES Coaches(coach_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Team_Snapshots(snapshot_id)
)

Captaincy_History (
    captaincy_history_id SERIAL PRIMARY KEY,
    team_id INTEGER NOT NULL,
    captain_id INTEGER NOT NULL,
    snapshot_id INTEGER NOT NULL,
    format VARCHAR(10)
    CHECK(format IN ('test', 'odi', 't20i')),
    FOREIGN KEY (team_id)
    REFERENCES Teams(team_id),
    FOREIGN KEY (captain_id)
    REFERENCES Captains(captain_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Team_Snapshots(snapshot_id)
)
```

H.3 Cricketer

```
Player (
    player_id INT AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    PRIMARY KEY (player_id)
)

Career_Statistic (
    stat_id INT AUTO_INCREMENT,
    player_id INT NOT NULL,
    format VARCHAR(10) NOT NULL,
    matches INT,
    runs INT,
    bat_avg FLOAT,
    top_score INT,
    deliveries INT,
    wickets INT,
    bowl_avg FLOAT,
    best_bowling VARCHAR(20),
    catches INT,
    stumpings INT,
    PRIMARY KEY (stat_id),
    FOREIGN KEY (player_id)
    REFERENCES Player(player_id)
)

Snapshot (
    snapshot_id INT AUTO_INCREMENT,
    player_id INT NOT NULL,
    snapshot_date DATETIME NOT NULL,
    stat_id INT NOT NULL,
    PRIMARY KEY (snapshot_id),
    FOREIGN KEY (player_id)
    REFERENCES Player(player_id),
    FOREIGN KEY (stat_id)
    REFERENCES Career_Statistic(stat_id)
)
```

H.4 Cyclist

```
Riders (
  RiderID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  FullName VARCHAR(255) NOT NULL,
  Nickname VARCHAR(255),
  BirthDate DATE,
  BirthPlace VARCHAR(255),
  DeathDate DATE,
  DeathPlace VARCHAR(255),
  Height DECIMAL(3, 2),
```

```
  Weight INT,
  Discipline VARCHAR(50),
  Role VARCHAR(50),
  RiderType VARCHAR(50)
)

Teams (
  TeamID INT AUTO_INCREMENT PRIMARY KEY,
  TeamName VARCHAR(255) NOT NULL
)

 ProYears (
  ProYearID INT AUTO_INCREMENT PRIMARY KEY,
  RiderID INT NOT NULL,
  TeamID INT NOT NULL,
  StartYear INT NOT NULL,
  EndYear INT,
  FOREIGN KEY (RiderID)
  REFERENCES Riders(RiderID),
  FOREIGN KEY (TeamID)
  REFERENCES Teams(TeamID)
)

AmateurYears (
  AmateurYearID INT AUTO_INCREMENT PRIMARY KEY,

  RiderID INT NOT NULL,
  TeamID INT NOT NULL,
  StartYear INT NOT NULL,
  EndYear INT,
  FOREIGN KEY (RiderID)
  REFERENCES Riders(RiderID),
  FOREIGN KEY (TeamID)
  REFERENCES Teams(TeamID)
)

MajorWins (
  MajorWinID INT AUTO_INCREMENT PRIMARY KEY,
  RiderID INT NOT NULL,
  WinType VARCHAR(50) NOT NULL,
  WinName VARCHAR(255) NOT NULL,
  Year INT,
  FOREIGN KEY (RiderID)
  REFERENCES Riders(RiderID)
)

MedalTemplates (
  MedalTemplateID INT AUTO_INCREMENT PRIMARY
      KEY,
  RiderID INT NOT NULL,
  Country VARCHAR(50) NOT NULL,
  Sport VARCHAR(50) NOT NULL,
  Competition VARCHAR(50) NOT NULL,
  MedalType VARCHAR(50) NOT NULL,
  Year INT,
  FOREIGN KEY (RiderID)
  REFERENCES Riders(RiderID)
)

Snapshots (
  SnapshotID INT AUTO_INCREMENT PRIMARY KEY,
  RiderID INT NOT NULL,
  SnapshotDate DATETIME NOT NULL,
  CurrentTeam VARCHAR(255),
  FOREIGN KEY (RiderID)
  REFERENCES Riders(RiderID)
)
```

H.5 Economy

```
Countries (
    country_id INT AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    PRIMARY KEY (country_id)
)

Indicators (
    indicator_id INT AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    PRIMARY KEY (indicator_id)
)

EconomicSnapshots (
    snapshot_id INT AUTO_INCREMENT,
    country_id INT,
    year INT,
    PRIMARY KEY (snapshot_id),
    FOREIGN KEY (country_id)
    REFERENCES Countries(country_id)
)

EconomicData (
    data_id INT AUTO_INCREMENT,
    snapshot_id INT,
    indicator_id INT,
    value VARCHAR(255),
    PRIMARY KEY (data_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES EconomicSnapshots(snapshot_id),
    FOREIGN KEY (indicator_id)
    REFERENCES Indicators(indicator_id)
)
```

H.6 Equestrian

```
Athletes (
    id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    nationality VARCHAR(100) NOT NULL,
    birthdate DATE,
    discipline VARCHAR(100)
)

Medals (
    id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    competition VARCHAR(255) NOT NULL,
    year INT NOT NULL,
    medal_type VARCHAR(50) NOT NULL
)

Competitions (
    id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    year INT NOT NULL
)

Athlete_Medals (
    athlete_id INT NOT NULL,
    medal_id INT NOT NULL,
    FOREIGN KEY (athlete_id)
    REFERENCES Athletes(id),
    FOREIGN KEY (medal_id)
    REFERENCES Medals(id)
)
```

H.7 Field Hockey

```
Athletes (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    birth_place VARCHAR(255),
    nationality VARCHAR(255),
    position VARCHAR(255),
    height VARCHAR(20),
    weight VARCHAR(20),
    birth_date DATE
)

Athlete_Medals (
    id INT,
    medal VARCHAR(255),
    athlete_id INT,
    FOREIGN KEY (athlete_id)
    REFERENCES Athletes (id)
)

Teams (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    country VARCHAR(255),
    sport VARCHAR(255)
)

Team_Medals (
    id INT,
    medal VARCHAR(255),
    team_id INT,
    FOREIGN KEY (team_id)
    REFERENCES Teams (id)
)

Competitions (
    id INT PRIMARY KEY,
    name VARCHAR(255),
    year INT
)

Competitor_Medals (
    id INT,
    medal VARCHAR(255),
    competitor_id INT,
    competition_id INT,
    FOREIGN KEY (competitor_id)
    REFERENCES Athlete_Medals (id),
    FOREIGN KEY (competition_id)
    REFERENCES Competitions (id)
)

Yearly_Snapshots (
    id INT PRIMARY KEY,
    athlete_id INT,
    team_id INT,
    FOREIGN KEY (athlete_id)
    REFERENCES Athletes (id),
    FOREIGN KEY (team_id)
    REFERENCES Teams (id),
    year DATE
)

Yearly_Snapshot_Medals (
    id INT,
    medal VARCHAR(255),
    snapshot_id INT,
    FOREIGN KEY (snapshot_id)
    REFERENCES Yearly_Snapshots (id)
)
```

H.8 Golfer

```
Golfers (
    id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    fullname VARCHAR(255),
    nickname VARCHAR(255),
    birth_place VARCHAR(255),
    height VARCHAR(255),
    weight VARCHAR(255),
    nationality VARCHAR(255),
    residence VARCHAR(255),
    spouse VARCHAR(255),
    children VARCHAR(255),
    college VARCHAR(255),
    year_pro INT,
    tour VARCHAR(255),
    extour VARCHAR(255),
    retired INT,
    prowins INT,
    pgawins INT,
    eurowins INT,
    champwins INT,
    seneurowins INT,
    otherwins INT,
    majorwins INT,
    masters VARCHAR(255),
    usopen VARCHAR(255),
    open VARCHAR(255),
    pga VARCHAR(255),
    wghofid VARCHAR(255),
    wghofyear INT,
    birth_date DATE,
    death_date DATE,
    death_place VARCHAR(255)
)

Awards (
    id INT PRIMARY KEY,
    golfer_id INT,
    award_name VARCHAR(255),
    award_year INT,
    FOREIGN KEY (golfer_id)
    REFERENCES Golfers(id)
)

Timestamps (
    id INT PRIMARY KEY,
    golfer_id INT,
    timestamp VARCHAR(255),
    data TEXT,
    FOREIGN KEY (golfer_id)
    REFERENCES Golfers(id)
)
```

H.9 Gov Agencies

```
Agency (
    agency_id INT AUTO_INCREMENT,
    agency_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (agency_id)
)

Minister (
    minister_id INT AUTO_INCREMENT,
    minister_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (minister_id)
```

```
)

Chief (
    chief_id INT AUTO_INCREMENT,
    chief_name VARCHAR(255) NOT NULL,
    PRIMARY KEY (chief_id)
)

Snapshot (
    snapshot_id INT AUTO_INCREMENT,
    snapshot_date DATETIME NOT NULL,
    agency_id INT NOT NULL,
    budget DECIMAL(10, 2) DEFAULT 0.00,
    employees INT DEFAULT 0,
    PRIMARY KEY (snapshot_id),
    FOREIGN KEY (agency_id)
    REFERENCES Agency(agency_id)
)

Minister_Snapshot (
    minister_snapshot_id INT AUTO_INCREMENT,
    snapshot_id INT NOT NULL,
    minister_id INT NOT NULL,
    portfolio VARCHAR(255) NOT NULL,
    PRIMARY KEY (minister_snapshot_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Snapshot(snapshot_id),
    FOREIGN KEY (minister_id)
    REFERENCES Minister(minister_id)
)

Chief_Snapshot (
    chief_snapshot_id INT AUTO_INCREMENT,
    snapshot_id INT NOT NULL,
    chief_id INT NOT NULL,
    position VARCHAR(255) NOT NULL,
    PRIMARY KEY (chief_snapshot_id),
    FOREIGN KEY (snapshot_id)
    REFERENCES Snapshot(snapshot_id),
    FOREIGN KEY (chief_id)
    REFERENCES Chief(chief_id)
)
```

H.10 Table Tennis Player

```
Players (
    PlayerID INT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    FullName VARCHAR(255),
    Nationality VARCHAR(50),
    BirthPlace VARCHAR(100),
    Residence VARCHAR(100),
    Height VARCHAR(20),
    Weight VARCHAR(20),
    BirthDate DATE
)

Competitions (
    CompetitionID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Description VARCHAR(255)
)

MedalTemplates (
    MedalTemplateID INT PRIMARY KEY,
    MedalType VARCHAR(20) NOT NULL CHECK
    (MedalType IN ('Gold', 'Silver', 'Bronze'))
        ,
    Year INT,
    PlayerID INT,
    CompetitionID INT,
    FOREIGN KEY (PlayerID)
    REFERENCES Players(PlayerID),
    FOREIGN KEY (CompetitionID)
    REFERENCES Competitions(CompetitionID)
)

Snapshots (
    SnapshotID INT PRIMARY KEY,
    DateTime DATETIME NOT NULL,
    PlayerID INT,
    FOREIGN KEY (PlayerID)
    REFERENCES Players(PlayerID)
)

PlayerRanks (
    PlayerRankID INT PRIMARY KEY,
    Rank INT,
    RankType VARCHAR(20) NOT NULL CHECK
    (RankType IN ('World', 'Country')),
    SnapshotID INT,
    PlayerID INT,
    FOREIGN KEY (PlayerID)
    REFERENCES Players(PlayerID),
    FOREIGN KEY (SnapshotID)
    REFERENCES Snapshots(SnapshotID)
)

PlayerMedals (
    PlayerMedalID INT PRIMARY KEY,
    MedalTemplateID INT,
    SnapshotID INT,
    FOREIGN KEY (MedalTemplateID)
    REFERENCES MedalTemplates(MedalTemplateID),
    FOREIGN KEY (SnapshotID)
    REFERENCES Snapshots(SnapshotID)
)

Clubs (
    ClubID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL
)

PlayerClubs (
    PlayerClubID INT PRIMARY KEY,
    PlayerID INT,
    ClubID INT,
    SnapshotID INT,
    FOREIGN KEY (PlayerID)
    REFERENCES Players(PlayerID),
    FOREIGN KEY (ClubID)
    REFERENCES Clubs(ClubID),
    FOREIGN KEY (SnapshotID)
    REFERENCES Snapshots(SnapshotID)
)

Equipment (
    EquipmentID INT PRIMARY KEY,
    Type VARCHAR(50) NOT NULL,
    Description VARCHAR(255)
)

PlayerEquipment (
    PlayerEquipmentID INT PRIMARY KEY,
    PlayerID INT,
    EquipmentID INT,
    SnapshotID INT,
    FOREIGN KEY (PlayerID)
    REFERENCES Players(PlayerID),
```

```
    FOREIGN KEY (EquipmentID)
    REFERENCES Equipment(EquipmentID),
    FOREIGN KEY (SnapshotID)
    REFERENCES Snapshots(SnapshotID)
)

PlayingStyles (
    PlayingStyleID INT PRIMARY KEY,
    Type VARCHAR(50) NOT NULL,
    Description VARCHAR(255)
)

PlayerPlayingStyles (
    PlayerPlayingStyleID INT PRIMARY KEY,
    PlayerID INT,
    PlayingStyleID INT,
    SnapshotID INT,
    FOREIGN KEY (PlayerID)
    REFERENCES Players(PlayerID),
    FOREIGN KEY (PlayingStyleID)
    REFERENCES PlayingStyles(PlayingStyleID),
    FOREIGN KEY (SnapshotID)
    REFERENCES Snapshots(SnapshotID)
)
```

GRACE: Graph-Grounded Reflective Agent Copilot Engine for Expert-in-the-Loop Knowledge Expansion

John Seon Keun Yi[1], **Joshua R. Minot**[2], **Dokyun Lee**[1]
[1]Boston University [2]MassMutual
{jskyi, dokyun}@bu.edu, jminot26@massmutual.com

Abstract

Large language models deployed in high-stakes settings frequently generate plausible but ungrounded claims. Standard retrieval-augmented generation (RAG) pipelines offer limited remedy, since they retrieve isolated passages without tracking cross-document evidence relationships or quantifying uncertainty. We introduce GRACE (Graph-grounded Reflective Agent Copilot Engine), a framework that deconstructs LLM responses into atomic claims and grounds them against trusted knowledge priors within a weighted bipartite graph. Edge weights encode the closeness of each claim to the priors, enabling weighted centrality analysis that classifies claims as Grounded, Refuted, or Boundary. Such classification identifies not just hallucinations but also novel or contested claims at the frontier of the model's knowledge. To efficiently allocate human or agent resources, we formulate a Return on Attention (RoA) objective that defers a claim to expert review only when its priority-weighted uncertainty exceeds the cost of verification. Claims verified by experts are promoted to new evidence anchors, closing a validator-LLM evolutionary loop that expands the knowledge base across iterations. We evaluate GRACE across multiple language models and on datasets spanning both general and domain-specific knowledge. Our results show that our knowledge base serves as a reliable foundation for retrieval that outperforms RAG baselines, and that the RoA framework efficiently selects valuable boundary knowledge for expert verification. These findings demonstrate that graph-structured representations combined with expert-in-the-loop verification can mitigate hallucination at the system level rather than at the generation level. [1]

1 Introduction

Large language models (LLMs) have driven widespread adoption across industries, with recent analyses reporting that over 78% now regularly use generative AI in at least one business function (Singla et al., 2025). Despite this surge in adoption, scaled deployment remains low. More than 90% of function-specific use cases remain stuck in the pilot stage (McKinsey&Company, 2024). The primary bottleneck is reliability: inaccuracy is consistently cited as the top risk in deploying generative AI systems (Sakib et al., 2024; Sukharevsky et al., 2025), preventing their use in high-stakes decision-making where precision is non-negotiable. When prompted with out-of-knowledge queries, LLMs confabulate rather than abstain, producing fluent, confident-sounding outputs that are factually ungrounded. Any system that relies solely on LLM-internal consistency to judge truthfulness cannot distinguish between well-supported facts and shared hallucinations (Lin et al., 2022; Xu et al., 2024): false claims that are confidently and repeatedly generated across multiple samples.

Current mitigation strategies are insufficient to solve this problem. Retrieval-augmented generation (RAG) (Lewis et al., 2020) retrieves isolated document chunks to condition generation, but cannot link patterns across documents or track how evidence relates to specific claims. Uncertainty estimation methods (Jiang et al., 2024; Xia et al., 2025) offer complementary signals but still rely on the capacity of the base language model to judge its own reliability. None of these approaches provides the full auditability that high-stakes enterprise deployment demands. We argue that grounding LLM outputs against validated external *knowledge priors*, combined with structured expert verification, provides the necessary signal for safe and scalable deployment.

We introduce GRACE (Graph-grounded Reflec-

[1]Code available at https://github.com/johnsk95/grace_code

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 224–233
October 28, 2026 ©2026 Association for Computational Linguistics

tive Agent Copilot Engine), a framework that decomposes LLM responses into atomic claims and grounds them within a weighted bipartite graph alongside trusted knowledge priors. Edge weights encode the closeness of each claim to these priors, and weighted centrality analysis classifies claims into three categories: Grounded (high consensus and anchored to priors), Refuted (contradicting established knowledge), and Boundary (novel, contested, or sparsely connected). To convert this map into a growing knowledge base, GRACE formulates a Return on Attention (RoA) objective that routes only the most valuable boundary claims to experts for verification. Verified claims are promoted to new priors, closing a co-evolutionary loop that iteratively expands the knowledge base with each round of expert feedback. We evaluate GRACE across multiple language models and on datasets spanning both general and domain-specific knowledge. Our experiments demonstrate that: (1) the co-evolved knowledge base serves as a reliable foundation for downstream retrieval, outperforming RAG-based baselines; (2) the RoA framework efficiently selects high-value boundary claims for verification and knowledge expansion; and (3) displays robustness in scaled settings.

2 Related Work

Retrieval-Augmented Generation. RAG (Lewis et al., 2020) established the paradigm of conditioning LLM generation on retrieved passages, combining parametric and non-parametric memory for knowledge-intensive tasks. Self-RAG (Asai et al., 2024) extends this with adaptive retrieval and self-critique, while GraphRAG (Han et al., 2024; Edge et al., 2024) constructs knowledge graphs and community summaries to support query-focused summarization over broader contexts. Despite these advances, existing RAG systems retrieve isolated passages or subgraph summaries without tracking how individual claims relate to trusted evidence sources or quantifying claim-level uncertainty.

Uncertainty Quantification in LLMs. Methods for quantifying LLM reliability span token-level entropy (Ye et al., 2024), semantic entropy over meaning-clustered generations (Farquhar et al., 2024), and self-consistency checks across sampled responses (Manakul et al., 2023). Most directly related to our work, Jiang et al. (2024) model response-claim relationships as a bipartite graph and use centrality metrics to estimate claim-level uncertainty. However, all approaches rely on LLM-internal consistency and therefore cannot separate genuine facts from shared hallucinations. GRACE extends the graph-based framework of Jiang et al. (2024) by introducing external knowledge priors as anchors, shifting the basis of uncertainty from internal consistency to external grounding, and coupling it with a expert-in-the-loop expansion mechanism.

3 Method

Given user queries and a set of data sources, GRACE produces claims with auditable status labels (Grounded, Refuted, Boundary) and iteratively expands a trusted knowledge base through targeted expert verification. Figure 1 provides an overview of the full pipeline.

3.1 Preliminaries: Graph-Based Uncertainty Estimation

GRACE builds on the graph-based uncertainty framework of Jiang et al. (2024), which models the relationship between sampled responses and their constituent claims as a bipartite graph. Given a query, multiple responses $R = \{r_1, r_2, \ldots\}$ are sampled from an LLM, and each response is decomposed into atomic, sentence-level claims $C = \{c_1, c_2, \ldots\}$. A bipartite graph $G = (R \cup C, E)$ is constructed where an edge $(r_i, c_j) \in E$ exists if response r_i entails claim c_j. Graph centrality metrics (e.g., closeness centrality) then estimate each claim's uncertainty, since "true" claims tend to be more central in the graph, supported by many independent responses. However, since this approach relies entirely on LLM-internal consistency, it cannot distinguish genuine facts from shared hallucinations, i.e., false claims that receive high centrality simply because they are repeatedly generated.

3.2 Bipartite Graph Construction

To overcome this limitation, we introduce *knowledge priors*: trusted external information sources such as verified documents, compliance rulebooks, standard operating procedures, or expert-confirmed records. We integrate these priors directly into the graph structure to serve as anchors of truth, enabling the system to distinguish genuinely grounded claims from shared hallucinations.

Graph structure. We construct a weighted bipartite graph $G = ((R \cup P, C), E)$ where $R = \{r_1, r_2, \ldots\}$ is the set of sampled LLM responses,

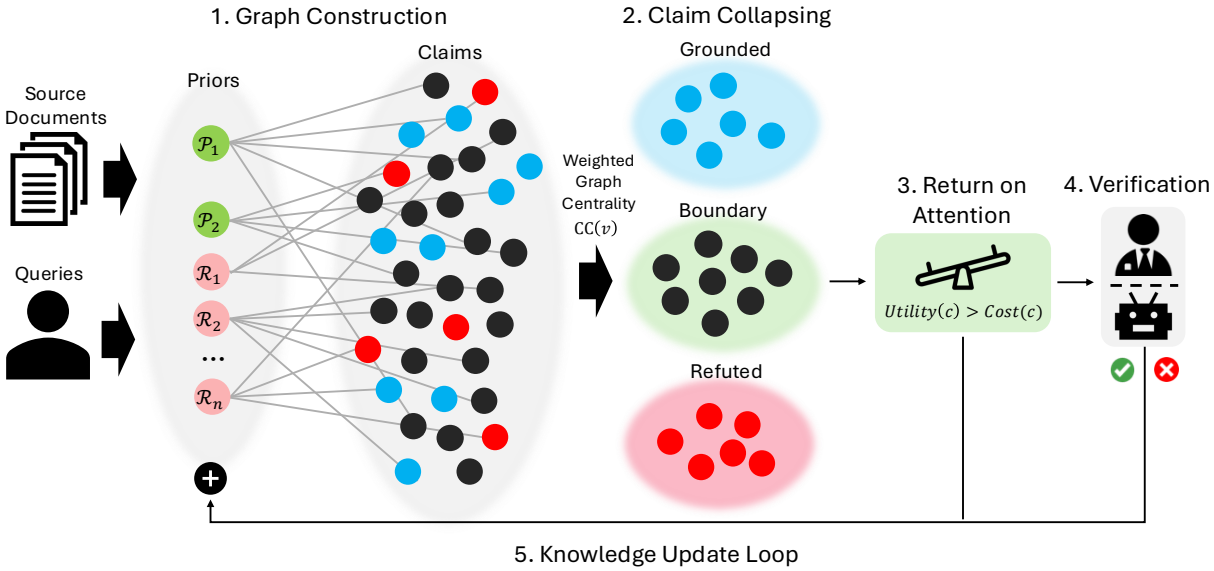


Figure 1: **Overview of the GRACE framework.** Given user queries and source documents, priors $\mathcal{P}$ and LLM responses $\mathcal{R}$ are decomposed into atomic claims and organized into a weighted bipartite graph. Closeness centrality $CC(v)$ classifies each claim as Grounded, Boundary, or Refuted. The Return on Attention (RoA) framework routes only boundary claims whose expected utility exceeds verification cost to experts (human or LLM agent). Confirmed claims are promoted to new priors, closing one knowledge expansion loop.

$P = \{p_0, p_1, \ldots\}$ is the set of trusted priors, and C is the set of unique atomic claims decomposed from both R and P. An edge $e = (u, v) \in E$ exists if a source node $u \in (R \cup P)$ entails a claim node $v \in C$. Let $C_P \subseteq C$ denote the subset of claims that are entailed by at least one prior.

Edge weighting. Unlike the unweighted baseline, we define a weighting function $W : E \to \mathbb{R}^+$ that encodes closeness to priors. Edges are assigned weights according to their relationship type:

$$w_{uv} = \begin{cases} w_{\text{anchor}} & \text{if } u \in P \text{ and } v \in C_P \\ w_{\text{agree}} & \text{if } u \in R \text{ and } v \in C_P \\ w_{\text{neutral}} & \text{if } u \in R \text{ and } v \notin C_P \\ w_{\text{contra}} & \text{if } v \text{ contradicts } P \end{cases} \quad (1)$$

where $w_{\text{anchor}} > w_{\text{agree}} > w_{\text{neutral}} > w_{\text{contra}}$. We define the traversal cost of an edge as $\text{cost}(u, v) = 1/w_{uv}$, so that paths through priors have the lowest cost. This weighting hierarchy ensures that centrality metrics naturally favor claims that are not merely consistent across LLM responses but actively grounded in verified knowledge.

3.3 Scoring and Knowledge Boundary Classification

With the weighted graph in place, GRACE computes centrality-based scores to classify each claim and map the LLM's knowledge boundary.

Grounded closeness centrality. We adopt closeness centrality as the primary scoring metric, following the findings of Jiang et al. (2024) that it is highly effective for detecting false claims. To handle potential graph disconnectedness, we use the modified closeness centrality variant adapted for our weighted structure:

$$CC(v) = \frac{N-1}{\sum_{u \in V} d_W(v, u)} \cdot \frac{|V_v|}{N} \quad (2)$$

where N is the total number of nodes, $|V_v|$ is the size of the connected component containing v, and $d_W(v, u)$ is the shortest-path distance between v and u computed using edge costs $1/w_{uv}$ via Dijkstra's algorithm. This metric assigns higher centrality to claims reachable via low-cost Anchor and Agreement links, effectively boosting grounded facts while pushing out ungrounded hallucinations.

Boundary classification. Using the closeness centrality scores, GRACE classifies each claim into one of three statuses: **Grounded**: Claims with high centrality scores that are strongly connected to priors through Anchor and Agreement links. These represent verified, trustworthy knowledge. **Refuted**: Claims with very low centrality scores, connected primarily through Contradictory links. These are identified hallucinations and are

archived or discarded. **Boundary**: Claims with intermediate centrality scores that are neither well-supported by priors nor clearly contradicted. These include *novel* claims (potentially true but unverified information outside existing priors) and *contested* claims (ambiguous or low-consensus). Boundary claims define the frontier of the LLM's knowledge and are candidates for verification.

Formally, we define the set of boundary claims as $C_{\text{boundary}} = C \setminus C_P$, or equivalently as claims below a centrality threshold δ: $C_{\text{boundary}} = \{c \in C \mid CC(c) < \delta\}$.

3.4 Return on Attention: Validator Deferral and Knowledge Expansion

Identifying the knowledge boundary is only the first step. GRACE closes the loop by routing the most valuable boundary claims to expert validators and promoting verified claims to new priors. We formalize this as the *Return on Attention* (RoA) framework.

Priority scoring. For each boundary claim $c \in C_{\text{boundary}}$, we compute a priority score as a weighted combination of two complementary signals:

$$\text{Priority}(c) = \lambda_c \cdot S_{\text{centrality}}(c) + \lambda_n \cdot S_{\text{novelty}}(c) \quad (3)$$

where λ_c, λ_n are tunable hyperparameters that balance the selection strategy.

The *centrality score* identifies "source" claims inside the unresolved subgraph whose verification would cascade to resolve clusters of related claims. Let $G_{\text{boundary}} = ((R, C_{\text{boundary}}), E')$ be the subgraph consisting only of LLM responses and boundary claims. We compute the betweenness centrality $C_B(c, G_{\text{boundary}})$ for each claim c in this subgraph:

$$S_{\text{centrality}}(c) = \frac{C_B(c, G_{\text{boundary}})}{\max_{c' \in C_{\text{boundary}}} C_B(c', G_{\text{boundary}})} \quad (4)$$

The *novelty score* identifies claims that are most distant from any established knowledge, representing frontiers for knowledge expansion. Using the weighted shortest-path distance, we compute:

$$\begin{aligned} d_{\min}(c) &= \min_{c^* \in C_P} d_W(c, c^*) \\ S_{\text{novelty}}(c) &= \frac{d_{\min}(c)}{\max_{c' \in C_{\text{boundary}}} d_{\min}(c')} \end{aligned} \quad (5)$$

Deferral objective. For each claim, the system chooses between two resolution paths: (1) *auto-resolve* (upper line in the equation below) by accepting the graph-based classification, or (2) *defer* (lower line in equation) to an expert at some cost. The utility of each claim is:

$$\text{Utility}(c) = \max \begin{cases} \sigma(CC(c)) \times \text{Priority}(c) \\ \text{Priority}(c) - \text{Cost}(c) \end{cases} \quad (6)$$

where $\sigma(CC(c))$ is the grounded closeness centrality normalized via a sigmoid function, representing the system's confidence in its own classification, and $\text{Cost}(c)$ is the cost of expert review. The system selects expert verification if and only if:

$$\text{Priority}(c) \cdot \big(1 - \sigma(CC(c))\big) > \text{Cost}(c) \quad (7)$$

This formalizes the RoA principle: validator effort is invested only when the claim is both important (high priority) and uncertain (low confidence), and the expected gain exceeds the cost of intervention. Given a limited verification budget B, the system selects a subset $\mathcal{D} \subseteq C_{\text{boundary}}$ of claims to defer such that $\sum_{c \in \mathcal{D}} \text{Cost}(c) \leq B$. A greedy selection strategy ranks candidates by their cost-adjusted marginal utility:

$$\text{Score}(c) = \frac{\text{Priority}(c) \cdot \big(1 - \sigma(CC(c))\big)}{\text{Cost}(c)} \quad (8)$$

and selects in decreasing order until the budget is exhausted. High-centrality nodes act as sources that can collapse clusters of related claims, while high-novelty nodes expand the knowledge frontier into previously unexplored regions.

Knowledge update loop. Expert validator review deferred claims and decide to either confirm or reject. Confirmed claims are promoted to new priors $P' = P \cup \{c_{\text{verified}}\}$, and edge weights throughout the graph are recomputed to reflect the expanded knowledge base. Rejected claims are archived as known hallucinations. Over successive iterations, this co-evolutionary loop accumulates the knowledge base: each round of expert feedback expands the set of priors, which in turn enables the graph to resolve more claims autonomously in subsequent iterations, progressively reducing the need for expert intervention.

4 Experiments

In this section, we detail a set of experiments to evaluate the performance of our GRACE frame-

work. Section 4.1 tests its retrieval ability, section 4.2 validates knowledge expansion through oracle feedback, and section 4.3 tests scalability to a large set of documents.

4.1 GRACE as a Retrieval Tool

We design this experiment to answer the research question: *Can GRACE serve as a reliable knowledge base for retrieval?* Unlike traditional retrieval methods that rely solely on surface-level similarity between a query and a document corpus, GRACE constructs a structured knowledge graph where every claim is scored by its closeness centrality to trusted priors. This experiment tests whether that graph structure can surface the most relevant and trustworthy passages when presented with a downstream query, effectively functioning as a grounded retrieval engine.

Datasets. We evaluate on two QA benchmarks containing long-form documents as context, replicating real-world scenarios where reference documents are lengthy and complex. QASPER (Dasigi et al., 2021) is a dataset of NLP research papers paired with free-form questions. Each paper is divided into named sections; we use the Title+Abstract, Introduction, and Methods sections as the three priors for graph construction. We randomly sample 100 papers from the test split. QuALITY (Pang et al., 2022) is a multiple-choice QA dataset over long documents (articles, stories, essays) averaging roughly 5,000 tokens in length. Since these documents lack explicit section boundaries, we segment each article into sequential chunks of approximately 1,000 words, each serving as a prior. We randomly select 50 articles from the validation set.

Models. We test with two language models as the QA backbone to measure adaptability across model scales: GPT-4o-mini (OpenAI, 2024), a large cloud-based model, and LLaMA 3.1 8B (Dubey et al., 2024), a small locally-hosted model. For QASPER, open-ended answers are judged for correctness by GPT-4.1; for QuALITY, multiple-choice accuracy is computed via exact match against the gold label.

Baselines. We compare GRACE against two zero-shot conditions and two retrieval-based methods. **Zero-shot + Part** provides only a partial document context to the LLM: the Title+Abstract section for QASPER, and the first chunk for QuALITY. This represents the minimal-context baseline. **Zero-shot + Full** passes the full document as context. This serves as an upper bound on available information. **RAG** (Lewis et al., 2020) uses a standard retrieval-augmented generation pipeline: passages are retrieved by embedding similarity and top-k results are provided as context. **GraphRAG** (Han et al., 2024; Edge et al., 2024) constructs a knowledge graph and community summaries from the document, then retrieves from these summaries to answer queries.

Metrics. We report two metrics: **Accuracy**, the percentage of questions answered correctly, and **Answerability**, the percentage of questions for which the model attempts an answer rather than abstaining. Answerability captures the model's calibration—an ideal system should abstain when its retrieved context is insufficient, rather than hallucinating an answer. We omit answerability for the QuALITY benchmark since all methods end up providing a "best answer" among the multiple choice options even with limited context.

GRACE Retrieval. For GRACE, we first construct the weighted bipartite graph for each document as described in Section 3. At query time, source passages are scored using a combined retrieval metric:

$$\text{score}(q, p) = \alpha \cdot \text{sim}(\mathbf{e}_q, \mathbf{e}_p) + (1-\alpha) \cdot CC(p) \quad (9)$$

where $\text{sim}(\mathbf{e}_q, \mathbf{e}_p)$ is the cosine similarity between the question and passage embeddings, $CC(p)$ is the normalized closeness centrality of the passage's associated claims, and $\alpha = 0.5$. Passages are deduplicated by source prior and the top-5 are provided as context for question answering. This scoring mechanism ensures that retrieved passages are not only semantically relevant to the query but also well-grounded in the knowledge graph.

Results. Table 1 reports accuracy and answerability across both benchmarks. On QASPER, GRACE matches the full-context zero-shot baseline on both models while retrieving only a small subset of the document, and substantially outperforms standard RAG. GraphRAG achieves comparable accuracy but reports answerability above 90% on both models, indicating substantial overconfidence. GRACE, by contrast, reports answerability closely aligned with its accuracy, demonstrating that closeness centrality provides a natural calibration mechanism. On QuALITY, GRACE performs

	QASPER [(Dasigi et al., 2021)]		QuALITY [(Pang et al., 2022)]
	Accuracy (%)	Answerable (%)	Accuracy (%)
GPT-4o-mini [(OpenAI, 2024)]			
Zero-shot + Part	10.8	26.6	56.2
Zero-shot + Full	29.0	53.6	79.6
RAG	19.8	45.2	66.5
GraphRAG	27.8	90.4	85.3
GRACE (Ours)	**28.1**	**50.6**	**77.0**
LLaMA 3.1 8B [(Dubey et al., 2024)]			
Zero-shot + Part	10.7	26.6	41.1
Zero-shot + Full	29.0	29.0	45.1
RAG	18.5	35.6	58.9
GraphRAG	29.0	95.2	29.1
GRACE (Ours)	**28.1**	**50.5**	**44.2**

Table 1: **GRACE serves as a reliable retrieval tool over other retrieval and zero-shot methods.** Numbers indicate average accuracy (%) and percentage of questions answerable (%) across two QA benchmarks. GRACE displays competitive QA performance, while not being overly confident in question answerability. Underlined numbers indicate the full context ceiling.

within a few points of the full-context baseline with GPT-4o-mini, while GraphRAG collapses to 29.1% on LLaMA despite leading on the larger model. Notably, GRACE's accuracy is robust to model scale: on QASPER it maintains nearly identical performance across GPT-4o-mini and LLaMA, whereas every other method degrades. This suggests that smaller models benefit most from a pre-structured, grounded retrieval source that reduces the reasoning burden.

4.2 Knowledge Expansion

This experiment is aimed at answering the question: *Can grounded and verified claims serve as a reliable context for question answering?* The retrieval experiment in Section 4.1 evaluates GRACE as a static knowledge base. However, the core premise of our framework is that the knowledge base should *grow*: expert-verified boundary claims become new priors that expand coverage and improve downstream performance over successive iterations. This experiment tests whether the co-evolutionary loop produces a knowledge base that serves as increasingly reliable context for question answering.

Setup. We evaluate the knowledge expansion loop on 100 QASPER papers (334 questions total), using pre-built bipartite graphs initialized with Abstract + Introduction priors (Round 0). In each of three expansion rounds (Round 1-3), a responder LLM (GPT-4o-mini) generates answers using a progressively revealed batch of held-out paper sections as out-of-distribution (OOD) context. The responses are decomposed into claims and verified by an LLM verifier (GPT-4.1) that has the full paper as context. We use the LLM verifier as a proxy for human experts, allowing fully automated evaluation of the expansion loop. Because this LLM verifier setting underpins every result in this section, we also test with human verifiers to compare alignment. A separate study replaces the verifier with 15 graduate students familiar with NLP topics on a 15 paper subset, reported at the end of this section (Table 4). We set $Cost(c)$ as the minimum embedding distance between claim c and the nearest prior, reflecting the assumption that claims most distant from established knowledge are hardest to validate. For human validation, we also add time for validation per claim to the cost function. Verified claims are promoted to Grounded status with anchor-weight edges ($w = 3.0$), and a cascade re-check propagates grounding to additional Boundary claims that are entailed by the newly verified claim. For downstream QA evaluation, the accumulated Grounded claims at each round serve as the sole context provided to the responder LLM. We compare against an abstract-only baseline and a full-paper upper bound, where the entire paper text is given as context.

Metrics. In addition to QA accuracy, we track four metrics across expansion rounds. **Prior Coverage Rate (PCR)** is the fraction of all response claims classified as Grounded, measuring overall knowledge coverage. **Knowledge Frontier (KF)** is the average number of Boundary claims remaining per paper, indicating how much unresolved knowl-

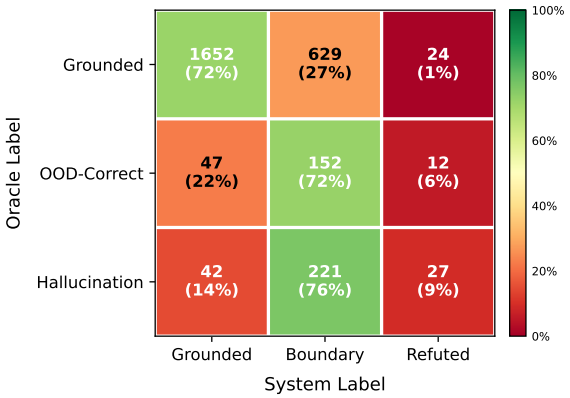


Figure 2: **GRACE claim classification align with ground truth.** Confusion matrix of system-assigned labels versus oracle labels.

edge persists. **Verification Yield** is the fraction of oracle-routed OOD claims confirmed as correct, measuring the efficiency of the claim selection algorithm. **Cascade** is the number of Boundary claims that flipped to Grounded *without* being directly verified, capturing the indirect grounding propagation triggered by each verification round.

Context	**Accuracy (%)**	**Δ**
Abstract only	0.1413	—
Round 0 (Abstract + Intro)	0.2225	+8.1
Round 1 (Batch 1)	0.2447	+10.3
Round 2 (Batch 2)	0.3716	+23.0
Round 3 (Batch 3)	0.4913	+35.0
Full paper	0.3360	+19.5

Table 2: **Downstream QA accuracy improves monotonically with each expansion round.** Δ is relative to the abstract-only baseline.

Results. Table 2 reports downstream QA accuracy across expansion rounds. Starting from 14.1% with abstract-only context, accuracy rises monotonically to 49.1% after three rounds of expansion. Notably, Round 3 accuracy surpasses the full-paper oracle (33.6%) by 15.5 percentage points, demonstrating that a compact pool of verified claims provides more effective QA context than raw full-text, which contains noise from unrelated sections. Table 3 shows the underlying knowledge graph dynamics: PCR grows from 63.1% to 83.4%, while KF shrinks from 10.0 to 4.6 Boundary claims per paper. Verification yield increases across rounds (56.3% → 69.5%), indicating that later batches (covering results and conclusions) produce claims that are more readily verifiable. Cascade propagation is frontloaded (3.77 claims/paper in Round 1, decaying to 0.73 by Round 3), reflecting the dense mutual entailment structure of methods sections. Additionally, the confusion matrix in Figure 2 confirms the classifier's safety properties: grounding precision reaches 97.6%, meaning contamination of the Grounded pool by hallucinations is rare (42 out of 1,741 system-Grounded claims, or 2.4%). The system's primary failure mode is conservative—22.3% of OOD-correct claims are prematurely promoted to Grounded because they closely paraphrase the abstract—while 85.5% of hallucinated claims are correctly classified as Boundary or Refuted. These results demonstrate that GRACE's co-evolutionary loop produces a knowledge base that is both safer and more effective than raw document context, with each verification round compounding coverage gains through cascade propagation.

Human Verification. The verifier in the experiment above is an LLM agent. To our method's validity with human experts, we re-ran one expansion round on a 15-paper subset with the LLM verifier replaced by 15 NLP graduate students, who produced 354 annotations covering 43 Boundary claims at five or more raters overlapped per claim. Table 4 shows that the two verifiers are interchangeable on coverage and downstream accuracy but not on how much of the frontier they resolve. PCR is almost equivalent (93.63% vs 93.58%), and QA accuracy is identical on all 48 questions. For the knowledge frontier (KF), human verification leaves 2.07 Boundary claims per paper against the LLM's 2.93, a gap of 0.87 claims. This is due to the high rate of abstention from the LLM verifier: given the same 29 claims, the judge returns 10 confirmations, 6 rejections and 13 abstentions, while the human validators return 11, 17 and 1. An abstention leaves a claim on the frontier whereas a rejection removes it, so the judge's 44.83% abstention rate against the validators' 3.45% accounts for the KF gap. We interpret this as a difference in decisiveness rather than in accuracy: the study has no gold key against which either verifier can be scored, so Yield is an acceptance rate and not a correctness measure. Overall, we find that human validators bring similar accuracy benefits to the LLM ones, but expand the knowledge base even more.

Round	Context	PCR↑	KF↓	Yield	Cascade
0	Abstract + Intro	63.1%	10.02	—	—
1	+ Batch 1	77.3%	6.25	56.3%	3.77
2	+ Batch 2	80.6%	5.30	58.4%	0.95
3	+ Batch 3	83.4%	4.57	69.5%	0.73

Table 3: **Knowledge coverage grows and the boundary shrinks with each expansion round.** PCR rises while verification yield increases across rounds.

Round	Accuracy	PCR↑	KF↓	Yield	Abstention
Round 0	16.67%	89.55%	5.33	—	—
Round 1 (LLM validator)	18.75%	93.58%	2.93	34.48%	44.83%
Round 1 (Human)	18.75%	93.63%	2.07	37.93%	3.45%

Table 4: **Human validators match the LLM judge on knowledge coverage but shrink the knowledge frontier substantially further.** One expansion round on a 15-paper QASPER subset, verified by 15 human experts compared with LLM validators.

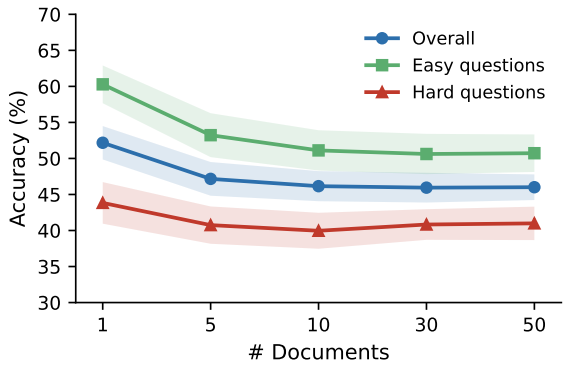


Figure 3: **GRACE displays reliable retrieval performance on large knowledge bases with multiple long documents.** QA accuracy drops initially at $1 \rightarrow 5$ documents, but plateaus when more documents are added.

4.3 Scalability

This experiment extends Section 4.1 to measure whether GRACE can reliably retrieve relevant context as the knowledge base grows. RAG systems are known to degrade as the number of indexed documents increases, since retrievers must distinguish relevant passages from a larger pool of semantically similar but irrelevant candidates (Shi et al., 2023; Cuconasu et al., 2024). This is a realistic challenge in enterprise settings where knowledge bases span hundreds of documents (Edge et al., 2024).

We reuse the 50 QuALITY documents and questions from Section 4.1. Instead of retrieving from a single reference document, we add multiple documents to the knowledge graph (5–50), increasing the claim pool from an average of 91 claims per single-article graph to 4,554 claims at 50 documents. For all settings, $k = 20$ highest-scoring claims are retrieved and passed as numbered context to the responder (GPT-4o-mini), which selects among four multiple-choice options.

Results. As shown in Figure 3, GRACE maintains largely stable performance as the knowledge base scales from 1 to 50 documents. The largest accuracy drop occurs at the $1 \rightarrow 5$ document transition ($52.18\% \rightarrow 47.16\%$, -5.02pp), with minimal further degradation beyond that point. Easy questions degrade more than hard ones, likely because precise factual recall is more sensitive to retrieval noise than multi-step reasoning. Overall, despite the claim pool growing $50\times$, accuracy falls by less than 6 percentage points and stabilizes well before the largest pool sizes are reached.

5 Conclusion

We introduce GRACE, a graph-grounded framework that addresses LLM hallucination at the systems level by structuring claims and trusted knowledge priors into a weighted bipartite graph. Through weighted closeness centrality analysis, GRACE maps the LLM's knowledge frontier and enables separation of verified facts from hallucinations and novel knowledge. The Return on Attention framework allocates scarce expert verification effort by only routing claims whose utility exceeds the cost of review. Our experiments demonstrate that GRACE outperforms RAG-based baselines as a retrieval tool, that the ROA-guided claim se-

lection efficiently identifies high-value boundary knowledge for expansion, and that the framework scales reliably as the underlying corpus grows. In future work, we plan to integrate real-world data from various use cases in the industry to further validate GRACE's utility in expert-in-the-loop knowledge expansion.

Limitations

Three limitations bound this work. First, graph construction is the dominant cost. Claim decomposition and pairwise prior-claim entailment sweep rely on LLM calls, and it is reducible by replacing LLM decomposition with rule-based extraction for structured priors and by pre-filtering the entailment sweep so that only ambiguous pairs reach the LLM. Second, our human verification study is small: one expansion round with 15 participants. Time and resources allowing, we plan to recruit more participants to validate claims from more sources, and expand to more rounds to measure GRACE's function on multiple human-in-the-loop expansion iterations. Lastly, GRACE assumes every prior is equally trustworthy, giving all prior-claim edges the same weight w_{anchor}. However, real sources go out of date, are superseded, or are uncertain in themselves. In future developments, we plan to introduce a per-prior trust term derived from provenance, recency, and expert reliability, propagated through $CC(v)$ so that grounding reflects confidence in the priors and not only proximity to them.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pages 9112–9141.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*.

Mingjian Jiang, Yangjun Ruan, Prasanna Sattigeri, Salim Roukos, and Tatsunori Hashimoto. 2024. Graph-based uncertainty metrics for long-form language model generations. *Advances in Neural Information Processing Systems*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 3214–3252.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017.

McKinsey&Company. 2024. Mckinsey's ecosystem of strategic alliances brings the power of generative ai to clients. https://www.mckinsey.com/about-us/new-at-mckinsey-blog/mckinsey-alliances-bring-the-power-of-generative-ai-to-clients.

OpenAI. 2024. Gpt-4o mini: advancing cost-efficient intelligence.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. 2022. Quality: Question answering with long

input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358.

Md Nazmus Sakib, Md Athikul Islam, Royal Pathak, and Md Mashrur Arifin. 2024. Risks, causes, and mitigations of widespread deployments of large language models (llms): A survey. In *2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)*, pages 1–7. IEEE.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Alex Singla, Alexander Sukharevsky, Lareina Yee, Michael Chui, and Bryce Hall. 2025. The state of ai: How organizations are rewiring to capture value. *McKinsey & Company*.

Alexander Sukharevsky, Dave Kerr, Klemens Hjartar, Lari Hämäläinen, Stéphane Bout, Vito Di Leo, and Guillaume Dagorret. 2025. Seizing the agentic AI advantage. Technical report, McKinsey & Company.

Zhiqiu Xia, Jinxuan Xu, Yuqian Zhang, and Hang Liu. 2025. A survey of uncertainty estimation methods on large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21381–21396.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek F Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. 2024. Benchmarking llms via uncertainty quantification. *Advances in Neural Information Processing Systems*, 37:15356–15385.

Acknowledgments

This project was supported by a grant from MassMutual.

Can We Do Interpretable NLI with Graphs Based on Atomic Propositions?

Younes Boufouss[1]*, **Luc Pommeret**[1,2]*
Thomas Gerald[1], **Patrick Paroubek**[1], **Sophie Rosset**[1]
[1]Université Paris-Saclay, CNRS, LISN, Orsay, France
[2]SCIAM
Correspondence: younes.boufouss@universite-paris-saclay.fr

Abstract

While Large Language Model (LLM)-based Natural Language Inference (NLI) systems achieve high accuracy, their decision-making processes lack auditable structures. This paper explores whether NLI can be performed using only interpretable, graph-based representations of evidence. We introduce a fully graph-based pipeline where the classifier never directly processes the input text. Instead, sentences are decomposed into atomic propositions, converted into `ConceptNet` triples via constrained decoding, and represented as three graphs per pair: premise, hypothesis, and a retrieved `ConceptNet` subgraph. These graphs are then fed into a fine-tuned 0.8-billion-parameter language model. On the SNLI dataset, our pipeline achieves 89.7% accuracy, just 1.9 points below an identically trained text-based model. On ANLI, it matches the published performance of `RoBERTa-large` on rounds R2 and R3 (48.0% vs. 48.9% and 44.9% vs. 44.4%) but trails by 16 points on R1, resulting in an overall gap of 9 to 14 points compared to its text counterpart. We term this gap the *price of interpretability* and demonstrate that it stems from representational limitations rather than data constraints. Ablation studies further reveal that graphs and text are complementary: combining both modalities achieves 92.1% accuracy on SNLI.

1 Introduction

Natural Language Inference (NLI) is a fundamental task in Natural Language Understanding (NLU). Accurate predictions in NLI require multi-level reasoning, encompassing lexical (e.g., synonyms, antonyms), syntactic (e.g., negation, quantifiers), and semantic dimensions (e.g., logical entailment, common-sense knowledge). While current systems achieve state-of-the-art performance on benchmarks like SNLI (Bowman et al., 2015) and MultiNLI (Williams et al., 2018), these datasets are not without flaws. For instance, annotation biases, negation, and vagueness are strongly correlated with specific classes, enabling even shallow models to perform surprisingly well. Gururangan et al. (2018) demonstrated that a simple text classifier trained solely on the hypothesis achieves 67% accuracy on SNLI and 53% on MultiNLI. This suggests that a significant portion of the high performance in these benchmarks may stem from exploiting linguistic artifacts rather than genuine reasoning capabilities.

To address these artifacts, ANLI (Nie et al., 2020) was introduced as a more robust benchmark. Constructed through an iterative human-in-the-loop protocol, ANLI consists of three rounds (R1, R2, and R3) of increasing difficulty. In each round, annotators are tasked with crafting hypotheses that state-of-the-art models would misclassify, given a premise. These examples are then verified by other annotators to ensure their validity. After each round, the adversarial examples are incorporated into the training data, and the model is retrained.

Given this design, ANLI provides a natural testbed to evaluate the effectiveness of our knowledge graph-based approach. By assessing performance on ANLI, we can determine whether our system leverages structured knowledge to achieve genuine reasoning or if it, too, falls back on superficial patterns.

In this context, two successive measures of the same metric — Accuracy — are typically used. The first measure is obtained by providing raw text to a large pretrained model, which contains no explicit reasoning cues. Interpretability, however, is achieved by constraining the decision-making process to rely on interpretable intermediate structures (*e.g.*, knowledge graphs).

In this work, we quantify the *price of interpretability*, *i.e.*, the performance gap between these two approaches.

Our central research question is: *Do graphs suf-*

*Equal contribution.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 234–246
October 28, 2026 ©2026 Association for Computational Linguistics

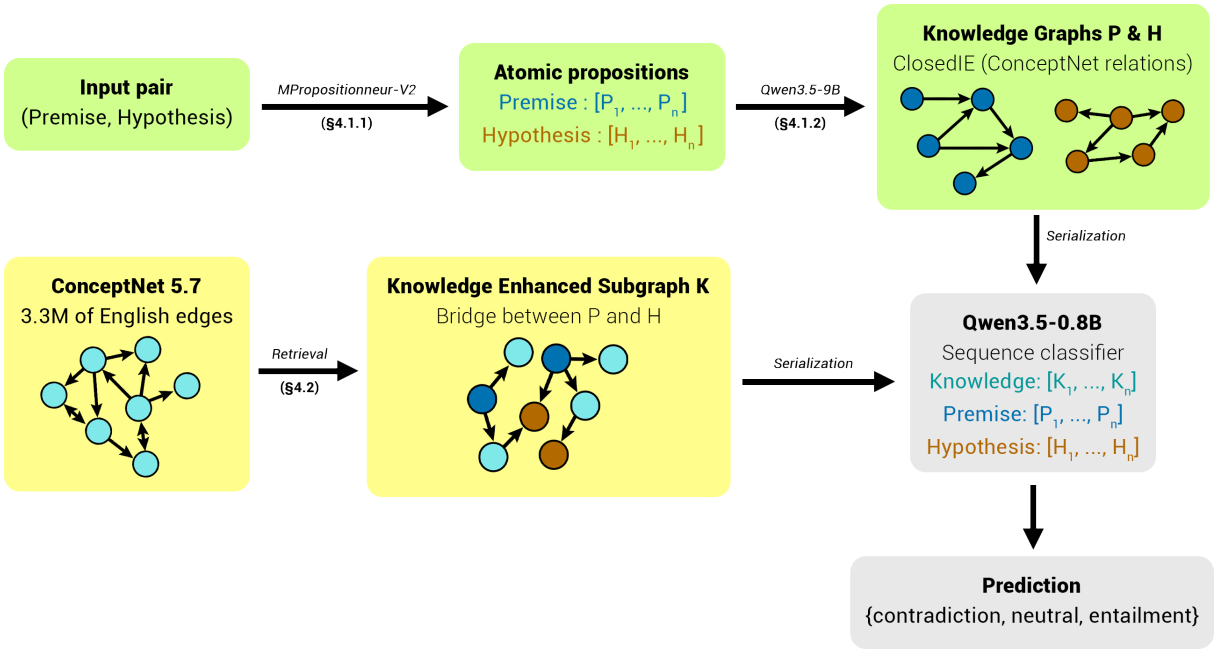


Figure 1: The four stages of the pipeline. Both sentences of the pair are decomposed into atomic propositions, each proposition is converted into `ConceptNet` triples under a constrained JSON schema to form the premise graph P and the hypothesis graph H, a third graph K is retrieved from `ConceptNet`, and the three graphs are serialised in the order K, P, H for the classifier. No natural language text reaches the classification stage.

fice? In other words, can structured representations alone achieve competitive performance, or is there an inherent trade-off between interpretability and accuracy? Our goal is to measure and analyse this trade-off systematically.

Our contribution. Here, we provide a pipeline to perform the NLI task entirely on knowledge graphs, in the syntax of `ConceptNet` (Speer et al., 2017). The graphs are extracted from atomic propositions as in Pommeret et al. (2026), augmented with a retrieved `ConceptNet` subgraph, serialised, and classified by a fine-tuned `Qwen3.5-0.8B-Base`. The design guarantees that no raw text reaches the classifier, so every piece of information used for the decision is present in an interpretable list of triplets.[1] Our contributions are as follows:

1. A full graph-based NLI pipeline, featuring a 28-relation extraction vocabulary derived from `ConceptNet`[2], and a constrained JSON schema ensuring structural consistency (see Section 4.1.2).

2. Quantification of the *price of interpretability*: we measure the accuracy drop between text-based and graph-based models *ceteris paribus*: -1.9 points on SNLI and -9 to -14 points on ANLI (see Table 3).

3. Representation, not data, as the bottleneck: the performance gap stems from structural limitations rather than data scarcity. Adding 600,000 training pairs from MNLI and FEVER-NLI yields only $+0.6$ points on ANLI (see Section 6.4).

4. Ablation studies to isolate key factors: (i) the contribution of external knowledge from `ConceptNet`, and (ii) the combination of graph and text models (see Section 6.3).

2 Related Work

Leveraging external knowledge for Natural Language Understanding (NLU) tasks is a well-established approach. In Question Answering (QA), systems like QA-GNN (Yasunaga et al., 2021) and KagNet (Lin et al., 2019) bridge the semantic gap between questions and candidate answers by integrating `ConceptNet` (Speer et al., 2017) as an external knowledge base. For Natural Language Inference (NLI), KIM (Chen et al., 2018) enhances lexical understanding by incorporating WordNet-based relations, such as synonymy,

[1] Code: `https://github.com/l-pommeret/graph-nli`; see Appendix B.

[2] We use ConceptNet 5.7. The 28 relations are those selected for our extraction schema (Appendix A); ConceptNet contains additional relations.

antonymy, hyperonymy, and hyponymy. Similarly, KGNLI (Wang et al., 2020) extracts key concepts from premises and hypotheses to construct a knowledge subgraph linking the two.

While decomposing premises and hypotheses into atomic propositions improves interpretability and enables the diagnosis of logical flaws, this approach alone does not inherently boost accuracy without fine-tuning the models (Srikanth and Rudinger, 2025; Huang, 2026). The pipeline described in Section 3 addresses this limitation by making the atomic propositions the input of the triplet extraction stage.

Closest to our setting are the verification systems whose verdict is a function of an explicit structure rather than of the sentences themselves. ProoFVer (Krishna et al., 2022) generates a natural-logic proof — a sequence of lexical mutations between spans of the claim and of the retrieved evidence, each marked with a natural-logic operator — and reads the verdict off the operator sequence alone, so that the explanation is faithful by construction. Yuan and Vlachos (2024) decompose claim and evidence into semantic triples, augment them with an external knowledge graph, and pass the result to an off-the-shelf NLI model, in a zero-shot setting. We share with both previous works the commitment to a decision that depends on the structure alone, and we differ on three points: the decomposition goes through atomic propositions before the triplet extraction, the relation inventory is a closed enumeration of 28 `ConceptNet` relations enforced by constrained decoding rather than an open predicate set, and the classifier is fine-tuned on the serialised graphs, which is what makes the *ceteris paribus* comparison with a text model of the same backbone possible.

Finally, our classifier processes graphs as serialised text. Serialising structured inputs for language models rather than encoding them with a dedicated graph network is now a common choice (Fatemi et al., 2024), and we follow it: it leverages the pretrained semantics of relation names and ensures that each triplet remains individually accessible to the model's attention mechanism. While this approach offers clear advantages, a systematic comparison with graph encoders remains an avenue for future work.

3 Proposed Approach

Atomic facts are first extracted from both the premise and the hypothesis, and only then converted into *(subject, relation, object)* triplets. This decomposition improves triplet extraction (Pommeret et al., 2026) by simplifying the task assigned to the extractor: rather than isolating multiple facts from a complex sentence in a single pass, it processes one minimal semantic unit at a time, which reduces the number of facts to extract per call and ensures that no atomic fact from the source sentence is omitted.

The pipeline maps a pair (premise, hypothesis) to a label in the set {entailment, neutral, contradiction} through four stages, so that no original text from the pair takes part in the final decision (see Figure 1).

1. **Atomisation**. Both elements from the pair are decomposed into atomic propositions by `MPropositionneur-V2-large`, a model distilled for multilingual atomisation with coreference resolution (Pommeret et al., 2026).

2. **Triplet extraction.** Each atomic proposition is converted into a set of (subject, relation, object) triplets by `Qwen3.5-9B`[3] under a constrained JSON schema whose relation field is an enumeration of 28 `ConceptNet` relations. The union of the triplets of its propositions forms the graph P of the premise, and the graph H of the hypothesis.

3. **Knowledge enhancement.** A third graph, K, is retrieved from the 3.3M edges of the `ConceptNet` base: bridge edges (1 and 2 hops) between the entities that occur in P but not in H, and that occur in H but not in P, plus a one-hop neighbourhood of each entity.

4. **Classification.** The three graphs are serialised into a single string, in the following order: K, P, H, and fed to `Qwen3.5-0.8B-Base`,[4] fine-tuned on the training sets of different NLI benchmarks.

Why we do not give raw text to the LLM. A model that reads both the text and the graph will use the graph only where the text is insufficient. Thus, any interpretation of its graph usage is *post-hoc*.

[3] https://huggingface.co/Qwen/Qwen3.5-9B

[4] https://huggingface.co/Qwen/Qwen3.5-0.8B-Base

By removing the text, we make the intermediate structure decisive in the sense that *an error in the graph is an error in the prediction*. Conversely, every prediction can be traced to a finite list of triplets. The cost of the interpretability choice is presented in Table 3.

4 Experimental Protocol

This section follows the pipeline in the order in which it is applied. Section 4.1 covers graph construction, first the atomisation of the sentences into propositions (Section 4.1.1), then the extraction of triplets from each proposition (Section 4.1.2); Section 4.2 describes the retrieval of the external subgraph K; and Section 4.3 the serialisation of the three graphs and the fine-tuning of the classifier. Each stage constrains the next: the propositions bound what the extractor can see, and the extracted entities bound what can be retrieved from `ConceptNet`.

4.1 Graph Construction

4.1.1 Propositionneur

Sentences are atomised by `MPropositionneur-V2-large`, a fine-tuning of `Qwen3`, distilled from a larger teacher on Wikipedia chunks in six European languages (Pommeret et al., 2026). The model is prompted with `Atomize: {sentence}` and its output is constrained to a JSON array of strings under `vLLM` (Kwon et al., 2023). Decoding is greedy (temperature $T = 0$).

Long adversarial premises from ANLI make the propositionneur degenerate into loops of repetition in 7% of ANLI sentences. We therefore apply a cleaning step: case-insensitive and whitespace-insensitive deduplication, with removal of any proposition longer than 1.6 times the source sentence plus twenty characters, and a cap of 16 propositions per sentence, with a fallback to the source sentence when the filter empties the list. SNLI and ANLI (R1, R2 and R3) have been atomised.

4.1.2 Information Extraction

Extraction. Extraction uses `Qwen3.5-9B` served by `vLLM`, with few-shot prompting: the prompt provides the list of relations with their descriptions, a set of encoding rules, and nine annotated examples (see Appendix A). Importantly, decoding is constrained by a JSON schema in which the relation field is an `enum` over the 28 relations, so out-of-vocabulary relations are impossible by design. The schema also limits the output to 8 triplets: the prompt asks for at least two, while the schema enforces at least one.

Relation vocabulary. The extraction vocabulary contains 28 relations, all of them `ConceptNet` relations (Speer et al., 2017). Using only `ConceptNet` relation names has benefits: the extracted graphs are directly alignable with the external base used at stage 3, and the relation names carry pretrained semantics that the classifier (fine-tuned LLM) can exploit.

Prompt rules. The extractions produce graphs that are locally plausible, but globally problematic, because the same fact is encoded differently in P and H, and the two graphs fail to align it. Therefore, the prompt fixes a normal form, which is enforced by the following rules:

1. **The nodes are lemmas.** A node is a lowercase English lemma reduced to its head word: modifiers are represented separately (*e.g.*, `old_man` is forbidden, and `man` plus `[man, HasProperty, old]` is required). Multiword nodes are reserved for proper names and lexical compounds. Non-intersective adjectives (*e.g.*, `former_senator`, `fake_gun`) are explicit exceptions, since detaching them would change the truth conditions. Verbs and their objects are represented separately, except when the negation rule below requires a full verb phrase.

2. **Argument structure is fixed.** The agent of a verb always takes `CapableOf` and the patient always `ReceivesAction`; both are from a transitive verb. The passive and active variants are normalised to a unique form, so that *the book was written* and *someone wrote the book* yield the same triplet.

3. **Adjuncts are typed.** Beneficiaries and recipients use `HasContext`, destinations `MotivatedByGoal`, locations `AtLocation`, instruments `UsedFor`, and temporal expressions `HasContext`.

4. **Cardinality is an object.** For example *two men* is expressed as `[man, HasProperty, two]`, but a bare plural expresses no numeral.

5. **Negation is a relation.** Negated facts use the Not* relations with the full verb phrase as a

tail. This is what allows contradiction to be visible in the graph.

6. **Created works have a fixed direction.** The created work is always the subject of `CreatedBy`.

Every content word of the proposition source must give (at least) one triple. After the decoding, the nodes are normalised to the `ConceptNet` form with the `spaCy` lemmatiser.[5] Function words are dropped (except `not`, `no` and `never`). The remaining tokens are lemmatised, lower-cased and joined by underscores.

Prompt validation. The rules described above were not written *a priori*. They are the result of two corrective iterations driven by an LLM-as-judge panel (with three evaluation lenses over 36 items). That evaluation surfaced five systematic failures in the first prompt: hallucinated cardinality on bare plurals, an agent/patient inversion, the beneficiaries encoded as the patients, worn or printed identifiers encoded as quantities, and an inversion of the `CreatedBy` direction.

4.2 Knowledge Enhancement

The knowledge base is `ConceptNet 5.7`, restricted to English assertions, excluding the relations from DBpedia, and the relation `ExternalURL` (that is not used here). High-degree hubs are truncated to their 64 best neighbours, ranked by informativeness first and by weight second, so that a frequent node (*e.g.*, `man`) does not invade every subgraph.

For a pair, let $\mathcal{P}$ and $\mathcal{H}$ be the sets of the entities that occur in P and H, anchored to `ConceptNet` by exact match, or, if it does not suffice, by lexical head (*e.g.*, `winter_hat` becomes `hat`). The subgraph K is built in three passes:

1. **Direct bridges.** Edges linking an entity of $\mathcal{P} \setminus \mathcal{H}$ (*i.e.*, entities of $\mathcal{P}$ that are not in $\mathcal{H}$), to an entity of $\mathcal{H} \setminus \mathcal{P}$, ranked by weight. Shared entities are excluded with a reason: they are already aligned by string identity, so a bridge between them carries no information.

2. **Two-hop bridges.** Paths $p \to m \to h$ with $p \in \mathcal{P} \setminus \mathcal{H}, h \in \mathcal{H} \setminus \mathcal{P}$ and the pivot m outside $\mathcal{P} \cup \mathcal{H}$, ranked by the weight of their weakest edge. This pass is intended to capture transitive evidence such as `london PartOf england PartOf europe`, which no single triplet of P or H contains.

3. **Informative neighbourhood.** Up to 5 one-hop edges per entity, and excluding lexical relations (as `RelatedTo`, `FormOf`, `DerivedFrom`, `Synonym`, `SimilarTo`). Lexical relations are admitted as bridges, where they are genuinely informative, but not as background, where they only add noise.

Bridges are capped at 30 direct edges and 60 two-hop edges, the neighbourhood at 5 edges per entity, and K at 80 edges in total. Note that K may contain eight relations the extractor cannot produce (`RelatedTo`, `Entails`, `FormOf`, `DerivedFrom`, `ObstructedBy`, `HasFirstSubevent`, `HasLastSubevent`, `SymbolOf`), so the serialised input uses 36 relation types in total.

4.3 Serialisation and Fine-tuning of Qwen

Each pair is rendered as three labelled sections, with the triplets separated by semicolons:

```
knowledge: london PartOf england ; ...
 premise: woman HasProperty two ; ...
hypothesis: sister CapableOf hug ; ...
```

This ordering places external knowledge K first, followed by the premise graph P, and finally the hypothesis graph H, closest to the classification head. Truncation is on the left, so an input that exceeds the length limit loses external knowledge before premise information, and premise information before the hypothesis.

What an auditor sees. Figure 2 follows one pair through the four stages. It is the whole of what the classifier receives, and therefore the whole of what has to be checked to accept or reject the prediction. Auditing it is a finite task with three questions, each answerable without running the model. Does every triplet of P and H follow from its atomic proposition? Does every atomic proposition follow from its sentence? Is the bridge in K that carries the decision — here `girl IsA child` and `park AtLocation outdoors`, without which nothing in P meets `child` or `outdoors` — a licit inference step? A wrong prediction is thus localised to a specific line rather than attributed to the model as a whole, and a corrected line can be fed back through the classifier unchanged. This is the sense in which we use *interpretable*: the object of the audit is the input, not a saliency map over it.

[5] https://spacy.io/api/lemmatizer

Pair. *premise:* Two young girls are playing soccer in a park while their coach watches. — *hypothesis:* The children are outdoors. — *gold:* entailment.

1. Atomic propositions.

P_1 Two young girls are playing soccer.
P_2 The girls are in a park.
P_3 The coach of the girls watches the girls.
H_1 The children are outdoors.

2–3. Serialised input to the classifier.

```
knowledge: girl IsA child ; park AtLocation outdoors ; soccer AtLocation park
premise: girl HasProperty two ; girl HasProperty young ; girl CapableOf play ; soccer ReceivesAction play ; play AtLocation park ; coach CapableOf watch ; girl ReceivesAction watch ; girl HasA coach
hypothesis: child AtLocation outdoors
```

4. Prediction. entailment.

Figure 2: A pair carried through the pipeline. No token of the two sentences reaches stage 4: the decision rests on the twelve triplets above. The entailment is not derivable from P and H alone — `child` and `outdoors` occur nowhere in P — and is licensed by the two bridge edges retrieved in K.

The classifier is `Qwen3.5-0.8B-Base`, with a three-way sequence classification head. Training uses a batch size of 8, with 8 gradient accumulation steps (so an effective batch size of 64), the learning rate is 2e-5, with cosine scheduler and a 3% warmup. For regularisation, we use a weight decay of 0.01, on 2 epochs. Training is in `bf16`, with a maximum sequence length of 1024 tokens (1536 on ANLI, whose premises are longer).

Checkpoint selection. The checkpoint is selected on the *validation* split of the dataset, not on the *test*. All figures below have been obtained following this protocol.

5 Datasets and Benchmarks

The two evaluation benchmarks are chosen because they sit at the two ends of the range over which a structural bottleneck can be measured. SNLI (Bowman et al., 2015) is the favourable end: its premises are single-sentence image captions describing one visible event, with a short concrete vocabulary, few named entities and almost no temporal or modal content, so the normal form of Section 4.1.2 has a faithful target for nearly every content word. ANLI (Nie et al., 2020) is the adversarial end: its premises are multi-sentence passages, drawn from Wikipedia for R1 and R2 and from a wider range of genres for R3, and its hypotheses were written by annotators with the explicit goal of defeating a strong text model, so they turn on exactly the fine-grained detail — a date, a number, a comparison, an embedded clause — that a fixed triplet vocabulary is likely to compress away. Holding the pipeline and the backbone fixed across the two, the gap moves from 1.9 to 14 points as the input moves from captions to adversarial passages, which is what isolates the cost of the representation rather than of the model.

For the data-scaling experiment of Section 6.4 only, we additionally extract MNLI (Williams et al., 2018), which adds ten written and spoken genres, and FEVER-NLI (Thorne et al., 2018; Nie et al., 2019), which adds retrieved encyclopaedic evidence, using the evidence as the premise. Together they contribute about 600,000 further training pairs. They are used as additional training data, never as test sets.

The full SNLI training set (549,367 pairs) is processed, yielding 889,944 unique atomic propositions. ANLI R1 to R3 yield 296,585 atomic propositions. MNLI has 953,734 unique propositions, and FEVER-NLI 507,593. Extraction produces an average of 3.6 triplets per proposition.

5.1 Metrics

We report accuracy. ANLI test sets contain 1000 to 1200 examples (depending on the round).

5.2 Defining the Price of Interpretability

We define as *price of interpretability* the drop between the original fine-tuning of `Qwen3.5-0.8B` on the texts of premise and hypothesis (with the format `premise: ...\nhypothesis: ...`) and the *ceteris paribus* fine-tuning of `Qwen3.5-0.8B` only on graphs extracted as we show in Section 4.1.2. We use the same backbone, optimiser and nominal number of epochs (see Appendix B); differences in seed coverage and completed training budgets are discussed in the Limitations section. The input is raw premise and hypothesis text for the text model, and serialised graphs for the graph model. The quantity is therefore not a comparison against the state-of-the-art, but a measurement of what the graph-interpretability costs. Since the text baseline is a single run while the graph model has three seeds, the SNLI price is defined against the three-seed mean.

6 Results and Analysis

6.1 Main Results

Table 1 reports SNLI. The graph pipeline reaches 0.897 ± 0.006 accuracy over three seeds. Below it, the zero-shot control matters for interpreting all of this. Without fine-tuning, the 0.8B base model scored by label likelihood is at chance on every split (0.343 on SNLI, 0.334 to 0.335 on ANLI). The zero-shot score is the same whether the input is text or graphs. All the competence reported below comes from fine-tuning, and none of it from the backbone model.

Table 2 reports ANLI. Transferring the SNLI model zero-shot gives 0.285 for R1, 0.330 for R2 and 0.328 for R3, that is at, or below, chance, which is expected because ANLI is designed to produce examples that fool models which are good at SNLI. After a second training phase, on ANLI R1 to R3, starting from the SNLI checkpoint with results shown in Table 1, the graph model reaches 0.576, 0.480 and 0.449, *i.e.*, +29.1 points on R1 over the zero-shot transfer, with a graph-only input. Sequential fine-tuning costs SNLI retention (dropping from 0.892 to 0.782).

Two readings of Table 2 pull in two directions: against the literature, the graph-only model is within one point of the published `RoBERTa-large` (Liu et al., 2019)[6] test accuracy on R2 (0.480 vs. 0.489) and slightly above it on R3 (0.449 vs. 0.444), while remaining 16 points behind on R1 (0.576 vs. 0.738) (Nie et al., 2020); it sits at the level of `BERT-large` (Devlin et al., 2019) on all three rounds (0.574, 0.483 and 0.435). An interpretable-by-design system that never sees the text is therefore competitive with strong text encoders on the two hardest rounds. Against the *ceteris paribus* control, the same graphs lose 9 to 14 points to their own text counterpart. The interpretability gap only means something when measured against the same backbone, so this is what we call the price of interpretability.

6.2 The Price of Interpretability

Table 3 quantifies the trade-off. The price is small on SNLI (we see that the graph pipeline retains 97.9% of the text model's accuracy), and big on ANLI, where it retains only 80% on R1. The adversarial signal lives in what the structure of triplet extraction compresses away.

[6] `https://huggingface.co/FacebookAI/roberta-large`

System	Input	SNLI
Zero-shot 0.8B-Base	text	0.343
Zero-shot 0.8B-Base	graphs	0.343
RoBERTa-large (ours)	text	0.899
Qwen3.5-0.8B (ours)	graphs	0.897 ± 0.006
Qwen3.5-0.8B	text	0.916
Qwen3.5-0.8B	text+graphs	**0.921**

Table 1: SNLI test accuracy. Our graph model is the mean over three seeds (0.892, 0.894 and 0.904). All other rows are seed 0.

Model	R1	R2	R3	SNLI
Graphs only				
Zero-shot transfer	0.285	0.330	0.328	0.892
Train on ANLI only	**0.576**	**0.480**	0.449	0.782
ANLI + SNLI replay	0.565	0.462	**0.463**	**0.887**
Text				
RoBERTa-large (ours)	0.580	0.359	0.357	0.899
RoBERTa-large (published)	0.738	0.489	0.444	–
BERT-large (published)	0.574	0.483	0.435	–
Qwen3.5-0.8B	0.704	0.572	0.553	0.910

Table 2: ANLI accuracy, with SNLI transfer in the last column. Bold marks the best graph-only system per column. The published `RoBERTa-large` and `BERT-large` results are those of Nie et al. (2020).

By inspecting the disagreements, we see four recurring categories accounting for most of the gap. Numerical reasoning (or order reasoning) survives only when the numeral is explicit. The prompt expressly forbids inventing one from a bare plural. Temporal reasoning is flattened: dates become undifferentiated with the `HasContext` relation, so *since 1953* and *in 1953* are indistinguishable (see the final paragraph of the Limitations section). Also, the coreferences on long premises are resolved by the propositionneur, and in case of error, it propagates through the pipeline. Finally, ANLI premises are long, and the fine-grained detail on which an adversarial hypothesis hinges is often the modifier that the normal form strips or the clause that the 8-triple cap drops.

6.3 Ablations

Table 4 reports two ablations.

External knowledge is neutral on SNLI. We see that removing K (that is, the graph coming from `ConceptNet`) costs nothing (0.895 against 0.892). We can take it as a negative result: SNLI inference is so simple that the premise and hypothesis graphs suffice, and the retrieved `ConceptNet` subgraph is (at best) redundant. It does not follow that K is useless in general. But on SNLI our multi-hop retrieval carries nothing.

	Graph	Text	Price of Interp.
SNLI	0.897 ±0.006	**0.916**	−1.9
ANLI R1	0.565	**0.704**	−13.9
ANLI R2	0.462	**0.572**	−11.0
ANLI R3	0.463	**0.553**	−9.0

Table 3: Price of Interpretability: `Qwen3.5-0.8B` on raw text (`premise: ...\nhypothesis: ...`) vs. the graph model, *ceteris paribus*.

Configuration (SNLI test)	Accuracy
$P + H + K$ (full, graphs only)	0.892
$P + H$, no external knowledge	0.895
Text only	0.916
Text + graphs	**0.921**

Table 4: Ablations, all with the same `Qwen3.5-0.8B` backbone and protocol (seed 0).

Graphs and text are complementary. Adding the graphs to the text raises the accuracy from 0.916 to 0.921. As a *complement*, the graphs contribute.

6.4 Is the Gap a Data Problem?

A natural objection to Table 3 is that the graph model is simply under-trained: graphs are a less familiar input format, so it may need more data to reach the same performance. We tested this directly. We extracted MNLI and FEVER-NLI with exactly the same pipeline, and ran a classical (in the literature) two-phase recipe: phase 1 on SNLI + MNLI + FEVER, and phase 2 on ANLI with SNLI replay. Phase 1 improves SNLI a little (0.896 against 0.892). Phase 2 on ANLI with SNLI replay then gives 0.568, 0.479 and 0.461 on R1 to R3, against 0.565, 0.462 and 0.463 without the additional data. We read this as evidence that the adversarial gap is representational, and not data-limited.

7 Conclusion

The answer to our title question (can we do interpretable NLI with graphs based on atomic propositions?) is: yes on SNLI, and only partly on ANLI. We built an NLI system whose classifier never sees a natural-language sentence, and measured the performance cost of relying on graphs of atomic propositions alone. On SNLI, the cost is 1.9 points. On ANLI it is 9 to 14, depending on the ANLI round. The same system matches the published `RoBERTa-large` text model on ANLI R2 and R3 while taking no text as input.

Beyond that measurement, the pipeline itself is the contribution: atomic propositions are carried forward to the decision, and the combination of the `ConceptNet` vocabulary, the constrained extraction, and the serialised three-graph input makes every prediction traceable to a finite list of triplets. The cost of that guarantee is now measured, which is what a neuro-symbolic model needs if its intermediate representations are to be inspected by humans.

The practical consequence is that the `ConceptNet` triplet vocabulary, as we use it, is lossy in identifiable ways: numeral mentions, the temporal granularity, and the fine details of long premises. To close this gap, we therefore have to enrich the target structure, rather than scaling the classifier. Finally, since the interpretability we claim is the auditability of the intermediate structures (atomic propositions and triplets), it should be validated as such by a human study in which annotators repair a wrong prediction by editing the triplets. That would test the claim directly.

Limitations

The interpretability is qualitative, not quantitative. We do not measure the interpretability (*i.e.*, the readability by humans) of atomic propositions and triplets of our pipeline explicitly. It remains to be tested in future work.

Validation of the graph by the evaluation. We do not directly evaluate the quality of the graph, but evaluate it indirectly by means of NLI benchmarks. We want to conduct further evaluation on the quality of the graph, using other domains, or a direct approach.

Upstream errors are unrecoverable. The guarantee that an error in the graph is an error in the prediction has a symmetric cost: the classifier has no text to fall back on, so any failure of the two upstream models propagates unchecked. Two are known and unmeasured. The atomiser degenerates into repetition loops on 7% of ANLI sentences; our cleaning step removes the loops but is itself lossy, since the length filter, the cap of 16 propositions per sentence and the fallback to the source sentence can each drop a fact that the source did contain, and we do not measure how often they do. Coreference is resolved inside the atomiser, on long premises where it is hardest, and a wrong antecedent is silently frozen into the triplets of P. Quantifying the two — by scoring propositions and triplets against a reference, rather than only through

downstream accuracy — is the most direct way to separate the price of the representation from the price of the extractors that build it.

Only English. The propositionneur is multilingual, but the extraction vocabulary, the `ConceptNet` base, and every experiment, are in English. ConceptNet's coverage is culturally biased.

Single-seed baselines and unequal budgets. The graph model is averaged over three seeds, but the text baselines and all ANLI models are single runs; two of the three graph seeds were also interrupted before the end of the second epoch. The SNLI price of interpretability should therefore be read as an upper bound of roughly two points, of the same order as run-to-run variation. The ANLI gap (9 to 14 points) is an order of magnitude larger and is unaffected.

The contribution of K is measured on SNLI only. We did not run the no-K ablation on ANLI, so the usefulness of multi-hop `ConceptNet` retrieval for adversarial inference remains open. The SNLI result does not settle it, and arguably says little about it: SNLI premises and hypotheses share most of their vocabulary, so $\mathcal{P} \setminus \mathcal{H}$ and $\mathcal{H} \setminus \mathcal{P}$ are small and there is little for a bridge to connect, whereas ANLI hypotheses are written to avoid lexical overlap, which is precisely the configuration the two-hop pass was designed for. In future work, we will assess whether K helps there, or whether its 80 edges mostly add noise to an already truncated input.

What the vocabulary can and cannot express. The inventory is closed by construction: the `enum` makes an out-of-vocabulary relation impossible, so every predicate of the source proposition is either mapped onto one of the 28 relations or lost. The normal form of Section 4.1.2 absorbs the bulk of English predication — argument structure, adjunction, possession, class membership, negation — which is why the loss is almost invisible on SNLI. Three families of meaning, however, have no faithful target. First, temporal and spatial granularity collapses onto `HasContext` and `AtLocation`, so *since 1953* and *in 1953* become indistinguishable. Second, scalar and comparative predication (*taller than*, *most of*, *at least three*) has no relation of its own, and cardinality survives only as a `HasProperty` object on an explicit numeral. Third, propositional attitudes and modality (*X believes that Y*, *X may Y*) are flattened, since no relation embeds a proposition under an operator. Section 6.2 shows that the first two families, together with coreference errors and the truncation of long premises, account for most of the residual gap on ANLI, where they are frequent, and are rare in SNLI captions.

Acknowledgments

This work was supported by the French National Research Agency (ANR) through the EQUATION project (ANR-25-CE23-2916). The authors acknowledge the computing resources provided by the Lab-IA cluster of the Saclay-IA platform within the Mésocentre Paris-Saclay at Université Paris-Saclay.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. 2018. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Minghui Huang. 2026. Atomic-snli: Fine-grained natural language inference through atomic fact decomposition. *Preprint*, arXiv:2601.06528.

Amrith Krishna, Sebastian Riedel, and Andreas Vlachos. 2022. ProoFVer: Natural logic theorem proving for fact verification. *Transactions of the Association for Computational Linguistics*, 10:1013–1030.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*, pages 611–626. Association for Computing Machinery.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *Preprint*, arXiv:1907.11692.

Yixin Nie, Haonan Chen, and Mohit Bansal. 2019. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6859–6866. AAAI Press.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Luc Pommeret, Thomas Gerald, Christophe Servan, Sahar Ghannay, Patrick Paroubek, and Sophie Rosset. 2026. LLM-based atomic propositions help weak extractors: Evaluation of a propositioner for triplet extraction. In *Proceedings of the Knowledge Graphs and Large Language Models Workshop (KG-LLM) @ LREC26*, pages 134–143, Palma, Mallorca, Spain. European Language Resources Association (ELRA).

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 4444–4451, San Francisco, California, USA. AAAI Press.

Neha Srikanth and Rachel Rudinger. 2025. NLI under the microscope: What atomic hypothesis decomposition reveals. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2574–2589, Albuquerque, New Mexico. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Zikang Wang, Linjing Li, and Daniel Zeng. 2020. Knowledge-enhanced natural language inference based on knowledge graphs. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6498–6508, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Moy Yuan and Andreas Vlachos. 2024. Zero-shot fact-checking with semantic triples and knowledge graphs. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 105–115, Bangkok, Thailand. Association for Computational Linguistics.

A Extraction Prompt

The prompt sent to the extractor is reproduced here in four parts. Figure 3 gives the task statement and the 28-relation vocabulary, Figure 4 the normal form imposed on nodes and on argument structure, Figure 5 the nine few-shot pairs, and Figure 6 the JSON schema passed to the constrained decoder.

B Reproduction Code

The code is available at `https://github.com/l-pommeret/graph-nli`. Commands assume the ConceptNet 5.7 English assertions in

```
You convert ONE English sentence (an atomic proposition) into knowledge-graph triples using
  the ConceptNet vocabulary. Output a JSON array of triples [{"h": head, "r": relation, "t":
  tail}, ...] and nothing else.
RELATIONS -- use ONLY these ConceptNet relations:

- IsA: class or type membership: [dog, IsA, animal]
- InstanceOf: named entity is an instance of a class: [parma, InstanceOf, city]
- HasProperty: attribute, state or cardinality: [coat, HasProperty, red] ; [man, HasProperty, two]
- NotHasProperty: negated attribute: [room, NotHasProperty, empty]
- CapableOf: the subject performs the action: [person, CapableOf, jump]
- NotCapableOf: the subject does NOT perform the action: [politician, NotCapableOf, attend_meeting]
- ReceivesAction: the object undergoes the action: [ball, ReceivesAction, kick]
- AtLocation: spatial location: [man, AtLocation, park] ; [jump, AtLocation, street]
- LocatedNear: proximity: [bench, LocatedNear, fountain]
- HasA: possession or attachment: [person, HasA, horse]
- PartOf: part-whole: [wheel, PartOf, car]
- MadeOf: material: [knife, MadeOf, steel]
- UsedFor: purpose or instrument of an action: [knife, UsedFor, cut]
- Desires: wants / intends: [child, Desires, ice_cream]
- NotDesires: does not want: [cat, NotDesires, bath]
- CausesDesire: makes one want: [heat, CausesDesire, swim]
- Causes: causal link: [rain, Causes, wet]
- HasSubevent: sub-action happening within the action: [cook, HasSubevent, stir]
- HasPrerequisite: necessary precondition: [win, HasPrerequisite, compete]
- MotivatedByGoal: action done for a goal: [train, MotivatedByGoal, competition]
- MannerOf: specific way of doing (verb-verb): [sprint, MannerOf, run]
- HasContext: temporal or contextual setting: [sleep, HasContext, night]
- CreatedBy: creator or origin: [painting, CreatedBy, artist]
- DefinedAs: definition or equivalence: [sum, DefinedAs, total]
- Synonym: same meaning: [couch, Synonym, sofa]
- Antonym: opposite meaning: [hot, Antonym, cold]
- SimilarTo: similar: [jog, SimilarTo, run]
- DistinctFrom: mutually exclusive alternative: [indoors, DistinctFrom, outdoors]
```

Figure 3: System prompt, part 1: task statement and the 28-relation ConceptNet extraction vocabulary.

`data/conceptnet/`. Sequential fine-tuning starts from the SNLI checkpoint selected on the validation split. The atomic proposition datasets are released on the HuggingFace Hub.[7] Figure 7 gives the end-to-end commands.

[7] https://huggingface.co/datasets/Younes2E/atomic-snli and https://huggingface.co/datasets/Younes2E/atomic-anli

```
NODE RULES:
- a node is a lowercase English lemma: the bare HEAD word, without its modifiers
- NEVER glue modifier+noun or verb+object into one node (wrong: old_man, go_package,
  wear_jacket); each modifier becomes its own HasProperty triple on the head noun
- multi-word nodes ONLY for proper names (new_york, forza_italia) and true lexical compounds
  (winter_hat, hot_dog); exception: non-intersective adjectives stay attached
  (former_senator, fake_gun)
- verbs: bare lemma (jump, not jumps/jumping); phrasal verbs keep their particle (jump_over)
- nouns keep their lexical form (linguistics stays linguistics); never use bare adverbs or
  prepositions as nodes (outside, just, very)
- numerals ONLY if the number is explicit: "two men" -> [man, HasProperty, two]; a bare
  plural ("the sisters") NEVER yields a numeral
- worn or printed identifiers are not quantities: "the number 2 jersey" -> [jersey,
  HasProperty, number_2]
STRUCTURE -- every content word of the sentence must land in at least one triple:
- agent of a verb: [agent, CapableOf, verb] -- the agent ALWAYS takes CapableOf, never
  HasProperty and never ReceivesAction
- direct object: [object, ReceivesAction, verb] -- for every transitive verb emit BOTH the
  agent triple AND the patient triple
- passive voice: the subject is the patient ("the book was written" -> [book,
  ReceivesAction, write]); a "by X" agent gets [X, CapableOf, verb]
- recipient or beneficiary ("to/for X"): [verb, HasContext, X] -- NEVER ReceivesAction
- destination ("walk to school"): [verb, MotivatedByGoal, school] -- not AtLocation
- place where it happens: [x, AtLocation, place]; time, date or period: [x, HasContext,
  time] -- dates are context, never HasProperty
- instrument: [instrument, UsedFor, verb] -- purpose: [verb, MotivatedByGoal, goal]
- possession: [owner, HasA, thing] -- part or aspect of: [part, PartOf, whole]
- role or profession: [person, IsA, role]; named entity class: [name, InstanceOf, class]
- created works: [work, CreatedBy, creator] -- the WORK is always the head, in active voice
  too: "Panini wrote a grammar" -> [grammar, CreatedBy, panini], NEVER [panini, CreatedBy,
  grammar]
- "outside/near X": attach the real place directly ([fight, LocatedNear, deli]), never a
  bare "outside" node
- NEGATION: use NotCapableOf / NotHasProperty / NotDesires, with the full verb phrase as
  tail when needed ([politician, NotCapableOf, attend_meeting]); NEVER create not_* nodes.
Produce 2 to 8 triples.
```

Figure 4: System prompt, part 2: the normal form imposed on nodes and on argument structure.

```
IN : A person is training his horse for a competition.
OUT: [person, CapableOf, train] [horse, ReceivesAction, train] [person, HasA, horse] [train,
     MotivatedByGoal, competition]

IN : Two young girls are playing soccer in the park.
OUT: [girl, HasProperty, two] [girl, HasProperty, young] [girl, CapableOf, play] [soccer,
     ReceivesAction, play] [play, AtLocation, park]

IN : The chef is cutting fresh bread with a steel knife.
OUT: [chef, CapableOf, cut] [bread, ReceivesAction, cut] [bread, HasProperty, fresh] [knife,
     UsedFor, cut] [knife, MadeOf, steel]

IN : The politicians did not attend the meeting.
OUT: [politician, NotCapableOf, attend_meeting]

IN : The sisters are selling lemonade to a customer.
OUT: [sister, CapableOf, sell] [lemonade, ReceivesAction, sell] [sell, HasContext, customer]

IN : A boy wearing a jersey with the number 2 walks to school.
OUT: [boy, CapableOf, wear] [jersey, ReceivesAction, wear] [jersey, HasProperty, number_2]
     [boy, CapableOf, walk] [walk, MotivatedByGoal, school]

IN : The Indian grammarian Panini wrote the Ashtadhyayi, a description of Sanskrit, in the
  4th century BCE.
OUT: [panini, IsA, grammarian] [panini, HasProperty, indian] [panini, CapableOf, write]
     [ashtadhyayi, ReceivesAction, write] [ashtadhyayi, CreatedBy, panini] [ashtadhyayi,
     DefinedAs, description] [description, HasContext, sanskrit] [write, HasContext,
     4th_century_bce]

IN : The Parma trolleybus system has been in operation since 1953.
OUT: [parma_trolleybus_system, InstanceOf, trolleybus_system] [parma_trolleybus_system,
     AtLocation, parma] [parma_trolleybus_system, ReceivesAction, operate] [operate,
     HasContext, since_1953]

IN : A man sleeps on the couch because he is tired.
OUT: [man, CapableOf, sleep] [sleep, AtLocation, couch] [man, HasProperty, tired] [tired,
     Causes, sleep]
```

Figure 5: The nine few-shot pairs, in compact triple notation. They are sent as alternating user and assistant turns, the assistant turns containing JSON arrays of `{h, r, t}` objects.

```
{
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "h": {"type": "string", "maxLength": 60},
      "r": {"enum": [
        "IsA", "InstanceOf", "HasProperty", "NotHasProperty", "CapableOf",
        "NotCapableOf", "ReceivesAction", "AtLocation", "LocatedNear",
        "HasA", "PartOf", "MadeOf", "UsedFor", "Desires", "NotDesires",
        "CausesDesire", "Causes", "HasSubevent", "HasPrerequisite",
        "MotivatedByGoal", "MannerOf", "HasContext", "CreatedBy",
        "DefinedAs", "Synonym", "Antonym", "SimilarTo", "DistinctFrom"
      ]},
      "t": {"type": "string", "maxLength": 60}
    },
    "required": ["h", "r", "t"],
    "additionalProperties": false
  },
  "minItems": 1,
  "maxItems": 8
}
```

Figure 6: JSON schema passed to the constrained decoder. The `enum` lists the 28 extraction relations of Figure 3.

```
# 0. atomic propositions (released on the HuggingFace Hub)
huggingface-cli download --repo-type dataset Younes2E/atomic-snli --local-dir data/atomic/snli
huggingface-cli download --repo-type dataset Younes2E/atomic-anli --local-dir data/atomic/anli

# 1. triples (GPU, vLLM)
python -m nli.extract --inputs data/atomic/snli/{train,validation,test}.jsonl \
    --out out/snli/triples.jsonl

# 2. three-graph datasets (P, H, K)
python -m nli.dataset --atomic data/atomic/snli/train.jsonl \
    --triples out/snli/triples*.jsonl --out out/snli/train.dataset.jsonl

# 3. fine-tuning on SNLI, then on ANLI from the SNLI checkpoint
python -m nli.finetune --train out/snli/train.dataset.jsonl \
    --val out/snli/validation.dataset.jsonl --test out/snli/test.dataset.jsonl \
    --out out/models/snli-s0 --seed 0 --no-grad-ckpt --epochs 2 --max-len 1024
python -m nli.finetune --model out/models/snli-s0/checkpoint-8000 \
    --train out/anli/train_r{1,2,3}.dataset.jsonl \
    --val out/anli/dev_r{1,2,3}.dataset.jsonl \
    --test out/anli/test_r{1,2,3}.dataset.jsonl out/snli/test.dataset.jsonl \
    --out out/models/anli-phase2 --max-len 1536 --epochs 3
```

Figure 7: End-to-end pipeline, from atomic propositions to the fine-tuned classifier.

A Rhetorical Citation Graph Model for Characterizing Scientific Findings: Resource Construction and Application to Plant Health

Anne-Sophie Foussat[1,2,3], Robert Bossy[1], Vincent Guigue[2], Nicolas Sauvion[3], Claire Nédellec[1]
[1] Université Paris-Saclay, INRAE, MaIAGE, 78350 Jouy-en-Josas, France,
[2]AgroParisTech, MIA Paris-Saclay, Palaiseau, France,
[3]PHIM, Univ Montpellier, INRAE, CIRAD, IRD, Institut Agro, Montpellier, France
Correspondence: anne-sophie.foussat@inrae.fr

Abstract

Understanding how scientific findings are discussed and evaluated over time provides valuable evidence for assessing their reliability. We propose the Rhetorical Citation Graph (RCG), a finding-centric graph model that represents citation events by explicitly linking scientific findings, citing articles, rhetorical functions, and citation contexts. We instantiate this model as the Plant Literature Reliability Network (PLIN), a graph of 262,807 citation events automatically constructed from the plant-health literature using Natural Language Processing methods for citation and finding extraction, and rhetorical stance classification. By integrating provenance data with the rhetorical interpretation expressed in citation contexts, the RCG provides structured evidence for contextual analyses of scientific findings and lays the foundation for longitudinal studies of their evolution, scientific consensus, and reliability. Finally, we illustrate the utility of PLIN through a case study on the curation and enrichment of the EPPO Global Database, a reference knowledge base for plant health.

1 Introduction

As scientific production accelerates, distinguishing well-established findings from contested or weakly supported ones has become increasingly difficult. Nissen et al. (2016) describe *canonization* as the process by which a scientific claim becomes widely accepted as fact. In empirical sciences, however, knowledge evolves through replication, agreement, and controversy: a consensus may emerge, be challenged, and later be reformulated (Shwed and Bearman, 2010). Scientific knowledge is therefore dynamic and provisional, continually improved or corrected in light of technical advances and new evidence (Miller, 2013). A temporal perspective is consequently essential for identifying the trajectories through which consensus develops (Shwed and Bearman, 2010).

Citation networks provide a means of tracing the evolution of scientific knowledge. Bibliometric studies examine temporal patterns across papers, authors, journals, institutions, and fields (Yan and Ding, 2012). However, understanding rhetorical function of citations is essential to reveal how peers interpret, support, or challenge a finding over time. Such interpretation requires analysing the context surrounding each citation.

Citation counts alone cannot capture these interpretive relations and may lead to misleading conclusions about scientific quality (Moravcsik and Murugesan, 1975). Understanding how scientific consensus emerges therefore requires representing citation events at the level of individual scientific findings rather than publications. In this paper, we introduce the Rhetorical Citation Graph (RCG), a finding-centric graph model for representing citation events at the level of individual findings. In RCG, a citation is modeled as an event linking a cited finding to a citing article, together with its rhetorical function and the surrounding citation context. Findings are represented as normalized relational assertions together with original textual realization, enabling equivalent findings to be compared across publications while preserving traceability to their original formulation. RCG is constructed from scientific documents using Natural Language Processing methods for identifying citation contexts, extracting findings, and classifying rhetorical functions.

By integrating normalized findings, with citation contexts, provenance information and rhetorical functions, RCG provides structured evidence for studying the dynamics of consensus formation and supporting downstream reliability assessment.

We instantiate RCG as the Plant Literature Reliability Network (PLIN), a graph constructed from plant-health literature, and use it as a case study throughout the paper. The graph construction pipeline is organized into successive

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 247–260
October 28, 2026 ©2026 Association for Computational Linguistics

processing stages, for which we adapted language models to perform finding extraction and rhetorical function classification. The code used to construct the dataset and the graph is publicly available on GitHub.[1] Finally, through a case study on the curation and enrichment of the EPPO Global Database, we illustrate how PLIN provides structured evidence to support knowledge base curation by facilitating the interpretation of scientific findings.

2 Previous work

2.1 Mapping scientific knowledge with graphs

Previous research has proposed graph-based representations to model different aspects of the scientific literature, including citation networks and structured scientific knowledge. For example, (Viswanathan et al., 2021) combine citation graph embeddings with citances, i.e. the sentences containing the citation marker, to improve information extraction by incorporating inter-articles relationships. Rather than extracting information from individual papers, other studies focus on analyzing corpora of articles to map the evolution of knowledge. For instance, Luo et al. (2018) combine an influence-relation graph, representing causal-like relations between concepts extracted from children's health publications, with a citation graph to identify gaps in scientific knowledge. Similarly, Kargupta et al. (2025) propose a structured framework to explore the novelty of scientific claims.

2.2 Citation graph

Graph-based models of citation relationships differ in the types of entities and relations they emphasize. Berrebbi et al. (2022) proposes a graph representation in which nodes represent papers, authors, or venues, and edges are the citations between these entities. Quan et al. (2025) also propose a model that emphasizes multiple citation intents for the same citing-cited pair, since a single article may be cited in different sections and for different rhetorical purposes. However, in their representation, citations remain directly attached to the cited article, without distinguishing its different scientific contributions. Addressing this limitation, Liang et al. (2024) introduce a fine-grained representation by modeling both papers and their individual findings as nodes, with edges representing citation links between a finding and the citing article. This approach accounts for the fact that different findings within the same paper can be cited independently. However, in Liang et al.'s representation, the citation type is not characterized, and the model does not capture the possibility of multiple citation relations between the same finding and a citing article. Others normalize findings into structured semantic assertions to facilitate semantic comparison and integration Pramanick et al. (2026). Existing citation graph models, however, do not jointly represent normalized findings, individual citation events, and their rhetorical functions. Because an article may cite the same finding multiple times for different purposes, methods are needed to extract and normalize findings and link them to individual citation events.

2.3 Scientific Finding Extraction and Normalization

Scientific finding extraction methods often represent findings as textual spans extracted from articles. Their contextual embeddings are then compared to match statements across publications (Cohan and Goharian, 2017; Tan et al., 2026), supporting tasks such as citation contextualization, claim alignment, and miscitation detection. Such textual representations do not enable interpretable comparisons across publication statements or between statements and structured knowledge bases. Recent work has proposed richer semantic decompositions of scientific claims, for example by distinguishing their topic, stance, and claim object (Gangi Reddy et al., 2022), but these representations remain centered on the source document rather than on how claims are subsequently cited and interpreted. In contrast, scientific knowledge graph construction represents findings as normalized semantic relations or events, obtained through entity recognition, relation extraction, entity linking, and ontology grounding enabling comparison and integration across publications (Viswanathan et al., 2021). Recent prompt-based large language models have further improved scientific information extraction, including in plant health through the EPOP corpus (Nédellec et al., 2026). Although these methods provide the building blocks for extracting and normalizing scientific findings, they are generally developed independently of citation analysis and

[1] https://github.com/AnneSophie148/PhD/tree/main/new_dataset

therefore do not support modeling how normalized findings are interpreted through individual citation events.

2.4 The rhetorical citation classification task

Capturing citation events at a fine-grained level also requires characterizing their rhetorical function. Rhetorical citation classification, also referred to as citation intent classification (Zhang et al., 2025), aims to characterize the role of citations within the argumentative structure of scientific discourse. Early work relied on rule-based methods (Garzone and Mercer, 2000), which were later replaced by supervised machine learning approaches, using manually engineered linguistic features and document structure features (Teufel et al., 2006; Jurgens et al., 2018). More recently, pretrained language models such as SciBERT have become the dominant approach (Beltagy et al., 2019; Zhang et al., 2022; Jiang and Chen, 2023; Xu et al., 2025), while recent studies have investigated large language models under prompting and in-context learning settings (Kunnath et al., 2023; Koloveas et al., 2025). Other methods explore ways to incorporate relational information between citing and cited entities, using graph embeddings to model the citation intent (Berrebbi et al., 2022; Tsai et al., 2023; Quan et al., 2025). These approaches aim to capture structural properties of citation. While studies on rhetorical citation classification provide a rich characterization of citations, they rely on different annotation schemas with varying typologies and number of classes, thereby limiting cross-study comparison.

3 A Fine-Grained Rhetorical Citation graph

We propose the Rhetorical Citation graph (RCG) a fine-grained graph model that represents how individual scientific findings are cited and rhetorically interpreted over time. Its structure is shown in Figure 1. The central element of the graph is the *Citation Event*, which links a unique *In-context Citation* and *Finding* pair together with the corresponding citing and cited *Article* nodes. The explicit representation of *Citation Events* as nodes is a conceptual modeling choice that gives each unique (*In-context Citation*, *Finding*) association its own identity and makes it directly addressable as the unit of rhetorical analysis. A *Finding* node represents a scientific finding from a cited article. A single *Finding* node may participate in multiple Citation Events when the same finding is cited in different citation contexts. The *In-context Citation* node stores the citation passage, including a sentence-level decomposition that enables contextual analysis, the citation marker, and the cited findings extracted from the passage. Findings are represented as normalized relational assertions in which the two entity arguments are mapped to reference classes defined in hierarchical semantic resources while the relation is directed and labeled. We distinguish between original findings extracted from the article that states the finding, and cited findings extracted from the context in which another article cites the original article. *Citation Event* nodes store the result of the alignment between the original finding and cited findings, indicating whether one of the cited findings matches the original finding. This design preserves every citation event, including cases where no reliable alignment can be established. Modeling citation events as first-class nodes allows each association between a citation context and a scientific finding to be represented and characterized individually. A single citation context may give rise to multiple *Citation Event* nodes when it refers to several findings, while a finding may participate in multiple citation event nodes across different citation contexts or citing articles. Conversely, a cited article typically contains multiple findings that can each be cited independently. This representation preserves fine-grained citation patterns that are collapsed in conventional article-level citation graphs. *Rhetorical Class*, *In-context Citation*, *Section* and *Self-Citation* nodes are connected to *Citation Event* nodes, thereby enriching it with rhetorical contextual information. *Article* nodes stores bibliographic metadata, including DOI, authors, title, and year of publication, and are linked to the findings they report.

The *Rhetorical Class* nodes represent the rhetorical function expressed in a *Citation Event*. The *Comparison or Contrast* class is further refined into four additional categories that indicate whether the rhetorical function concerns methods or results, and distinguish *Comparison*, *Contrast* and *Contradictory results* in order to identify citation events expressing contrasting results. The *Self-Citation* nodes indicate whether the citation is a self-citation, with possible values “yes”, “no”, or “unknown”. The *Section* nodes specify the

structural section of the citation within the citing article.

4 RCG Automated construction

4.1 Implementation in plant health

To illustrate the application of the RCG graph model, we implemented it on a corpus of plant-health literature, resulting in the Plant Literature Reliability Network (PLIN). We focus on scientific publications describing pathogen transmission by insect vectors. This domain provides a coherent body of literature with relatively homogeneous citation practices while involving complex biological interactions that have been the subject of both scientific consensus and ongoing debate. It therefore offers an appropriate testbed for structuring citation events over time. The construction of the PLIN graph comprised four main stages: (i) document acquisition and preprocessing; (ii) finding extraction, and alignnment of original finding and cited finding ; and (iii) rhetorical function classification.

4.2 Document acquisition and preprocessing

To construct a plant-health citation corpus, we first identified articles on the transmission of plant pathogens by insect vectors in the *Web of Science* (WoS) database. The query targeted English-language research articles and combined plant-related terms, insect-related terms derived from CABI arthropod datasheets, and vector-transmission terms, while excluding terms related to human health. Articles without a DOI were discarded to facilitate full-text retrieval. The resulting reference collection is referred to as the *Query Article Set*. The metadata of the citing-articles were then retrieved primarily from *OpenAlex*, selected for its broad coverage, and enriched with self-citation information through *COCI* API and duplicates identified. During this process, we applied a set of filters to reduce errors caused by bibliographic metadata inconsistencies. The citing-article links were then consolidated from full-texts.

Full-text documents were then collected from multiple sources, including *PMC*, *ISTEX*, *Crossref*, *Unpaywall*, *ScienceDirect*, *Wiley Online Library*, and the *PLOS* Open Dataset. PDF full text documents were processed with *GROBID* (v0.8.2) (gro) to extract document structure, bibliographic references, and in-text citation markers. We also used *ScienceDirect*, and *PLOS* citation information. Finally, sentence segmentation was performed to enable consistent extraction of citation contexts.

To associate extracted citation context with the correct cited articles, we matched bibliographic references in the full text to the citing-cited DOI pairs recorded in the citation metadata. This matching relied on article metadata, including DOI, author names, publication year, and title, and was complemented by *CrossRef* queries when exact matching was not possible. Only articles from the *Query Article Set* with at least one citation were indexed in the graph. *Article*, *Section*, and *In-context Citation* nodes were instantiated. *Article* nodes store bibliographic metadata, while each *In-context Citation* node stores the sentence containing the in-text citation marker and a citation window consisting of the three preceding and three following sentences. At this stage, the graph captures the document structure and the bibliographic citation network. Appendix B displays the distribution of the citations.

4.3 Relation extraction and finding identification

Findings are automatically extracted from cited article abstracts and citation contexts. We used the hard prompting approach of (Nédellec et al., 2026) with the same prompt as it achieved the highest F_1-score (72%) among the methods evaluated by the authors on the EPOP plant health corpus. Following Nédellec et al. (2026), we selected Qwen3-32B for the LLM-based experiments because it was among the best-performing models, alongside DeepSeek-V3, while being substantially less computationally demanding.

To distinguish the original findings from background knowledge in the cited article abstract, candidate findings are classified by a Qwen3-32B LLM. These findings are instantiated as *Finding* nodes storing their normalized relational representations. For each citation context–finding pair, a *Citation Event* node is then instantiated and linked to the corresponding *Finding* and *In-Context Citation* nodes. The findings extracted from the citation context are stored in the corresponding *In-context Citation* node.

The cited findings are compared with the associated original finding to determine whether the citation context refers to that finding. A positive match is assigned when the relation labels are identical and the normalized arguments are semantically

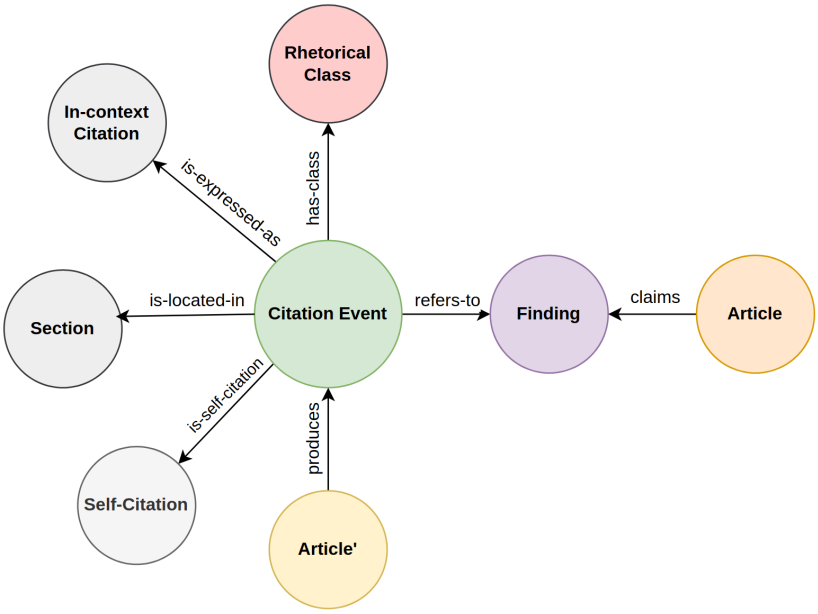


Figure 1: The Rhetorical Citation Graph schema

compatible according to the hierarchical resources used for normalization. Depending on the outcome, the alignment attribute of the *Citation Event* node is set to "yes" or "no". For the plant health application we adopted the same relational schema as the EPOP corpus. Relation labels are *transmits* (between insect and pathogen), and *hosts* (between insect and plant). Relation arguments are species normalized to NCBI taxonomy classes that considered compatible if one belongs to the taxonomic lineage of the other. For example, (Bin et al., 2026) cites (Zhang et al., 2019). Abstract of (Zhang et al., 2019): *This is the first report of the ability of D. citri to transmit CYVCV from infected to healthy citrus under laboratory conditions.* Citation by (Bin et al., 2026): *Additionally, laboratory studies have shown that CYVCV can also be transmitted by insect vectors, including Aphis spiraecola, A. craccivora, and Dialeurodes citri (Onelge et al. 2010; Zhang et al. 2019a). However, the mechanisms underlying insect-mediated transmission of CYVCV under natural field conditions remain unclear.* The abstract of Zhang et al. (2019) yields the finding *D. citri (Dialeurodes citri) - transmits - CYVCV (Citrus yellow vein clearing virus)*, where the reference names of the vector insect and the transmitted virus according to the NCBI taxonomy are given in parentheses. This finding is represented by a single *Finding* node. The citation context in Bin et al. (2026) yields three cited findings involving *Aphis spiraecola*, *A. craccivora*, and *Dialeurodes citri*. They are stored in the *In-context Citation* node. During alignment, the relation involving *Dialeurodes citri* is identified as matching the original finding. The alignment attribute of the Citation Event Node is then set to *yes*. The two other cited findings are aligned with different findings extracted from the cited article through their respective Citation Event nodes.

4.4 Rhetorical citation function classification

To assign rhetorical functions to citations and populate the corresponding *Rhetorical Class* nodes, we used successively two language models: a coarse-grain classifier followed by a refinement model designed to distinguish comparison from contrast, and method from result.

Six-class rhetorical citation function classification

In the absence of an annotated corpus in the plant health domain, we adopted a transfer learning strategy. We first fine-tuned language models on rhetorical function datasets from the NLP domain, where such resources are well established, and then applied them to plant health. To evaluate the effectiveness of this transfer, we manually annotated a new corpus, PD100Cit, comprising 100 citation contexts centered on Pear Decline, a disease affecting pear trees. For training, we selected the citation function corpus of Jurgens et al. (2018) because of the high quality of annotation and the relevance of the six-class rhetorical

annotation scheme: *Background*, *Motivation*, *Uses*, *Extends*, *Comparison/Contrast*, and *Future*.

We compared five BERT-based transformer models. The input consists of the citance together with its surrounding context sentences, truncated to a maximum of 512 tokens. Citation markers are replaced by a special token, `CITSEG`, whose contextual representation is extracted by the pretrained language model. The contextual representation of the `CITSEG` token is used as a compact representation of the citation, under the hypothesis that it captures the argumentative cues expressed in its local discourse context. The `CITSEG` embedding is then passed to a multi-layer perceptron consisting of a single hidden layer of size 768 with a non-linear activation, followed by a softmax output layer for rhetorical function classification.

We experimented under different experimental settings to evaluate the impact of the size of citation context on rhetorical function classification. Citation context was evaluated using three context-window sizes: 1–1, 2–2, and 3–3, where an N–N window includes the citance together with the N preceding and N following sentences.

Because the rhetorical classes are highly imbalanced with *Background* as the dominant class, we also evaluated two data augmentation strategies. First, we incorporated citation instances from Teufel et al. (2006), using the class mapping proposed by Jurgens et al. (2018). Second, to increase the representation of the *Comparison or Contrast* class, we added citation instances from Cohan et al. (2019) labeled as *Comparing Results*. Training hyperparameters and complete evaluation results are detailed in Appendix A. On the Jurgens et al. (2018) dataset the best configuration used a 3–3 context window with BioBERT, reaching a macro-F1 score of 76.14%. Data augmentation further improved performance yielding a best macro-F1 score of 77.81% with PubMedBERT.

The resulting performance is comparable to the best transformer-based models reported in the literature, although it remains slightly below the graph-enhanced HGFN model of Quan et al. (2025).

To assess the transferability of the BERT-based classifiers, we evaluated them on the PD100Cit dataset. The best performances were obtained with BioBERT using a 3–3 context window, which achieved a macro-F1 score of 58.89% and a weighted-F1 score of 83.66%. Increasing the context window consistently improved performance across all models. The performance gap between the source and target datasets suggests that the task remains challenging in the plant health domain. Performance was dominated by the majority *Background* class (F_1 = 89.20%), whereas minority classes were classified less accurately. The *Uses* class achieved the second-highest F_1 score (66.20%), while the *Comparison/Contrast* class exhibited high recall (87.50%) but low precision (36.20%), indicating a tendency to overpredict this class. The *Motivation* class proved the most difficult to identify (F_1 = 29.20%). The *Extends* and *Future* classes were absent from the PD100Cit test set and were therefore excluded from the class-level evaluation.

Prompting-based refinement of comparative and contrastive citations

The *Comparison/Contrast* class, as defined by Jurgens et al. (2018), encompasses citation events expressing both agreement and disagreement, as well as comparisons of methods and results. To better support the analysis of consensus and disagreement, we further refined the citation events assigned to this class using prompt-based classification with Qwen3-32B. This targeted approach enables finer-grained classification from only a few examples without requiring reannotation of the original training corpus.

Two prompt-based classification tasks were defined: the first distinguishes method- from result-based comparisons, and the second, applied only to result-based citations, distinguishes agreement (*Compare*) from disagreement, divergence or contradiction (*Contrast*). Each prompt included a task definition, a description of the expected input and output format, and two examples per class.

The first prompt was evaluated using the original Teufel et al. (2006) corpus. Since this distinction is not explicitly annotated, we derived a binary evaluation set from the original annotation by mapping *Contrast/Comparison in Goals or Methods* (*CoCoGM*) to *Method* and *Contrast/Comparison in Results* neutral (*CoCoR0*),and negative *CoRes-*, *CoCo-* to *Result*. This yielded an evaluation set of 104 method-based and 42 result-based citation events. The model achieved a macro-F1 score of 87.19% and a weighted-F1 score of 89.61%, as reported in Table 1. Performance was higher for the *Method*

class (F1 = 92.89%) than for the *Result* class (F1 = 81.48%).

The second prompt, which distinguishes result-based comparisons from contrasts, was evaluated using the Cohan et al. (2019) corpus. Citations labeled as *comparing results* were assigned to *Compare* or *Contrast* according to their secondary *supportive/not supportive* annotation. To increase the number of contrast examples, we additionally incorporated Teufel et al. (2006) citations labeled *CoRes-*, corresponding to cases in which the citing article reports lower results than the cited work. The resulting evaluation set comprised 289 *Compare* and 89 *Contrast* citation events.

Qwen3-32B achieved a macro-F1 score of 81.58% and a weighted-F1 score of 86.41% on this task, as shown in Table 2. The *Compare* class was identified more accurately (F1 = 90.71%) than the *Contrast* class (F1 = 72.45%), reflecting both the smaller number of contrast examples and the greater difficulty of identifying disagreement in citation contexts.

Table 1: Performance of Qwen3-32B on *Method /Result* classification.

Class	Precision	Recall	F1-score	Support
Method	0.916	0.942	0.929	104
Result	0.846	0.786	0.815	42
Macro average	0.881	0.864	0.872	146
Weighted average	0.896	0.897	0.896	146

Table 2: Qwen3-32B performance on *Compare /Contrast* classification.

Class	Precision	Recall	F1-score	Support
Compare	0.958	0.862	0.907	289
Contrast	0.664	0.798	0.725	89
Macro average	0.811	0.830	0.816	378
Weighted average	0.888	0.847	0.864	378

These results indicate that prompt-based classification provides an effective means of refining the *Comparison/Contrast* class into more informative rhetorical categories. The best configuration was applied to the citation contexts of the PLIN graph and the predicted rhetorical classes were used to populate the corresponding *Rhetorical Class* nodes.

Table 3: Corpus statistics.

Collection	Count
WoS primary references	8,181
Query Article Set (with DOI)	7,318
Citing references	94,042
Union	97,800
Full texts retrieved	62,878

5 Plant Literature Reliability Network

Together, these components define a generic and modular pipeline for constructing graphs conforming to the Rhetorical Citation Graph (RCG) model. Each component was selected based on reported or experimentally validated performance and can be replaced as improved methods become available.

5.1 PLIN Graph Description

Table 3 shows the corpus statistics for the article collection retrieved by 5 August 2025. The dataset exhibited an average of 31 citations per article, with a standard deviation of 48.59. The maximum number of citations for a single article was 1,208, while the minimum was 1. A summary of the counts per node type is presented in Table 4.

Table 4: Numbers of nodes and edges in the graph.

Graph element	Instances
Articles (primary and citing)	87,974
Original findings	18,854
Citation events (CEs)	262,807
Characterized CEs	169,925

Note: Characterized CEs are citation events linked to a rhetorical class, an in-context citation, a section, and a self-citation node.

Among all citation events, 169,925 were characterized with rhetorical class and contextual information. 92,882 citing-cited article pairs could not be characterized, either because the citing article was not successfully retrieved or could not be converted to a structured format due to missing OCR, accounting for 78.52% of characterization failures, or because the association between a cited DOI and its corresponding in-text citation failed, accounting for 21.48% of characterization failures. To establish this association, metadata for the cited article recorded in the graph (first-author name, publication year, and title) were compared with the bibliography entries in the citing article. When

this comparison produced no match, Crossref was queried using the complete bibliographic reference. The association was accepted only when the DOI returned by Crossref matched the DOI recorded in the graph.

The rhetorical citation function classification was applied on the PLIN citations using BioBERT with the 3–3 context window. The distribution after classification and refinement of the *Comparison or Contrast* class is reported in Table 5. *Contrast – contradictory results* denotes cases in which the results of the citing study explicitly contradict those reported in the cited study for the same study object. It is distinguished from *Contrast results*, which refers to broader differences or discrepancies between the results of the citing and cited studies that do not constitute a direct contradiction. For example, *"The transmission efficiency of M. hiroglyphicus (55%) is higher than that of Y. flavovittatus (45%) (CITSEG)"* is classified as *Contrast results* while *"Our observations contradict the conclusions of (CITSEG)."* is classified *Contrast – contradictory results*.

Table 5: Distribution of predicted rhetorical classes across citation events (CE)

Rhetorical Class	CE number
Background	124,401
Compare results	12,043
Usage	10,009
CoCoGM	8,865
Contrast results	8,651
Extends	2,623
Motivation	1,428
Contrast – contradictory results	968
Future	937
Total	169,925

6 Supporting EPPO Base Curation

The EPPO Global Database (EPPO, 2026) contains curated structured information about pathogen transmission to plants. Curating these interactions requires extensive examination of the scientific literature. We investigated whether the PLIN graph could facilitate this process by providing rapid access to the evidence supporting or questioning a reported interaction.

As a case study, we considered interactions that EPPO had annotated as uncertain, namely insect–pathogen relations labeled "Potential vector" and insect–plant relations labeled "Doubtful host", representing 129 interactions. These interactions correspond to cases where the available literature was considered insufficient to establish the vector role with certainty. For these interactions, we identified 181 original articles in the PLIN graph whose abstracts mentioned both the source and target entities of the corresponding interaction. To keep the manual analysis tractable, we retained only articles associated with more than five citation events, yielding 134 articles with citation contexts. For each, we calculate the average distribution of rhetorical functions across all citation events in order to assess whether citation behaviour could provide evidence of scientific uncertainty or disagreement. As shown in Table 6, the debate-related rhetorical classes (Contrast – contradictory results, Contrast results, Compare results, and CoCoGM) account for only a small proportion of citation events on average (0.34%, 5.30%, 7.24%, and 5.28%, respectively), indicating that the EPPO uncertainty labels generally do not arise from conflicting claims in the literature.

Table 6: Mean percentage of citation events assigned to each rhetorical class across the 134 selected articles.

Rhetorical class	Mean (%)
Background	73.87
Motivation	0.68
Uses	4.97
Extends	1.45
Future	0.87
Contrast – contradictory results	0.34
CoCoGM	5.28
Compare results	7.24
Contrast results	5.30

Manual examination of the original articles showed that the EPPO annotations reflected a lack of conclusive evidence for disease transmission rather than conflicting claims in the literature. The low average prevalence of debate-related rhetorical functions was consistent with this interpretation, although citation behaviour varied substantially among articles.

This case study constitutes a proof of concept, illustrating the potential of the PLIN graph to support knowledge base curation by providing rapid access to normalized findings together with the citation contexts in which they are discussed. This structured representation enables to trace the evidence underlying reported interactions and to identify the specific biological relationships addressed in the literature. Furthermore,

debate-related rhetorical functions could provide a useful means of prioritising findings for manual review in cases where the literature contains explicit scientific disagreement.

7 Conclusion

In this work, we introduced the Rhetorical Citation Graph (RCG), a graph model that represents scholarly citations as fine-grained citation events linking citation contexts to individual scientific findings together with their rhetorical function. We instantiated this model by constructing PLIN, a graph of the plant health literature that integrates articles, normalized findings, citation contexts, citation events, and rhetorical classifications.

By moving beyond document-level citation networks, the proposed framework enables the analysis of how individual findings are cited, interpreted, and discussed across the scientific literature. Although illustrated in the plant health domain, the model is generic and can be applied to other scientific fields. The current PLIN graph corresponds to a partial instantiation of the RCG model centered on the Query Article Set. As additional collections are incorporated, previously distinct subgraphs can be merged through shared article nodes, progressively yielding a more comprehensive representation of the citation network. The proposed framework could also support the development of scientific fact-checking systems in experimental fields (Labbé et al., 2019), as well as scientific claim verification tasks (Wadden et al., 2020) by enabling the structured analysis of how findings are cited, interpreted, and evaluated through citation context and rhetorical functions.

Future work will focus on improving the automatic identification of findings and how they are framed as scientific claims within texts, for example using recent finding identification and generation methods (Liang et al., 2024). This would strengthen the alignment of findings across publications and further enhance applications such as scientific claim verification and expert knowledge curation. More broadly, the proposed framework provides a computational foundation for studying the epistemic dynamics of scientific knowledge. Beyond characterizing the rhetorical profile of individual findings, it opens the way to longitudinal analyses of how findings are interpreted, debated, corroborated, and progressively established through scientific communication.

8 Limitations

PLIN graph was constructed from English-language documents. Extending PLIN to multilingual corpora would enable a more comprehensive and inclusive analysis of citation dynamics across scientific communities.

Our approach does not explicitly associate extracted relations with individual references within a citation context. Consequently, residual attribution ambiguities may arise when multiple cited works report semantically equivalent findings. Since the objective of the pipeline is to characterize citation behaviour around normalized findings rather than to reconstruct exact citation attributions, this ambiguity has only a limited impact on the analyses presented here. Finally, the rhetorical classification assigns a single label to each citation passage whereas prior work suggests that citation intent can be multi-faceted, with multiple rhetorical functions potentially co-occurring within the same citation context (Zhang et al., 2025; Lauscher et al., 2022). The RCE schema is extensible to multi-label annotation: a *Citation Event* could be linked to multiple *Rhetorical Class* nodes. Implementing this extension would require adapting the classification, and evaluation but would not require redesign of the graph structure.

Acknowledgments

We are grateful to the INRAE MIGALE bioinformatics facility (MIGALE, INRAE, 2020. Migale bioinformatics Facility, doi: 10.15454/1.5572390655343293E12) for providing help and/or computing and/or storage resources. This work was initiated through collaborations established within the BEYOND project (ANR grant no. 20-PCPA-0002). Anne-Sophie Foussat is currently undertaking a PhD funded by the INRAE Directorate General.

References

GROBID. https://github.com/grobidOrg/grobid.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural*

Language Processing (EMNLP-IJCNLP 2019), pages 3615–3620. Association for Computational Linguistics.

Dan Berrebbi, Nicolas Huynh, and Oana Balalau. 2022. GraphCite: Citation Intent Classification in Scientific Publications via Graph Embeddings. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion)*, page 779–783.

Yu Bin, Liqing Jiang, Aobo Zhuang, Muzi Li, Junjun Li, Zhen Song, Changyong Zhou, Taiyun Wei, and You Li. 2026. Semipersistent transmission of citrus yellow vein clearing virus by asian citrus psyllid: Insights from field surveys and infectious clone-based studies. *Phytopathology Research*, 8(1):32.

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, pages 3586–3596, Minneapolis, Minnesota. Association for Computational Linguistics.

Arman Cohan and Nazli Goharian. 2017. Contextualizing citations for scientific summarization using word embeddings and domain knowledge. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, page 1133–1136. Association for Computing Machinery.

EPPO. 2026. EPPO Global Database.

Revanth Gangi Reddy, Sai Chetan Chinthakindi, Zhenhailong Wang, Yi Fung, Kathryn Conger, Ahmed ELsayed, Martha Palmer, Preslav Nakov, Eduard Hovy, Kevin Small, and Heng Ji. 2022. NewsClaims: A new benchmark for claim detection from news with attribute knowledge. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*, pages 6002–6018, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mark Garzone and Robert E. Mercer. 2000. Towards an Automated Citation Classifier. In *Advances in Artificial Intelligence: Canadian AI 2000*, pages 337–346. Springer Berlin Heidelberg.

Xiaorui Jiang and Jingqiang Chen. 2023. Contextualised segment-wise citation function classification. *Scientometrics*, 128:5117–5158.

David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. 2018. Measuring the Evolution of a Scientific Field through Citation Frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

Priyanka Kargupta, Ishika Agarwal, Tal August, and Jiawei Han. 2025. Tree-of-Debate: Multi-Persona Debate Trees Elicit Critical Thinking for Scientific Comparative Analysis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, pages 29378–29403. Association for Computational Linguistics.

Paris Koloveas, Serafeim Chatzopoulos, Thanasis Vergoulis, and Christos Tryfonopoulos. 2025. Can LLMs Predict Citation Intent? An Experimental Analysis of In-Context Learning and Fine-Tuning on Open LLMs. In *Linking Theory and Practice of Digital Libraries: 29th International Conference on Theory and Practice of Digital Libraries (TPDL 2025)*, page 207–224.

Suchetha N. Kunnath, David Pride, and Petr Knoth. 2023. Prompting Strategies for Citation Classification. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*, pages 1127–1137. Association for Computing Machinery.

Cyril Labbé, Natalie Grima, Thierry Gautier, Bertrand Favier, and Jennifer A. Byrne. 2019. Semi-automated fact-checking of nucleotide sequence reagents in biomedical research publications: The Seek & Blastn tool. *PLoS ONE*.

Anne Lauscher, Brandon Ko, Bailey Kuehl, Sophie Johnson, Arman Cohan, David Jurgens, and Kyle Lo. 2022. MultiCite: Modeling realistic citations requires moving beyond the single-sentence single-label setting. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2022)*, pages 1875–1889. Association for Computational Linguistics.

Yuan Liang, Massimo Poesio, and Roonak Rezvani. 2024. A Fine-grained citation graph for biomedical academic papers: the finding-citation graph. In *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing (BioNLP 2024)*, pages 416–426. Association for Computational Linguistics.

Fan Luo, Marco A. Valenzuela-Escárcega, Gus Hahn-Powell, and Mihai Surdeanu. 2018. Scientific Discovery as Link Prediction in Influence and Citation Graphs. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs@NAACL-HLT-2018)*, pages 1–6.

Boaz Miller. 2013. When is consensus knowledge based? Distinguishing shared knowledge from mere agreement. *Synthese*, 190:1293–1316.

Michael J. Moravcsik and Poovanalingam Murugesan. 1975. Some Results on the Function and Quality of Citations. *Social Studies of Science*, 5(1):86–92.

Silas Boye Nissen, Tali Magidson, Kevin Gross, and Carl T. Bergstrom. 2016. Research: Publication bias and the canonization of false facts. *eLife*, 5.

Claire Nédellec, Marine Courtin, Xinzhi Yao, Marie Grosdidier, Isabelle Pieretti, Sandy Duperier, and Robert Bossy. 2026. Epop: A benchmark corpus for assessing nlp models on structured information extraction in plant health. In *Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026)*, pages 1331–1340, Palma, Mallorca, Spain. European Language Resources Association (ELRA).

Aniket Pramanick, Yufang Hou, Saif M. Mohammad, and Iryna Gurevych. 2026. Claimflow: Tracing the evolution of scientific claims in nlp. *Preprint*, arXiv:2603.16073.

Zhibang Quan, Jin Mao, and Gang Li. 2025. Multi-intent prediction of scientific literature based on heterogeneous graph fusion network. *Scientometrics*.

Uri Shwed and Peter S. Bearman. 2010. The Temporal Structure of Scientific Consensus Formation. *American Sociological Review*, 75(6):817–840.

Neşet Tan, Niket Tandon, Oyvind Tafjord, Michael Witbrock, Peter Clark, and Mark Gahegan. 2026. Echoes of citations: Automated extraction of claims from full scientific papers. *Proceedings of the AAAI Symposium Series*, 8:568–576.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic Classification of Citation Function. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP '06)*, pages 103–110. Association for Computational Linguistics.

Hong-Jin Tsai, An-Zi Yen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2023. Citation Intent Classification and Its Supporting Evidence Extraction for Citation Graph Construction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23)*, page 2472–2481. Association for Computing Machinery.

Vijay Viswanathan, Graham Neubig, and Pengfei Liu. 2021. CitationIE: Leveraging the citation graph for scientific information extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*, pages 719–731. Association for Computational Linguistics.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*, pages 7534–7550. Association for Computational Linguistics.

Xiujuan Xu, Yueyue Xie, Xiaowei Zhao, and Yu Liu. 2025. MF-Cite: Citation intent classification in scientific papers based on multi-feature fusion. *Scientometrics*, 130(8):4465–4493.

Erjia Yan and Ying Ding. 2012. Scholarly network similarities: How bibliographic coupling networks, citation networks, cocitation networks, topical networks, coauthorship networks, and coword networks relate to each other. *Journal of the American Society for Information Science and Technology*, 63(7):1313–1326.

Yan-Hua Zhang, Chun-Hua Liu, Qian Wang, Yan-Ling Wang, Chang-Yong Zhou, and Yan Zhou. 2019. Identification of Dialeurodes citri as a vector of Citrus yellow vein clearing virus in china. *Plant Disease*, 103(1):65–68.

Yang Zhang, Yufei Wang, Quan Z. Sheng, Lina Yao, Haihua Chen, Kai Wang, Adnan Mahmood, Wei Emma Zhang, Munazza Zaib, Subhash Sagar, and Rongying Zhao. 2025. Deep learning meets bibliometrics: A survey of citation function classification. *Journal of Informetrics*, 19(1):101608.

Yang Zhang, Rongying Zhao, Yufei Wang, Haihua Chen, Adnan Mahmood, Munazza Zaib, Wei Emma Zhang, and Quan Z. Sheng. 2022. Towards Employing Native Information in Citation Function Classification. *Scientometrics*, 127(11):6557–6577.

Appendix A. Rhetorical classification training and scores

Models were implemented in PyTorch 1.10.2+cu102 and trained using the following hyperparameters: batch size of 32, 20 epochs, AdamW optimiser, learning rate of 2e-5, weight decay of 0.002, dropout of 0.5, and warmup ratio of 0.2. Experiments were repeated using four random seeds: 42, 5171, 798, and 1965. The dataset was split into 65% training, 15% validation, and 20% test data.

The average macro-F1 and weighted-F1 scores are reported in Table 7. Adding section information as a 32-dimensional embedding concatenated to the CITSEG embedding did not consistently improve performance. Larger section embeddings of 256 dimensions, and positional section embeddings were evaluated only on the ACL-ARC dataset, but they produced limited variation across models, as shown in Table 7, which reports experiments conducted using ACL-ARC data only.

The shortest context window, 1-1, yielded the lowest scores. Therefore, in the augmented-data experiments using additional examples from Teufel et al. (2006) and Cohan et al. (2019), we retained only the 3-3 and 2-2 context windows, with either no section embedding or a 32-dimensional section embedding. Results for the five evaluated models are reported in Table 8.

Table 7: Macro-F1 scores for six-class rhetorical citation function classification on the Jurgens dataset. Results are shown for different context-window sizes and section representations without additional training data. Bold values indicate the best score for each model, and the best mean score across models.

Context	Section representation	SciBERT	BioBERT	BioLinkBERT	PubMedBERT	RoBERTa	Mean
3–3	256	**0.7597**	0.7398	0.7293	0.7213	0.7458	0.7392
3–3	32	0.7424	0.7351	0.7453	0.7370	0.7178	0.7355
3–3	Position	0.7163	**0.7614**	0.7333	0.7466	0.7403	0.7396
3–3	None	0.7481	0.7501	**0.7605**	0.7411	0.7543	**0.7508**
2–2	256	0.7513	0.7505	0.7402	0.7431	**0.7592**	0.7488
2–2	32	0.7356	0.7442	0.7440	0.7304	0.7558	0.7420
2–2	Position	0.7374	0.7400	0.7379	0.7267	0.7373	0.7359
2–2	None	0.7532	0.7221	0.7509	0.7420	0.7394	0.7415
1–1	256	0.7519	0.7064	0.7179	0.7276	0.7543	0.7316
1–1	32	0.7369	0.7333	0.7314	0.7139	0.7320	0.7295
1–1	Position	0.7448	0.6991	0.7203	**0.7557**	0.7392	0.7318
1–1	None	0.7474	0.7237	0.7388	0.7411	0.7274	0.7357

Table 8: Macro-F_1 scores for six-class rhetorical citation function classification using additional Teufel and Cohan training data. Results are shown according to citation-context size, section representation, and language model.

Context	Section representation	Model	Macro-F1
3–3	None	SciBERT	0.7518
3–3	None	BioBERT	0.7474
3–3	None	BioLinkBERT	0.7365
3–3	None	PubMedBERT	0.7733
3–3	None	RoBERTa	0.747440
3–3	32	SciBERT	0.7471
3–3	32	BioBERT	0.7537
3–3	32	BioLinkBERT	0.7436
3–3	32	PubMedBERT	0.7782
3–3	32	RoBERTa	0.753724
2–2	None	SciBERT	0.7454
2–2	None	BioBERT	0.7529
2–2	None	BioLinkBERT	0.7356
2–2	None	PubMedBERT	0.7321
2–2	None	RoBERTa	0.7682
2–2	32	SciBERT	0.7544
2–2	32	BioBERT	0.7464
2–2	32	BioLinkBERT	0.7467
2–2	32	PubMedBERT	0.7419
2–2	32	RoBERTa	0.7709

Table 9: Average macro-F_1 and weighted-F_1 scores across seeds for each model and context window on PD100Cit.

Model	2–2			3–3		
	N	Macro	Weighted	N	Macro	Weighted
BioBERT	4	0.562	0.794	4	**0.589**	**0.837**
BioLinkBERT	4	0.519	0.812	4	0.560	0.823
PubMedBERT	4	0.492	0.735	4	0.512	0.786
RoBERTa	4	0.536	0.780	4	0.560	0.804
SciBERT	3	0.470	0.787	4	0.494	0.796

Per-class performance on ACL-ARC augmented are reported in Table 11. Scores on PD100cit are reported Tables 9 and 10.

Table 10: Average precision, recall, and F_1 scores across four seeds for each class using BioBERT with a 3–3 context window on PD100Cit.

Class	Support	P	R	F1
Background	83	0.940	0.849	0.892
Compare/contrast	6	0.362	0.875	0.509
Motivation	2	0.375	0.250	0.292
Uses	9	0.742	0.611	0.662

Table 11: Per-class precision (P), recall (R), and F_1 scores obtained using a 3–3 context window without section information on ACL-ARC after data augmentation. Results are reported as mean ± standard deviation across four seeds.

	BioBERT			BioLinkBERT			PubMedBERT			RoBERTa			SciBERT		
Class	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Background	0.833± 0.037	0.898± 0.023	0.864± 0.019	0.843± 0.008	0.884± 0.032	0.863± 0.015	0.856± 0.036	0.905± 0.024	0.879± 0.018	0.847± 0.025	0.881± 0.021	0.863± 0.015	0.847± 0.034	0.895± 0.020	0.870± 0.018
Compare/contrast	0.817± 0.034	0.829± 0.031	0.822± 0.005	0.814± 0.043	0.823± 0.025	0.818± 0.023	0.832± 0.064	0.824± 0.037	0.826± 0.016	0.812± 0.058	0.820± 0.049	0.815± 0.035	0.805± 0.071	0.829± 0.054	0.814± 0.027
Extends	0.644± 0.053	0.532± 0.055	0.579± 0.019	0.540± 0.084	0.590± 0.072	0.557± 0.034	0.618± 0.095	0.586± 0.090	0.593± 0.052	0.547± 0.070	0.625± 0.088	0.581± 0.059	0.630± 0.073	0.554± 0.103	0.580± 0.042
Future	0.796± 0.151	0.575± 0.066	0.660± 0.053	0.729± 0.146	0.536± 0.119	0.599± 0.050	0.834± 0.050	0.725± 0.034	0.774± 0.018	0.765± 0.131	0.612± 0.105	0.676± 0.096	0.719± 0.150	0.712± 0.112	0.708± 0.096
Motivation	0.859± 0.076	0.694± 0.133	0.756± 0.061	0.854± 0.065	0.680± 0.075	0.754± 0.043	0.786± 0.104	0.683± 0.036	0.727± 0.041	0.820± 0.089	0.654± 0.091	0.724± 0.075	0.821± 0.078	0.652± 0.078	0.722± 0.038
Uses	0.840± 0.051	0.823± 0.041	0.829± 0.013	0.844± 0.032	0.815± 0.020	0.828± 0.005	0.849± 0.025	0.833± 0.028	0.840± 0.008	0.839± 0.017	0.814± 0.016	0.826± 0.011	0.845± 0.039	0.813± 0.052	0.827± 0.023

Appendix B. Citations per article

Figure 2 displays the distribution of citations per article in the Plant Health Query Article Set.

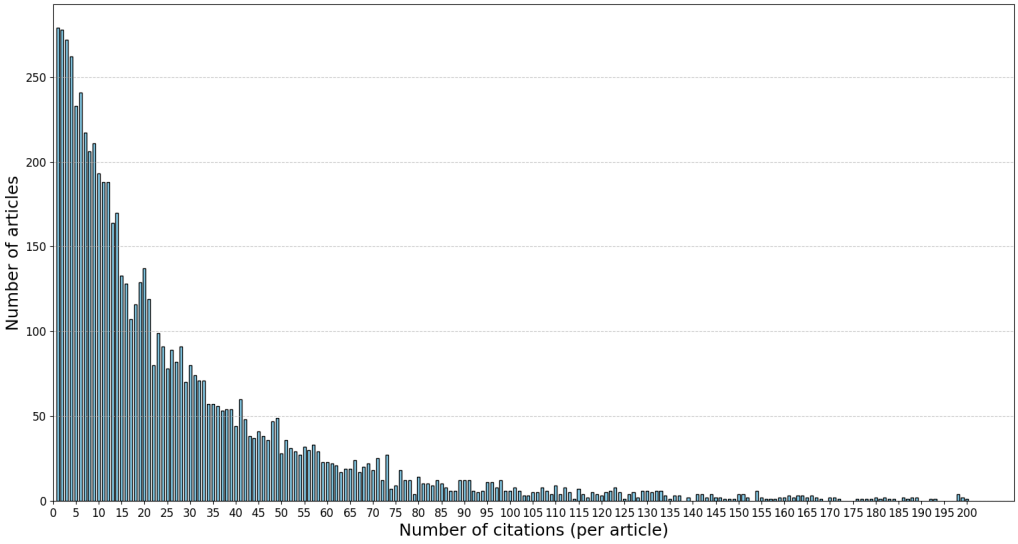


Figure 2: Distribution of citations per article in the Query Article Set (up to 200 citations per article).

C Prompting Qwen3-32B for six-class citation-function classification

Tables 12 and 13 report the performance of Qwen3-32B with thinking disabled and enabled, respectively.

Table 12: Per-class performance of Qwen3-32B on the PD100Cit dataset with thinking disabled.

Class	Precision	Recall	F1-score	Support
Background	0.9000	0.6506	0.7552	83
Motivation	0.2500	1.0000	0.4000	2
Uses	0.4167	0.5556	0.4762	9
Compare or contrast	0.1000	0.3333	0.1538	6
Macro average	0.4167	0.6349	0.4463	100
Weighted average	0.7955	0.6300	0.6869	100

Table 13: Per-class performance of Qwen3-32B on the PD100Cit dataset with thinking enabled.

Class	Precision	Recall	F1-score	Support
Background	0.9355	0.6988	0.8000	83
Motivation	0.2222	1.0000	0.3636	2
Uses	0.4615	0.6667	0.5455	9
Compare or contrast	0.2143	0.5000	0.3000	6
Macro average	0.4584	0.7164	0.5023	100
Weighted average	0.8353	0.6900	0.7384	100

Schema-Guided Extraction of Interaction Knowledge Bases from Narrative Text: A Folktale Case Study

Sergey Kovalchuk[1,2], Anastasia Ugleva[3], Oleg Kubryak[1,4]

[1]Sirius University of Science and Technology, Sirius Federal Territory, Russia,
[2]ITMO University, Saint Petersburg, Russia, [3]Higher School of Economics, Moscow, Russia,
[4]National Research University "Moscow Power Engineering Institute", Moscow, Russia
Correspondence: kovalchuk@itmo.ru

Abstract

We present MAGIC, a six-dimensional framework for extracting structured knowledge bases of hero and magic entity interactions from narrative texts of folktales and fairytales: each interaction between a story's protagonist and a magical helper, villain, or object is decomposed along trigger modality, actor profile and agency balance, interaction mechanics, outcome valence, narrative function, and narrative position. We build an LLM-based extraction pipeline and apply it to a corpus of 1306 folktales collected from around the world, then show the schema supports rich structural analysis: unsupervised clustering over the extracted features recovers five recurring, readily nameable interaction archetypes – invoked and unsolicited helpers, a morally tested gatekeeper, a hero-controlled magic object, and an adversarial obstacle – that are stable across the three full-corpus extraction models (0.95 mean cluster-centroid similarity) and cut across both tale type and regional tradition rather than reproducing existing typological catalogues. We further characterize the schema's reliability under LLM extraction (inter-model, not against a human gold standard): parse success and schema-conformance rates, a decomposition of extraction drift into field-confusion versus vocabulary-incompleteness failure modes, and cross-model field-level agreement, which varies substantially across the six dimensions and is sensitive to how many tales are compared across models. Despite this field-level variability, the archetype-level structure remains stable, suggesting that knowledge bases built from LLM extraction should be evaluated at the level of the downstream structure they support.

1 Introduction

Building a knowledge base from unstructured text requires a schema precise enough to be extracted consistently, yet expressive enough to capture what matters about the source material. This tension is easier to manage when the target schema stays close to the surface form of the text (entities, relations, dates) – not because such extraction is free of interpretive judgment, but because its judgments are usually anchored to an explicit textual mention – and harder when the schema requires inference with no such anchor: who holds power in an exchange, or what narrative role an event plays, rather than simply what was stated. Recent work on LLM-driven knowledge graph construction often faces this tension. For example, Shrimal et al. (2025) show that static, hand-specified extraction schemas produce unreliable output and propose optimizing the schema itself against extraction performance, with Han and Lai (2026) arguing that symbolic, closed-vocabulary relation schemas lose exactly the contextual nuance an LLM could otherwise represent in natural language.

Narrative text is close to a worst case for this problem: its structure is implicit, culturally variable, and has been the subject of a century of disagreement among human narratologists about how to formalize it at all. Within narrative text, folktales and fairytales are a particularly attractive target for knowledge base construction. Transmitted orally across generations and borders, they act as a condensed carrier of cultural knowledge: recurring plots and character configurations encode moral norms and archetypal patterns of human behavior that remain remarkably stable across regions and centuries (Lüthi, 1986; Uther, 2004; da Silva and Tehrani, 2016). This stability is precisely why folktales attracted the earliest systematic formalization efforts in narratology (Propp, 1968; Greimas, 1983) mainly with expert-designed schemas for narrative structure, hand-applied decades before machine extraction was conceivable. Among the structures these formalisms identify, the encounter between a hero and a magic-bearing entity (a helper, a donor, a villain, an enchanted object) is where much of a tale's "wisdom content" is staged: it is the lo-

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 261–283
October 28, 2026 ©2026 Association for Computational Linguistics

cus of moral testing, conditional aid, asymmetric agency, and reward, and it recurs across otherwise unrelated traditions. This makes the hero–magic interaction both a natural unit for an interaction-level knowledge base and a demanding testbed for schema-guided LLM-based extraction.

We introduce MAGIC (a recursive acronym for MAGIC as Actor-Governed Interaction Cycles), a framework with a six-dimensional schema for one recurring narrative structure focused on the interaction between a story's protagonist and a non-protagonist actor or object with unusual capabilities (a magic helper, a villain, an enchanted item) that often appear in folktales and fairytales. Unlike character-centered frameworks that describe the protagonist's overall arc (Campbell, 1949) or fixed-order plot grammars (Propp, 1968), MAGIC treats each such interaction as the unit of analysis, decomposed along six orthogonal dimensions: trigger modality (D1), actor profile and agency balance (D2), interaction mechanics (D3), outcome valence (D4), narrative function (D5), and narrative position (D6). This yields an interaction-level knowledge base rather than a single story-level label, dense enough to support downstream analysis – clustering, cross-corpus comparison – that a coarser schema would not.

Why the interaction, rather than the story or the character? The established formalisms attach their labels at coarser units: ATU (Aarne–Thompson-Uther) assigns one type per tale (Uther, 2004), the monomyth one arc per hero (Campbell, 1949), and character-network corpora one edge per co-occurring pair (Schmidt et al., 2021). Each is well suited to the question it was built to answer, and none of them can express, for a given exchange, what set it in motion, what it cost, on what condition the aid was granted, and what the hero was left holding afterwards which is where a tale's normative content actually sits. A single tale stages several such exchanges with different answers to those questions, so a story-level label cannot be refined into them after the fact; and a character-level edge discards the conditionality that separates a gift from a bargain. We therefore add a layer beneath the existing typologies rather than proposing a replacement for them, and the evaluation below is correspondingly about whether that layer can be populated repeatably, not about whether it classifies tales better than ATU does.

To evaluate the schema, we extract interactions from 1306 folktales from a publicly available corpus (Kaggle, 2024) with five LLMs of varying scale and provenance (DeepSeek V3.2, Gemini 2.5 Flash, Qwen 3.6 35B, GPT-5.2, GPT-OSS-120B): what structure does the schema reveal, and how reliably can that structure be extracted? For the first, unsupervised clustering over the extracted features recovers five recurring, readily nameable interaction archetypes, stable across the three full-corpus models and cutting across both tale type and regional tradition. For the second, we measure extraction reliability (parse and schema-conformance rates) and cross-model field-level agreement (Fleiss' κ (Fleiss, 1971)) in two conditions (all five models on a shared 50-tale subset; three full-corpus models on all 1306 tales in the corpus). The central empirical finding connects the two: the archetype-level structure is highly stable even though no individual dimension is reliably extracted in isolation, suggesting individual field values may be less trustworthy than the aggregate structure they compose – a distinction current single-model evaluations do not surface.

Our contributions are: (1) the MAGIC schema and a reproducible, multi-model extraction pipeline that produces schema-conformant interaction knowledge bases from narrative text at corpus scale; (2) an unsupervised taxonomy of hero–magic interaction archetypes, shown to be stable across extraction models and orthogonal to existing tale-type and regional classifications despite weak field-level agreement; and (3) a reliability and cross-model agreement evaluation, run at two sample sizes, that identifies which parts of a narrative-interaction schema are robustly extractable and which are not, including a decomposition of schema drift into *field confusion* versus *vocabulary-incompleteness* failure modes, and a demonstration that fine-grained field-reliability rankings from a few dozen shared items are considerably less stable than the coarser conclusions they are usually read alongside.

2 The MAGIC Schema

The MAGIC framework analyzes each interaction between a story's hero and a magic-bearing entity – a helper, villain, or object – along six orthogonal dimensions. Each dimension decomposes into multiple categorical or ordinal attributes, for 23 interaction-level attributes in total, plus two story-level fields (hero name, hero nature) that are constant across a tale's interactions.

2.1 Schema design

The six dimensions come from two sources borrowing from existing narratological descriptions and revision against pilot extractions.

Three dimensions have direct antecedents. D5 reuses Propp's function vocabulary (Propp, 1968) in condensed form, restricted to the seven functions that a hero-magic exchange can realize rather than the full inventory of 31, most of which describe plot moves with no magic-bearing counterpart. D2 takes from Greimas's actantial model (Greimas, 1983) the separation between what an actor *is* in the story world and what it *does* in a given episode; in our schema this becomes the `actor_type`/`actor_functional_role` pair. D6's story-arc values follow the departure-to-return progression common to Propp (1968) and Campbell (1949).

The other three dimensions have no direct antecedent, and for a structural reason: the unit those formalisms describe is larger than ours. Once the interaction rather than the tale is the coded object, three questions become answerable that a story-level schema cannot pose – who set this particular exchange in motion (D1), on what operational terms the magic was available (D3), and what the hero concretely gained and gave up (D4). These are the dimensions along which a helper's gift differs from a donor's bargain and from a gatekeeper's test, distinctions that a coarser encoding collapses.

Two rounds of pilot extraction then revised the vocabulary, and both revisions are visible in the version reported here. A binary `requires_payment` field could not distinguish a material price from a demanded virtue and was replaced by the `cost_structure`/`cost_timing`/`moral_condition` triple; a single `magic_nature` field conflated where a power came from with what made it fire, and was split into `magic_source` and `activation_condition`. In both cases the trigger for revision was extractions that were fully schema-conformant yet visibly lossy against the source text – the schema had no way to record a distinction the tale plainly made.

Here, we use "orthogonal" the term in a specific and modest sense: fixing a value on one dimension does not determine a value on any other, so the dimensions are free to vary independently. The clearest case is D1 against D2, where the prompt states as an explicit rule that a villain (a D2 value) causing an interaction yields `trigger_source=external_event` and not a villain-specific trigger; the same holds within D2, where a `villain` may take the functional role `trickster_helper`. We do not claim statistical independence in the extracted data – helper actors and zero-cost interactions plainly co-occur, and Section 4.3 is in part a description of which combinations are common. The claim is only that the schema does not force those co-occurrences by construction.

It's worth mentioning that six dimensions is a working decomposition, not a proof of sufficiency. It is adequate in the sense that we did not need a residual or "other" dimension to code the interactions encountered in this corpus and dense enough to support the analysis in Section 4.3. Dimensions we considered and left out (like affective register, the hero's stated motivation, repetition structure across parallel episodes) are candidates for extension rather than evidence against the present set.

The extraction prompt elicits one further field, `interaction_actions`, which lists the observable actions performed during the exchange. It is an auxiliary extraction field rather than a MAGIC attribute. We consider it for further diagnostics and analysis and do not include it in the current experimental analysis.

2.2 Schema description

D1: Trigger Modality (2 attributes) separates *who* caused an interaction to begin (`trigger_source`: hero agency, actor agency, or an external event) from *how* it was expressed (`trigger_mode`: request, discovery, autonomous action, imposition, or crisis), so that, e.g., a villain-imposed trigger is not conflated with the villain's identity.

D2: Actor Profile & Agency (7 attributes) captures the actor's stable structural type (`actor_type`: helper, villain, or object), its functional role in this specific interaction (`actor_functional_role`: ally, donor, trickster-helper, tester, obstacle, or neutral – deliberately allowed to diverge from structural type, e.g. a villain who tests then rewards the hero), its physical/ontological nature, its observed autonomy, and the balance of control between hero and actor.

D3: Magic Mechanics (6 attributes) covers the operational constraints on the magic itself: `magic_source` (inherent, bestowed, environmental, stolen, or transferred), `activation_condition` (what makes it fire), `cost_structure` and `cost_timing` (what the

Dim.	Value
D1 Trigger	hero_agency · request
D2 Actor	helper (animal), donor
D2 Autonomy	medium · balanced
D3 Magic	inherent · moral_test
D3 Cost	material, upfront · kindness
D3 Duration	single_use
D4 Outcome	test_passed
D4 Gain / Cost	object + information / material
D5 Function	NF-2 (Provision)
D6 Position	threshold; follows [1], precedes [4,5]
Actions	gives_object, gives_advice

Table 1: Worked example: the "black cat" interaction from *Baba Yaga and the Orphans*, showing how one interaction event populates all six MAGIC dimensions.

hero pays and when), `moral_condition` (which virtue is demanded, where the cost is behavioral), and `duration` (how long the effect persists).

D4: Outcome Valence (3 attributes) separates the primary narrative result (success, qualified success, a passed test, failure, or an unintended consequence) from what the hero concretely gains and loses, each a multi-value field, so that a productive-looking interaction with a hidden cost is not collapsed into a single valence label.

D5: Narrative Function (1 attribute) assigns one or more of seven Proppian-derived function codes (NF-1 Reconnaissance through NF-7 Obstacle) capturing the interaction's role in the plot.

D6: Narrative Position (4 attributes) situates the interaction in the story arc (departure, journey, threshold, climax, resolution) and records direct causal dependencies (`precedes`/`follows`) on other interactions in the same tale, encoding a lightweight causal chain rather than only story order.

A complete definition of attributes of six dimensions is provided in Appendix A.

An optional *actor network* extension records explicit relationships between non-hero actors (e.g., a witch who controls an enchanted object later given to the hero), with fields for relationship type, structural role (mediation, subordination, opposition, cooperation, or provision chain), direction, and chain depth to the hero. We describe it because it is part of the released knowledge base, but it is deliberately outside the scope of the evaluation: none of the three experiments in Section 4 operates on it, and no claim in this paper depends on it. A complete definition of actor network is provided in Appendix B.

Table 1 shows one interaction coded against the full schema extracted for *Baba Yaga and the Orphans*[1]: a black cat, requested for help by two orphans, tests them morally before rewarding correct behavior with an object and information at the cost of a material payment – a single event that would collapse into an undifferentiated "helper gives item" edge in a coarser schema, but here carries an explicit trigger, cost structure, moral condition, and position in the tale's causal chain. The same tale's non-hero actors are also linked into an actor network (Appendix B, Figure 2): the orphans receive help from three animal donors and use two magic objects to escape a villain.

3 Extraction Pipeline

We extract from a corpus of 1306 folktales spanning many cultural traditions and regions, with a maximum concentration from German-language sources, including the Brothers Grimm collection (Kaggle, 2024). Each tale is passed once to an LLM with a single prompt (see Appendix C) specifying all 25 fields, their closed vocabularies, and worked examples of ambiguous distinctions (e.g., `actor_type` vs. `actor_functional_role`); the model returns one JSON object per tale, an interaction list plus the ATU tale-type index (Uther, 2004). We ran this pipeline with three models: DeepSeek V3.2, Gemini 2.5 Flash, and Qwen 3.6 35B over the full corpus. Additionally, we ran GPT-5.2 and GPT-OSS-120B over a fixed 50-tale subset shared by all five models for extended cross-model comparison. The 50 tales were drawn uniformly at random from the corpus with a fixed seed before any extraction was run, and the same 50 were used for all five models. The pipeline is resumable and validates each response against the schema's vocabulary before accepting it (Section 4.1).

4 Experimental evaluation

4.1 Extraction Conformance and Schema Drift

For each model, we measure whether its output parses as JSON at all, and, among parsed interactions, whether every field is present and every value falls within the schema's declared vocabulary (Table 2). Parse success is high across the board (90–99%), but schema conformance – whether the *values* inside a successfully parsed interaction obey the schema – ranges from 80.6% (GPT-OSS-120B)

[1]Here and further we refer to folktales' titles giving examples for narrative processing as they were given in the dataset.

Model	Parse	Conform.	Int./tale
DeepSeek V3.2	96.7%	93.9%	5.10
Gemini 2.5 Flash	98.3%	88.3%	6.54
Qwen 3.6 35B	99.0%	90.0%	4.42
GPT-5.2†	98.0%	99.2%	5.43
GPT-OSS-120B†	90.0%	80.6%	4.00

Table 2: Extraction reliability by model. Parse = fraction of outputs parsing as JSON; Conform. = fraction of interactions with zero missing or out-of-vocabulary fields; Int./tale = mean interactions extracted per tale. †: 50-tale subset.

Model	Drift	Confusion	%
DeepSeek V3.2	424	244	57.5%
Gemini 2.5 Flash	1057	986	93.3%
Qwen 3.6 35B	617	576	93.4%
GPT-5.2†	2	2	100.0%
GPT-OSS-120B†	41	14	34.1%

Table 3: Schema drift decomposed into field confusion vs. novel values. Drift = total out-of-vocabulary values; Confusion = count valid for a different field; % = confusion share. †: 50-tale subset.

to 99.2% (GPT-5.2, though only measured on the smaller 50-tale subset), with the three full-corpus models spanning 88.3–93.9%. Mean interactions extracted per tale also vary substantially (4.42–6.54, for models run on the whole corpus), a first sign that models do not agree on the granularity at which a tale's magic interactions should be segmented, a point we return to in Section 4.2.

Field confusion vs. vocabulary incompleteness. We further classify every out-of-vocabulary value as *field confusion* (the value is valid for a *different* MAGIC field – e.g. `duration=always_active`, a valid `activation_condition` value) or a *novel value* (not in any field's vocabulary at all – e.g. `hero_gain=transportation`), by checking every drifted value against the full cross-field vocabulary index.

This split (Table 3) is strongly model-dependent: field confusion accounts for 93.3% and 93.4% of drift for Gemini and Qwen respectively, but only 57.5% for DeepSeek and 34.1% for GPT-OSS-120B, meaning roughly half to two-thirds of these two models' drift is genuinely novel, invented values rather than confusion with an existing field – a pattern that does not track model scale in any simple way. Critically, the two failure modes are not evenly spread across fields. Novel-value drift concentrates almost entirely

Field	κ_{50}	κ_{1306}	Δ
`actor_type` (D2)	0.44	0.45	+0.01
`outcome_type` (D4)	0.25	0.35	+0.11
`cost_structure` (D3)	0.15	0.33	**+0.18**
`actor_functional_role` (D2)	0.28	0.32	+0.04
`trigger_source` (D1)	0.32	0.32	−0.01
`narrative_function` (D5)	0.29	0.31	+0.03
`story_phase` (D6)	0.19	0.27	+0.08

Table 4: Fleiss' κ for one representative field per MAGIC dimension, in two independent evaluation conditions: κ_{50} – 5 models, 50-tale shared subset; κ_{1306} – the 3 full-corpus models only, all 1,306 tales. $\Delta = \kappa_{1306} - \kappa_{50}$. Rows ordered by κ_{1306}.

in `interaction_actions` and `moral_condition` across *all five* models – models keep inventing plausible action verbs (`gives_information`, `alliance_formation`, `provides_remedy`) and moral conditions (`obedience`, `loyalty`, `faith`, `purity`) beyond the schema's fixed lists, suggesting these two fields need either an open vocabulary or a substantially expanded closed one. Field-confusion drift, by contrast, concentrates in a specific, recurring set of field pairs: `magic_source` receiving `none` (not a valid value for that field, but the default for several cost-related fields, and the single largest drift category overall at 410 occurrences for Gemini alone), `duration` receiving `activation_condition` values, `actor_type` receiving `actor_nature` values (e.g. `animal`, `spirit`), and `trigger_source` receiving `trigger_mode` values (e.g. `crisis`) – boundaries between specific fields that are evidently ambiguous to extractors independent of which model is extracting.

4.2 Cross-Model Field Agreement

Interaction segmentation differs across models (Table 2), so interaction-to-interaction alignment across models is unreliable without itself imposing a modeling assumption. We instead compare models at the *tale* level, using their agreement as a lens on schema reliability rather than as a target metric in its own right: for each tale, we take each model's dominant (most frequent) value per field across the interactions it found, then measure multi-rater agreement with Fleiss' κ (Fleiss, 1971), computed directly from category counts, since the `NO_INTERACTIONS` vote described below needs a rater-by-category matrix that packaged implementations do not accept, and cross-validated against the `statsmodels` reference where both ap-

ply. James (2026) note that metric choice materially affects reported reliability in NLP annotation; we chose Fleiss' κ specifically because it corrects for chance agreement, which matters here (see below). A model that extracted zero interactions from a tale is treated as voting `NO_INTERACTIONS` for every field, since whether a model judges a tale to contain any magic interactions at all is itself part of what we measure; this also folds in a small number of outright parse failures, which is a limitation we return to in Section 6.

We report this in *two* conditions rather than one, because a single 50-tale estimate turned out not to be trustworthy on its own. κ_{50} uses all five models on the 50-tale subset GPT-5.2 and GPT-OSS-120B were restricted to; κ_{1306} uses only the three full-corpus models (DeepSeek, Gemini, Qwen) but all 1,306 tales – trading two raters for a 26× larger item count. Table 4 shows both.

No field reaches better than moderate agreement in either condition, and `actor_type` is the single highest-agreement field in both ($\kappa_{50} = 0.44$, $\kappa_{1306} = 0.45$) – these two conclusions are robust to sample size and rater set, and together form the paper's central reliability finding. Beyond that, however, the picture that motivated our first attempt at this section – a clean, monotonic gradient from "observable" to "interpretive" fields – does not survive the larger sample. `cost_structure` moves from the *worst*-agreement field at n=50 ($\kappa_{50} = 0.15$) to solidly mid-table at n=1,306 ($\kappa_{1306} = 0.33$, $\Delta = +0.18$, the largest shift of any field), and at n=1,306 six of the seven fields cluster in the "fair" band (0.27–0.35) with only `actor_type` clearing into "moderate" – none falls to "slight," unlike at n=50. The Spearman rank correlation between the two conditions' field orderings is only 0.21 (mean $|\Delta| = 0.06$, max $|\Delta| = 0.18$): the fine-grained ranking of the five middle fields is not a stable property of the schema at n=50, only the two extremes are (`actor_type` highest in both; `story_phase` lowest or second-lowest in both, $\kappa_{50} = 0.19$, $\kappa_{1306} = 0.27$). We read this as a caution that generalizes beyond this paper: cross-model κ estimates over few dozen shared items – a scale forced on us here by API cost, and not unusual in similar reliability studies – can rank-order schema fields in ways that a substantially larger sample overturns, even when the sample size is enough to establish the coarser, headline conclusion reliably.

Possible mislead in agreement metrics. In both conditions, `cost_structure` shows a large gap between κ and raw agreement: at n=50, 80% of tales have ≥4-of-5 models agreeing on the dominant value despite $\kappa_{50} = 0.15$; at n=1,306, 67.5% of tales have all 3 models agreeing despite $\kappa_{1306} = 0.33$. A possible explanation can be that `cost_structure=none` is an overwhelming majority class (84.8% of tale-level votes at n=50), so models trivially "agree" on `none` near the rate the base rate alone would produce – exactly what a chance-corrected statistic discounts and raw agreement does not. `story_phase`, by contrast, has both low raw agreement (28% at n=50) and low κ: its five values are far more evenly used (journey 44%, departure 20%, climax/threshold/resolution splitting the rest), so its low κ reflects genuine disagreement rather than a base-rate artifact. Reporting only raw agreement would have overstated `cost_structure` as reliable in both conditions – a distinction visible only once a chance-corrected and a raw statistic are reported side by side, independent of which of the two field-ranking pictures above one trusts.

A secondary, weaker signal points the same direction as the rest of this section: the mean pairwise Spearman correlation of per-tale interaction counts is 0.573 across all five models at n=50 but rises to 0.747 across the three full-corpus models at n=1,306 – another instance of a cross-model estimate looking noisier at the smaller sample size. Interaction-count means still differ substantially at full scale (4.4 for Qwen to 6.4 for Gemini per tale) – models do not agree on *how much* is happening in a given tale, before any question of how to characterize it arises, and this disagreement is not merely small-sample noise.

Four further checks (see Appendix D) support this reading. A bootstrap CI and a same-model-set control together confirm the `cost_structure` shift is a genuine sample-size effect, not resampling noise or which models happen to be included. Pairwise Cohen's κ across all model pairs shows disagreement broadly distributed rather than concentrated on one model, ruling out a single idiosyncratic rater as the explanation for weak aggregate agreement. A same-pipeline calibration field (`hero_nature`, nearly unambiguous from text) reaches κ = 0.64–0.79, well above every MAGIC dimension at matching sample size. These dimensions are specifically hard to determine from text, not uniformly hard to agree on under this pipeline. Finally, a worked disagreement case

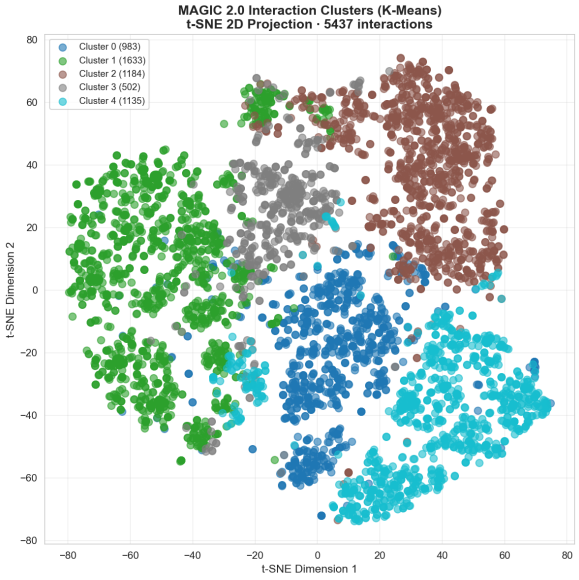


Figure 1: t-SNE projection of the five-cluster taxonomy discovered in the extraction.

shows some field-level "disagreement" is a segmentation artifact: models can agree on every interaction's content and still disagree on the tale-level summary if they extract different numbers of interactions overall.

4.3 Taxonomy Induction and Cross-Model Stability

Beyond individual-field agreement (Section 4.2), we ask a complementary question: does a higher-level structure (as an unsupervised taxonomy of interaction types) emerge from the extracted features, and is it stable across models? We build a 93-dimensional feature vector per interaction (one-hot categorical fields, ordinal fields normalized to $[0, 1]$, multi-value fields binarized) and run k-means. The optimal number of clusters k=5 was identified with the elbow method from $k \in [2, 10]$ using DeepSeek extraction and was fixed for cross-model comparability. Clustering was run independently on each of the three full-corpus extractions, excluding villain interactions (5437 / 6620 / 4097 interactions for DeepSeek, Gemini, and Qwen respectively). Figure 1 shows the resulting projection for DeepSeek.

The description that follows is a coherence check rather than evidence of the schema's superiority: clusters induced from MAGIC features are necessarily describable in MAGIC terms. What is not settled in advance is *which* dimensions organize the largest clusters. The two distinctions the schema draws most deliberately (D1's trigger modality and D2's split of structural type from functional role)

Archetype	N	Signature
Invoked helper	983	hero-requested (76%), helper (97%), NF-1/2/4 balanced
Bestowed object	1633	hero-controlled (91%), no cost (91%), NF-2, success (92%)
Moral-test gate	1184	moral cost (63%), NF-2/NF-3 mixed, mixed outcome
Object obstacle	502	actor-dominant (64%), NF-7 (57%)
Unsolicited helper	1135	actor-initiated (98%), no cost (99%), NF-1

Table 5: The five interaction archetypes discovered in the extraction.

could have been swamped by the other four; that they are instead the separating axes indicates they carry real variance in the corpus. A stronger test, clustering the same interactions under an alternative encoding (Proppian functions alone, or text embeddings), we leave to future work.

The five archetypes (Table 5) are readily nameable, recur across the corpus, and separate two schema dimensions that a coarser taxonomy would conflate. *Invoked helper* (e.g. *The Six Servants*) and *unsolicited helper* (*The Journey to the House of Herr Korbes*) share almost every attribute (helper actor, no cost, success) and differ essentially only in D1: whether the hero asked (76% hero-initiated) or the actor acted first (98% actor-initiated, e.g. a talking animal that volunteers advice before being asked) – exactly the trigger-modality distinction Section 2 introduces to avoid conflating an actor's identity with how an exchange began. *Bestowed object* (*Jack the Giant-Killer*) and *object obstacle* (*The Stolen Moon*) make the same separation on D2's actor-functional-role axis: both are dominated by `actor_type=object`, but one is almost purely helpful (91% hero-controlled, 92% success) while the other is almost purely adversarial (57% NF-7 Obstacle, actor- rather than hero-dominant). The same structural category (a non-sentient magic item) plays opposite functional roles depending on the interaction, which is exactly why D2 keeps `actor_type` and `actor_functional_role` as separate fields rather than one. *Moral-test gate* (*The Three Citrons*) is the one archetype defined by a reciprocal exchange rather than a one-way gift or obstacle: an actor (helper or object) withholds aid until the hero demonstrates a prescribed virtue, mixing NF-2 (Provision) and NF-3 (Threshold Test) roughly evenly and producing one of the least con-

centrated outcome distributions of the five (44% success, second only to *object obstacle*'s 34%; the rest split across qualified success, failure, and unintended consequence). The schema's cost/moral-condition fields exist specifically to capture this kind of conditional aid, which a simpler success/-failure outcome label would collapse into ordinary success. We match clusters across the three models via Hungarian assignment on cosine distance between centroids as the underlying one-hot feature space is defined by the fixed schema vocabulary, not by which model produced the data, so centroids from different models occupy the same coordinate space without any additional alignment step. We take centroid similarity as the primary criterion because what should be stable is the *content* of each archetype, which its centroid directly expresses. Partition-agreement measures (ARI, AMI) do not apply here: they compare two labelings of one item set, and the three extractions share no item set, since each model segments a tale into a different number of interactions (Section 4.2). An internal measure such as silhouette does apply, and would report separation within each partition rather than similarity across them; we report only the latter. The result is a mean matched-cluster cosine similarity of 0.953 across all three model pairs (DeepSeek–Gemini 0.964, DeepSeek–Qwen 0.944, Gemini–Qwen 0.949; minimum over all 15 individual matched-cluster pairs 0.858), with matched clusters sharing both their dominant attributes and, qualitatively, their example tales.

Additionally, we conducted analysis of the tale type and region (see Appendix E). Each archetype draws examples from a similar spread of ATU tale types – ATU 425 (*The Search for the Lost Husband*) is the single most frequent type in all five clusters (24–92 occurrences), and four more types (ATU 313, present in all five clusters' top five; ATU 300 and 750, each in three of five; ATU 301 and 480, each in two of five) recur across multiple archetypes rather than separating them; only a handful of types (e.g. ATU 563, 513, 470) are top-five for exactly one archetype. Region follows a similar pattern: German- and Scottish-sourced tales are the top two regions in every one of the five archetypes, so no archetype is a proxy for a single regional tradition. This indicates the MAGIC taxonomy captures a structurally distinct axis of variation (what kind of exchange this is) that cuts across tale type and regional tradition (which story this is and where it is from). Given this, we see the schema intended to complement the existing typological literature rather than replace or rename it (Section 5).

4.4 Reconciling Experiments

The taxonomy is stable at the level of *interaction archetypes* even though individual fields disagree at the level of *individual interactions*. We read this as evidence that most of the cross-model disagreement in Section 4.2 is about exactly where to draw a boundary within a shared underlying structure – e.g., precisely which `story_phase` an interaction falls into – rather than about whether that structure exists at all. For a knowledge base built this way, this suggests that aggregate, downstream structure derived from LLM extraction may be considerably more trustworthy than any single extracted field value, a distinction that reliability evaluations reporting only field-level or triple-level accuracy do not capture.

5 Related Work

Structural narrative theory. Propp's fixed-order function grammar (Propp, 1968) and Greimas's actantial model (Greimas, 1983) remain foundational accounts of folktale structure, and Campbell's monomyth (Campbell, 1949) traces the hero's arc at the story level. All three center the hero's overall trajectory or the plot's fixed sequence; MAGIC instead treats each hero–actor interaction as a bounded, independently-coded event, closer in spirit to a knowledge-base schema than to a narrative grammar, while reusing Propp's function vocabulary as one of its six dimensions (D5).

Folktale and narrative corpora. The ATU index (Uther, 2004) remains the standard typological catalogue; recent annotated corpora include the multilingual Bag-of-Tales (Hagedorn and Darányi, 2022), the German-language FairyNet character-network corpus (Schmidt et al., 2021), and a syntactically/semantically annotated French fairy-tale corpus (El Maarouf and Villaneau, 2012). These resources are richly annotated, but largely at the story or character level. MAGIC's interaction-level, cross-culturally scoped schema therefore extends this line of work rather than competing with it. Most directly related is concurrent work on narrative-centric knowledge structures for long-form text understanding (Tian et al., 2026), which builds entity- and storyline-level graphs to support retrieval-augmented reasoning over narrative text;

our schema operates at a finer grain (the individual interaction) and is evaluated for extraction reliability rather than downstream QA performance, making the two efforts complementary within the same broader problem.

LLM-based structured extraction. Generative information extraction with LLMs has become the dominant paradigm for populating knowledge bases from text (Xu et al., 2023; Bian, 2025). Closest to our reliability findings, Shrimal et al. (2025) show that static schemas cause unreliable extraction and propose iteratively optimizing the schema against measured extraction performance – our field-confusion/novel-value decomposition (Section 4.1) is a diagnostic of exactly the kind of schema weakness their approach is designed to correct, applied post-hoc rather than in a training loop. Srivastava et al. (2025) find that textual annotation guidelines materially improve LLM event extraction and cross-schema generalization, consistent with our prompt's reliance on worked examples for ambiguous field pairs (Section 3) and suggestive that our identified confusion pairs (e.g. `trigger_source` vs. `trigger_mode`) are precisely where such guidelines should be strengthened. Han and Lai (2026) argue that closed symbolic relation vocabularies lose the contextual nuance LLMs could otherwise represent in natural language; our finding that two of fourteen closed-vocabulary fields (`interaction_actions`, `moral_condition`) absorb the great majority of novel-value drift is direct empirical evidence for exactly this failure mode.

Agreement metrics for LLM annotation. James (2026) survey inter-annotator agreement metric selection for NLP annotation and evaluation, cautioning against uncritical use of raw agreement; our cost_structure result (Section 4.2) is a concrete instance of the failure mode such guidance warns against. We build directly on the classical chance-corrected agreement literature (Fleiss, 1971; Landis and Koch, 1977).

6 Conclusion and Future Work

The two central results – a taxonomy of interaction archetypes that is highly stable across extraction models, obtained even though no individual schema dimension is reliably extracted in isolation – together suggest a practical design principle for LLM-based knowledge base construction from schemas that, like ours, mix observable and interpretive fields: validate and report reliability at the level of the structure a knowledge base will actually be used for (clusters, types, aggregate statistics), not only at the level of individual extracted fields, since the latter can look considerably less trustworthy than the former. Practically, our field-confusion analysis also gives schema designers a concrete lever: fields whose drift is dominated by confusion with one specific neighboring field (`magic_source`, `duration`, `actor_type`, `trigger_source`) are candidates for clearer boundary definitions or worked disambiguating examples in the prompt (Srivastava et al., 2025), while fields whose drift is dominated by novel, plausible values (`interaction_actions`, `moral_condition`) are candidates for an explicitly open or substantially expanded vocabulary (Han and Lai, 2026) rather than further prompt engineering against a fixed list.

We see four directions. First, a human-annotated gold subset: the results here measure consistency under LLM extraction, and only annotation against the same schema can turn them into statements about accuracy or expose correlated error. Second, sensitivity of the taxonomy: how matched-centroid similarity and internal cluster quality behave across k, whether the merges predicted in Section 4.3 occur, and whether the archetypes survive restriction to tales all three models segment identically. Third, comparison against alternative encodings of the same interactions – Proppian functions alone, or text embeddings – separating what is specific to MAGIC's distinctions from what any reasonable representation recovers. Fourth, replacing mode-based tale-level aggregation with interaction alignment (Appendix D); and beyond the evaluation, folkloristic analysis of the induced archetypes with domain experts, and propagation of the schema's core configuration (a protagonist engaging an entity of bounded, extraordinary capability under explicit triggers, costs and conditions) to other domains of complex human behavior.

Acknowledgments

This work was supported by the grant of the state program of the «Sirius» Federal Territory «Scientific and technological development of the «Sirius» Federal Territory» (Agreement №34-03 date 26.08.2026).

Limitations

All reliability and agreement results in this paper are inter-model rather than model-versus-ground-truth: we have no human-annotated gold standard at this stage, and they should be read as measuring the consistency of the schema under LLM-based extraction, not extraction accuracy. The distinction matters more than the phrasing suggests. Agreement among extractors is compatible with all of them being wrong in the same way, and models with overlapping pretraining data are exactly the population in which such correlated error is plausible; a shared prior about what a folktale helper does would produce convergence indistinguishable, by our measurements, from convergence on the text. This applies to the taxonomy-stability result as well as to the field-level κ values. The taxonomy result mitigates the concern partially, by showing that structure survives a change of extractor and so is not any single model's idiosyncrasy, and the `hero_nature` calibration shows the pipeline can reach high agreement where the text is unambiguous – but neither substitutes for ground truth, and we do not treat them as doing so. A human-annotated subset is the first item of future work for this reason.

Two of the five models were only run on a 50-tale subset for cost reason. Section 4.2 reports agreement in two conditions specifically to compensate: all five models on small subset, and the three full-corpus models alone on all 1,306 tales. However, the taxonomy-stability result (Section 4.3) still uses the three full-corpus models only.

The taxonomy is induced at a single value of k, selected by the elbow method and held fixed across models for comparability. The archetypes in Table 5 are therefore a description of the corpus at one granularity, and a coarser or finer partition would yield a different, plausibly also stable, set of archetypes. The claim we defend is that the partition is reproducible across extractors, not that five is the correct number of interaction types.

Finally, our corpus obtained from (Kaggle, 2024), while multi-regional, is not a completely random or representative sample of world folk narrative, and the taxonomy in Section 4.3 should be read as a description of recurring patterns in this specific corpus rather than a claim of narratological universality.

We consider these limitations as the directions for further work in the scope of ongoing research project.

References

Haonan Bian. 2025. LLM-empowered knowledge graph construction: A survey. *Preprint*, arXiv:2510.20345.

Joseph Campbell. 1949. *The Hero with a Thousand Faces*. Pantheon Books, New York.

Sara Graça da Silva and Jamshid J. Tehrani. 2016. Comparative phylogenetic analyses uncover the ancient roots of Indo-European folktales. *Royal Society Open Science*, 3(1):150645.

Ismail El Maarouf and Jeanne Villaneau. 2012. A french fairy tale corpus syntactically and semantically annotated. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 345–350. European Language Resources Association (ELRA).

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Algirdas Julien Greimas. 1983. *Structural Semantics: An Attempt at a Method*. University of Nebraska Press, Lincoln, NE. Translated by D. McDowell, R. Schleifer, and A. Velie. Original work published 1966.

Julian Hagedorn and Sándor Darányi. 2022. Bearing a bag-of-tales: An open corpus of annotated folktales for reproducible research. *Journal of Open Humanities Data*, 8(16):1–10.

Kanyao Han and Yushang Lai. 2026. From symbolic to natural-language relations: Rethinking knowledge graph construction in the era of large language models. *Preprint*, arXiv:2601.09069.

Joseph James. 2026. Counting on consensus: Selecting the right inter-annotator agreement metric for NLP annotation and evaluation. *Preprint*, arXiv:2603.06865.

Kaggle. 2024. 1000+ Folk Stories around the World. `https://www.kaggle.com/datasets/chayanonc/1000-folk-stories-around-the-world`.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Max Lüthi. 1986. *The European Folktale: Form and Nature*. Indiana University Press, Bloomington, IN. Translated by J. D. Niles.

Vladimir Propp. 1968. *Morphology of the Folktale*. University of Texas Press, Austin, TX. Translated by L. Scott, revised by L. A. Wagner. Original work published 1928.

Diandra Schmidt, Albin Zehe, Janis Lorenzen, Lena Sergel, Sabine Dücker, Markus Krug, and Frank Puppe. 2021. The FairyNet corpus – character networks for german fairy tales. In *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 49–56. Association for Computational Linguistics.

Anubhav Shrimal, Aryan Jain, Soumyajit Chowdhury, and Promod Yenigalla. 2025. PARSE: LLM driven schema optimization for reliable entity extraction. *Preprint*, arXiv:2510.08623.

Saurabh Srivastava, Sweta Pati, and Ziyu Yao. 2025. Instruction-tuning LLMs for event extraction with annotation guidelines. *Preprint*, arXiv:2502.16377.

Qiuyu Tian, Fengyi Chen, Yiding Li, Youyong Kong, Fan Guo, Yuyao Li, Jinjing Shen, Zhijing Xie, Yiyun Luo, Xin Zhang, Yingce Xia, and Zequn Liu. 2026. Narrative knowledge weaver: Narrative-centric retrieval-augmented reasoning for long-form text understanding. *Preprint*, arXiv:2606.05724.

Hans-Jörg Uther. 2004. *The Types of International Folktales: A Classification and Bibliography*. Number 284–286 in FF Communications. Finnish Academy of Science and Letters, Helsinki.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2023. Large language models for generative information extraction: A survey. *Preprint*, arXiv:2312.17617.

A MAGIC Attribute Definitions

Table 6 gives the full value set and definition for every interaction-level attribute across the six MAGIC dimensions, expanding the summary in Section 2. This is the same vocabulary implemented in `magiclib/schema.py` and used to compute schema conformance in Section 4.1.

Dim.	Attribute	Values
D1	`trigger_source`	`hero_agency` – hero initiated through a deliberate request, search, or invocation; `actor_agency` – the magic entity acted first, unbidden by the hero; `external_event` – an outside force (villain, crisis, fate) imposed the trigger
D1	`trigger_mode`	`request` – explicit ask or invocation; `discovery` – hero encounters entity by chance or exploration; `autonomous` – entity acts on its own initiative; `imposed` – force, attack, or command pressed upon the hero; `crisis` – urgent emergency drives the contact
D2	`actor_name`	Literal identifier of the entity (string)
D2	`actor_nature`	`human` / `animal` / `plant` / `object` / `natural_force` / `spirit` / `other`
D2	`actor_type`	`helper` – exists to assist protagonists; `villain` – exists to obstruct or harm the hero; `object` – non-sentient or minimally sentient magic item
D2	`actor_functional_role`	`ally` – cooperates directly with the hero's goal; `donor` – provides object/ability/information then steps back; `trickster_helper` – structurally antagonistic but functionally enabling; `tester` – evaluates the hero's worthiness before granting anything; `obstacle` – actively blocks the hero's path or goal; `neutral` – magic is incidental, no stake in hero's goal
D2	`actor_autonomy` (ordinal)	`none` – inert without activation, purely reactive; `low` – reacts to commands or triggering conditions only; `medium` – limited independent action within a defined scope; `high` – makes autonomous decisions within situational rules; `full` – fully self-directed agency, acts on personal will
D2	`agency_balance` (ordinal)	`hero_dominant` – hero directs, actor executes; `balanced` – both parties contribute meaningfully; `actor_dominant` – actor determines what happens, hero responds or receives
D2	`interaction_hierarchy`	`primary` – directly aids or confronts the hero, no intermediary; `secondary` – aids the primary actor, not the hero directly; `servant_object` – controlled by another non-hero actor in service of the hero; `chain_mediated` – reaches the hero through one or more intermediary actors
D3	`magic_source`	`inherent` – always magical by nature (e.g. dragon, fairy); `bestowed` – enchanted externally, magic is transferred or lent; `environmental` – power drawn from a place, time, or natural condition; `stolen` – magic seized or taken from its rightful bearer; `transferred` – magic passed deliberately from one actor/object to another

Dim.	Attribute	Values
D3	activation_condition	always_active – magic operates continuously without a trigger; hero_request – activated by the hero's explicit invocation or command; crisis_triggered – activates automatically when the hero is in danger; moral_test – activates only if the hero passes a behavioral test; phrase_required – requires specific words, gesture, or ritual
D3	cost_structure	none – no cost to the hero; material – pays with a physical item (food, gold, blood); moral_behavior – must behave in a specific virtuous or prescribed way; knowledge_proof – must answer a question, solve a riddle, or demonstrate knowledge; sacrifice – gives up something of lasting personal value (years, a name, a memory); reciprocal_service – must perform a task or service in return
D3	cost_timing	none – no cost applies; upfront – cost paid before magic is granted; deferred – cost extracted after the benefit is received; ongoing – cost accumulates or repeats during use; conditional – cost only applies if a condition is later met or broken
D3	moral_condition	none / kindness / respect / courage / honesty / humility / specific_task (only meaningful when cost_structure=moral_behavior)
D3	duration	permanent – effect is irreversible and lasts indefinitely; time_limited – effect expires after a fixed period; task_limited – effect persists until a specific task is complete; single_use – magic is consumed in a single activation
D4	outcome_type	success – hero fully achieves the intended goal at no unexpected cost; qualified_success – goal achieved but with a meaningful complication or reduced result; test_passed – hero survives or passes an evaluation, gain is informational or permissive; failure – hero does not achieve the goal; unintended_consequence – outcome differs substantially from what the hero sought
D4	hero_gain (multi-value)	none / object / information / ability / rescue / alliance / permission / transformation
D4	hero_cost (multi-value)	none / material / moral_debt / constraint / transformation / delayed_consequence
D5	narrative_function (multi-value)	NF-1 Reconnaissance/Information – actor provides knowledge the hero needs; NF-2 Provision – actor gives an object, ability, or power; NF-3 Threshold Test – actor evaluates hero's virtue or skill; NF-4 Rescue/Intervention – actor saves the hero from immediate danger; NF-5 Transformation – actor changes the hero's form, nature, or status; NF-6 Alliance Formation – actor commits to an ongoing aid relationship; NF-7 Obstacle/Opposition – actor's primary function is to impede the hero
D6	story_phase	departure – hero is leaving home or entering the quest world; journey – hero is in transit, accumulating helpers and objects; threshold – hero encounters a gateway figure or test before the main challenge; climax – the decisive confrontation or challenge; resolution – events after the main challenge (return, reward, restoration)
D6	interaction_index	Sequential index (1, 2, 3, . . .) in order of appearance in the text

Dim.	Attribute	Values
D6	`precedes` / `follows`	Lists of `interaction_index` values this interaction directly enables/is enabled by (direct causal or enabling links only, not general story order)

Table 6: Full attribute definitions for the MAGIC schema (Section 2).

B Actor Network Edge Attributes

Table 7 defines the optional actor-network extension (Section 2), which records relationships between non-hero actors – e.g., in the extraction for *Geraint and Enid*, a `controls` edge from a sparrow-hawk knight to the dwarf who serves him. *Baba Yaga and the Orphans* (Figure 2) has no non-hero-to-non-hero relationships and so an empty actor network; this extension is not the object of any of the three experiments in this paper.

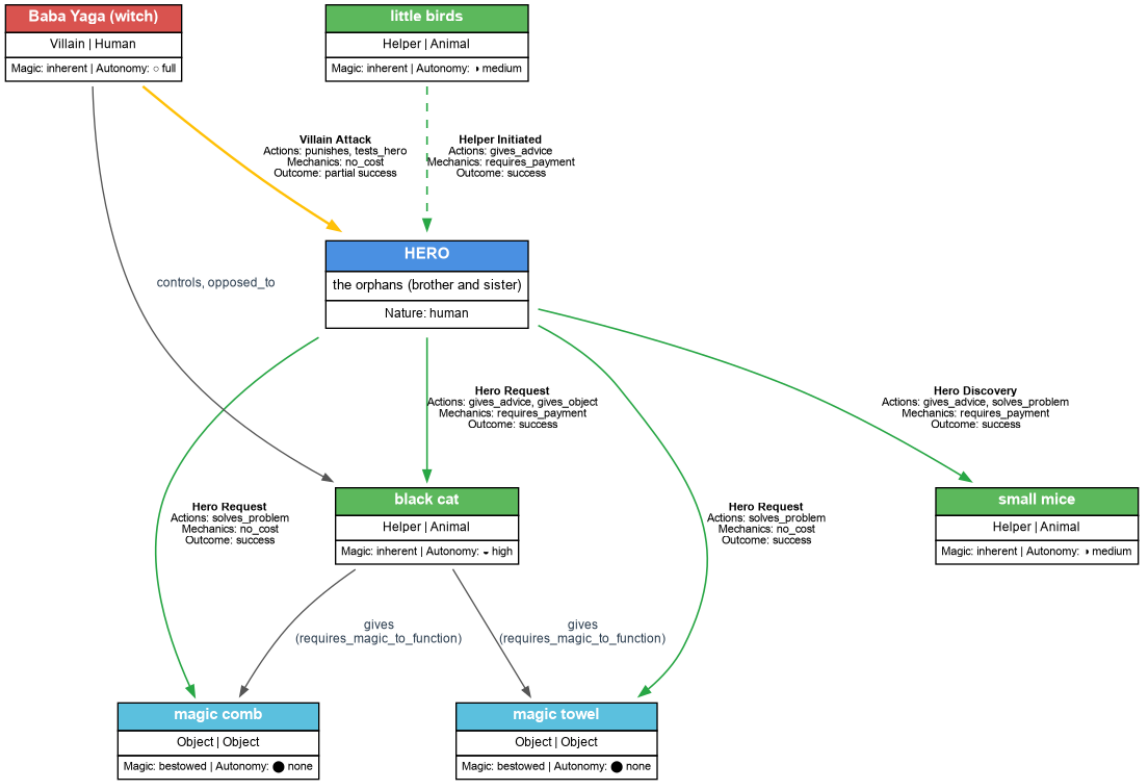


Figure 2: Actor network for *Baba Yaga and the Orphans*.

Attribute		Values
`source_actor` / `target_actor`	/	Names listed in `actor_name` (D2)
`direction`		`unidirectional` / `bidirectional`
`interaction_indices`		List of `interaction_index` (D6) values this relationship directly enables, explains, or produces; empty if none

Attribute	Values
relationship_type (multi-value)	controls – direct command or dominance over target; created – made or summoned the target; gives – transfers an object or ability to target (or to hero via target); takes – seizes something from target; allied_with – cooperative relationship or pact; opposed_to – active conflict or antagonism; depends_on – functional reliance, cannot operate without target; contains – target is physically enclosed within source; mediates – acts as intermediary between target and hero; transforms – changes target's form, nature, or power; deceives – misleads or entraps target
network_function	mediation – source passes something toward the hero via target; subordination – source has authority over target (master–servant or owner–tool); opposition – source and target are in active conflict affecting the hero; cooperation – source and target act jointly for a shared goal; provision_chain – source → target → hero forms a supply chain of magic objects or aid
magic_dependency	none / requires_magic_to_function / enhanced_by_magic
temporal_relation	before_hero_interaction – established before any linked interaction; concurrent – active during the linked interaction(s); after_hero_interaction – arises as a consequence of the hero's interactions
chain_depth	0 – either source or target interacts directly with the hero; 1 – one intermediary actor; 2+ – deeper chain (exact integer)

Table 7: Actor-network edge attribute definitions (Section 2).

C Extraction Prompt

The full prompt used by extract.py for all five models (Section 3), reproduced verbatim from code_draft/prompts/magic_extraction_v1.md, is given below. The story text is appended after this prompt at extraction time.

```
# TASK
Analyze the folktale to identify **all significant interactions between the HERO and MAGIC ACTORS/
    OBJECTS**. Focus exclusively on:
- Magic helpers (e.g., talking animals, spirits)
- Magic objects (e.g., self-moving items, enchanted tools)
- Villains using magic against the hero

---

## PART 1: HERO INTERACTIONS (PER EVENT)

For **EACH INTERACTION EVENT** between hero and magic entity (not per actor), extract all fields
    below.
Fields are grouped by the six MAGIC framework dimensions. Within each dimension, all fields are **
    required**.

---

### STORY-LEVEL FIELDS *(same value for every interaction)*

| # | Field | Values |
|---|-------|--------|
| 1 | **Hero Name** | Protagonist's name or identifier |
| 2 | **Hero Nature** | `human` / `animal` / `plant` / `object` / `natural_force` / `spirit` / `
    other` |

---
```

```
### DIMENSION 1  Trigger Modality
*Who and how the interaction was set in motion. Use TWO orthogonal fields instead of one.*

**3. `trigger_source`**  Who caused the trigger:
- `hero_agency`  The hero initiated (request, deliberate search)
- `actor_agency`  The magic entity acted first, unbidden
- `external_event`  An outside force (villain, crisis, fate) imposed the trigger

**4. `trigger_mode`**  How the trigger was expressed (independent of source):
- `request`  Explicit ask or invocation by the hero
- `discovery`  Hero encounters entity by chance or exploration
- `autonomous`  Entity acts on its own initiative, no external prompt
- `imposed`  Force, attack, or command pressed upon the hero
- `crisis`  Urgent emergency drives the contact

> **Rule:** Assign `trigger_source` and `trigger_mode` independently. Example  Baba Yaga seizing
    children: `trigger_source: external_event`, `trigger_mode: imposed`. Do **not** let the actor's
    identity dictate `trigger_source`.

---

### DIMENSION 2  Actor Profile & Agency Distribution
*Who the actor is (structurally and functionally) and how much control each party holds.*

**5. `actor_name`**  Name or description of the magic actor/object.

**6. `actor_nature`**  Physical/ontological nature:
`human` / `animal` / `plant` / `object` / `natural_force` / `spirit` / `other`

**7. `actor_type`**  **Structural identity** in the story world (stable across interactions):
- `helper`  Exists to assist protagonists
- `villain`  Exists to obstruct or harm the hero
- `object`  Non-sentient or minimally sentient magic item

**8. `actor_functional_role`**  **Narrative function in THIS interaction** (may differ from `
    actor_type`):
- `ally`  Cooperates directly with the hero's goal
- `donor`  Provides object, ability, or information and steps back
- `trickster_helper`  Structurally antagonistic but functionally enabling (e.g., Baba Yaga)
- `tester`  Evaluates the hero's worthiness before granting anything
- `obstacle`  Actively blocks the hero's path or goal
- `neutral`  Magic is incidental; entity has no stake in the hero's quest

> **Rule:** `actor_type` and `actor_functional_role` **can and often should differ**. A villain (`
    actor_type`) who tests then rewards the hero is a `trickster_helper` (`actor_functional_role`).
    Always fill both.

**9. `actor_autonomy`**  Operational independence *as observed in this interaction*:
- `none`  Inert without activation; purely reactive to physical use
- `low`  Reacts to commands or triggering conditions only
- `medium`  Limited independent action within a defined scope
- `high`  Makes autonomous decisions but within situational rules
- `full`  Fully self-directed agency; acts on personal will

> **Rule:** Base this on **observed behavior in the text**, not the entity's general reputation.

**10. `agency_balance`**  Balance of control between hero and actor in this interaction:
- `hero_dominant`  Hero directs; actor executes (e.g., commanded magic objects)
- `balanced`  Both parties contribute meaningfully to the outcome
- `actor_dominant`  Actor determines what happens; hero responds or receives

> **Rubric:** If the hero could substitute a different actor with the same outcome  `hero_dominant`.
    If removing either party's choice collapses the outcome  `balanced`. If the actor could have
    acted regardless of hero's presence  `actor_dominant`.

**11. `interaction_hierarchy`**  Position in the interaction dependency chain:
- `primary`  Directly aids or confronts the hero (no intermediary)
- `secondary`  Aids the primary actor, not the hero directly
- `servant_object`  Controlled by another non-hero actor in service of hero
```

```
- `chain_mediated`  Reaches hero through one or more intermediary actors

---

### DIMENSION 3  Magic Mechanics
*The operational constraints governing how the magic works.*

**12. `magic_source`**  Origin of the magical property (renamed from `magic_nature`):
- `inherent`  Always magical by nature (e.g., dragon, fairy)
- `bestowed`  Enchanted externally; magic is transferred or lent
- `environmental`  Power drawn from a place, time, or natural condition
- `stolen`  Magic seized or taken from its rightful bearer
- `transferred`  Magic passed deliberately from one actor/object to another

**13. `activation_condition`**  What causes the magic to fire:
- `always_active`  Magic operates continuously without a trigger
- `hero_request`  Activated by the hero's explicit invocation or command
- `crisis_triggered`  Activates automatically when the hero is in danger
- `moral_test`  Activates only if the hero passes a behavioral test
- `phrase_required`  Requires specific words, gesture, or ritual

**14. `cost_structure`**  What the hero must give, do, or risk to use the magic (replaces binary `
    requires_payment`):
- `none`  No cost to the hero
- `material`  Pays with a physical item (food, gold, blood)
- `moral_behavior`  Must behave in a specific virtuous or prescribed way
- `knowledge_proof`  Must answer a question, solve a riddle, or demonstrate knowledge
- `sacrifice`  Gives up something of lasting personal value (years, a name, a memory)
- `reciprocal_service`  Must perform a task or service in return

**15. `cost_timing`**  When the cost is extracted:
- `none`  No cost applies
- `upfront`  Cost paid before magic is granted
- `deferred`  Cost extracted after the benefit is received
- `ongoing`  Cost accumulates or repeats during use
- `conditional`  Cost only applies if a condition is later met or broken

**16. `moral_condition`**  Specific ethical requirement, if `cost_structure` is `moral_behavior` (
    otherwise `none`):
`none` / `kindness` / `respect` / `courage` / `honesty` / `humility` / `specific_task`

**17. `duration`**  How long the magical effect persists:
- `permanent`  Effect is irreversible and lasts indefinitely
- `time_limited`  Effect expires after a fixed period
- `task_limited`  Effect persists until a specific task is complete
- `single_use`  Magic is consumed in a single activation

---

### DIMENSION 4  Outcome Valence
*What actually resulted for the hero. Use THREE fields instead of one.*

**18. `outcome_type`**  The primary narrative result:
- `success`  Hero fully achieves the intended goal at no unexpected cost
- `qualified_success`  Goal achieved but with a meaningful complication or reduced result
- `test_passed`  Hero survives or passes an evaluation; gain is informational or permissive
- `failure`  Hero does not achieve the goal
- `unintended_consequence`  Outcome differs substantially from what the hero sought

**19. `hero_gain`**  What the hero concretely receives **(MULTIPLE ALLOWED)**:
`none` / `object` / `information` / `ability` / `rescue` / `alliance` / `permission` / `
    transformation`

**20. `hero_cost`**  What the hero concretely gives up or incurs **(MULTIPLE ALLOWED)**:
`none` / `material` / `moral_debt` / `constraint` / `transformation` / `delayed_consequence`

> **Example:** Baba Yaga tests the orphans: `outcome_type: test_passed`, `hero_gain: ["information",
    "permission"]`, `hero_cost: ["moral_debt"]`. This is distinct from a `qualified_success`, which
    implies partial goal attainment.
```

```
---

### DIMENSION 5  Narrative Function *(NEW)*
*What role this interaction plays in the plot architecture.*

**21. `narrative_function`**  Select the primary Proppian-grounded function code:

| Code | Name | When to use |
|---|---|---|
| `NF-1` | **Reconnaissance / Information** | Actor provides knowledge the hero needs to continue |
| `NF-2` | **Provision  Magical Agent** | Actor gives the hero an object, ability, or power |
| `NF-3` | **Threshold Test** | Actor evaluates hero's virtue or skill before granting anything |
| `NF-4` | **Rescue / Intervention** | Actor saves the hero from immediate physical danger |
| `NF-5` | **Transformation** | Actor changes the hero's form, nature, or social status |
| `NF-6` | **Alliance Formation** | Actor commits to an ongoing aid relationship with the hero |
| `NF-7` | **Obstacle / Opposition** | Actor's primary function is to impede the hero's progress |

> **Rule:** Choose the *dominant* function of the interaction. If an actor provides information **and
    ** an object, assign the function that drives the plot forward. Multiple functions may be listed
     if genuinely co-equal: `["NF-1", "NF-2"]`.

---

### DIMENSION 6  Narrative Position *(NEW)*
*Where in the story arc the interaction occurs and how it connects to others.*

**22. `story_phase`**  Macro position in the tale's narrative arc:
- `departure`  Hero is leaving home or entering the quest world
- `journey`  Hero is in transit, accumulating helpers and objects
- `threshold`  Hero encounters a gateway figure or test before the main challenge
- `climax`  The decisive confrontation or challenge
- `resolution`  Events after the main challenge (return, reward, restoration)

**23. `interaction_index`**  Sequential integer (1-based) indicating this interaction's position
    among all magic interactions in the tale, in order of appearance in the text.

**24. `precedes`**  List of `interaction_index` values this interaction **directly enables or causes
    **. Empty list `[]` if none.

**25. `follows`**  List of `interaction_index` values that **directly enabled or caused** this
    interaction. Empty list `[]` if none.

> **Rule:** Use `precedes`/`follows` only for direct causal or enabling links (e.g., Actor A gives
    Hero an object at index 2; Hero uses that object at index 5  index 2 has `"precedes": [5]`,
    index 5 has `"follows": [2]`). Do not record every story-order predecessor  only the
    interactions this one depends on.

---

### INTERACTION ACTIONS *(unchanged, multiple allowed)*

**26. `interaction_actions`**  Observable actions performed during the interaction **(MULTIPLE
    ALLOWED)**:
`gives_object` / `gives_advice` / `transforms_hero` / `transforms_object`
`performs_task` / `solves_problem` / `tests_hero` / `rescues`
`deceives` / `punishes` / `demands_payment` / `grants_passage` / `reveals_information`

---

## PART 2: ACTOR INTERCONNECTIONS (TOPOLOGICAL)

For **EACH SIGNIFICANT RELATIONSHIP** between non-hero actors (e.g., helper A controls object B):

**1. `source_actor`**  Name of influencing actor (must appear in `interactions`).

**2. `target_actor`**  Name of influenced actor (must appear in `interactions`).

**3. `relationship_type`** *(MULTIPLE ALLOWED)*:
- `controls`  Direct command or dominance over target
- `created`  Made or summoned the target
```

```
- `gives`  Transfers an object or ability to target (or to hero via target)
- `takes`  Seizes something from target
- `allied_with`  Cooperative relationship or pact
- `opposed_to`  Active conflict or antagonism
- `depends_on`  Functional reliance; cannot operate without target
- `contains`  Target is physically enclosed within source
- `mediates`  Acts as intermediary between target and hero
- `transforms`  Changes target's form, nature, or power
- `deceives`  Misleads or entraps target

**4. `direction`**: `unidirectional` / `bidirectional`

**5. `network_function`** *(NEW)*  Structural role of this relationship in the magic system:
- `mediation`  Source passes something (object, information, effect) toward the hero via target
- `subordination`  Source has authority over target (masterservant or ownertool)
- `opposition`  Source and target are in active conflict that affects the hero
- `cooperation`  Source and target act jointly for a shared goal
- `provision_chain`  Source  target  hero forms a supply chain of magic objects or aid

**6. `chain_depth`** *(NEW)*  Number of intermediary actors between this relationship and the hero:
- `0`  Either source or target interacts directly with the hero
- `1`  One actor between this relationship and direct hero contact
- `2+`  Deeper chain (use exact integer)

**7. `temporal_relation`** *(NEW)*  When this inter-actor relationship occurs relative to the hero's
    interactions:
- `before_hero_interaction`  Relationship established before any linked interaction
- `concurrent`  Active during the linked interaction(s)
- `after_hero_interaction`  Arises as a consequence of the hero's interactions

**8. `interaction_indices`** *(NEW)*  List of `interaction_index` values from Part 1 that this
    relationship directly enables, explains, or produces. Empty list `[]` if none.

**9. `magic_dependency`**: `none` / `requires_magic_to_function` / `enhanced_by_magic`

---

## RULES

### Annotation principles
- **ONE ENTRY PER INTERACTION EVENT.** If a magic sword is used three times, create three entries.
    Each use is a distinct event with its own trigger, outcome, and narrative function.
- **Order by text appearance.** `interaction_index` must reflect the sequence in the original
    narrative.
- **Observed behavior only.** All fields  especially `actor_autonomy`, `agency_balance`, and `
    actor_functional_role`  must reflect what the text shows, not general knowledge about the entity
     type.
- **`actor_type` vs `actor_functional_role` can diverge.** This is expected for Trickster-Tester
    figures. Both fields are always required.
- **`trigger_source` is independent of `actor_type`.** A villain causing a trigger  `trigger_source:
     external_event`, not a villain-specific trigger value.
- **`hero_name` and `hero_nature` are story-level.** Do not vary them across interaction entries.

### Completeness rules
- Include **all** interactions involving the HERO and MAGIC entities. Omit mundane human interactions
    .
- For hybrid actors (e.g., cursed prince), use the **dominant nature during this specific interaction
    **.
- If no magic interactions exist, return `"interactions": []`.
- `actor_network` covers **only non-hero actors**. Hero relationships belong in `interactions`.
- **Actor network relationships must be explicit in the text.** No speculative connections.
- If no non-hero relationships exist, return `"actor_network": []`.

### Causal chain rules
- `precedes`/`follows` encode **direct enabling links only**, not every prior interaction.
- `chain_depth` in `actor_network` counts actors between the relationship and the hero, not
    narrative steps.
- `interaction_indices` in `actor_network` entries must reference `interaction_index` values that
    exist in `interactions`.
```

```
### ATU classification
- Identify the Aarne-Thompson-Uther Index (ATU) according to the predominant tale type. Use the most
    specific applicable code.

---

## FORMAT

Output **STRICT JSON** with the structure below. No text before or after. No explanations. No
    comments inside the JSON.

```json
{
  "story_title": "Short story name",
  "ATU_Index": "Identified ATU Index",
  "hero_name": "Protagonist identifier",
  "hero_nature": "human|animal|plant|object|natural_force|spirit|other",
  "interactions": [
    {
      "interaction_index": 1,
      "actor_name": "...",
      "actor_type": "helper|object|villain",
      "actor_functional_role": "ally|donor|trickster_helper|tester|obstacle|neutral",
      "actor_nature": "human|animal|plant|object|natural_force|spirit|other",
      "actor_autonomy": "none|low|medium|high|full",
      "agency_balance": "hero_dominant|balanced|actor_dominant",
      "interaction_hierarchy": "primary|secondary|servant_object|chain_mediated",
      "trigger_source": "hero_agency|actor_agency|external_event",
      "trigger_mode": "request|discovery|autonomous|imposed|crisis",
      "magic_source": "inherent|bestowed|environmental|stolen|transferred",
      "activation_condition": "always_active|hero_request|crisis_triggered|moral_test|phrase_required
          ",
      "cost_structure": "none|material|moral_behavior|knowledge_proof|sacrifice|reciprocal_service",
      "cost_timing": "none|upfront|deferred|ongoing|conditional",
      "moral_condition": "none|kindness|respect|courage|honesty|humility|specific_task",
      "duration": "permanent|time_limited|task_limited|single_use",
      "interaction_actions": ["gives_object", "gives_advice"],
      "narrative_function": ["NF-1"],
      "story_phase": "departure|journey|threshold|climax|resolution",
      "outcome_type": "success|qualified_success|test_passed|failure|unintended_consequence",
      "hero_gain": ["object", "information"],
      "hero_cost": ["none"],
      "precedes": [],
      "follows": []
    }
  ],
  "actor_network": [
    {
      "source_actor": "...",
      "target_actor": "...",
      "relationship_type": ["gives", "controls"],
      "network_function": "mediation|subordination|opposition|cooperation|provision_chain",
      "direction": "unidirectional|bidirectional",
      "chain_depth": 0,
      "temporal_relation": "before_hero_interaction|concurrent|after_hero_interaction",
      "interaction_indices": [1, 3],
      "magic_dependency": "none|requires_magic_to_function|enhanced_by_magic"
    }
  ]
}
```

NO TEXT BEFORE/AFTER JSON. NO EXPLANATIONS.

TEXT
```

D Pairwise Agreement, Confidence Intervals, and a Disagreement Case Study

This appendix gives the full detail behind the four checks summarized in Section 4.2.

Bootstrap confidence intervals. Table 8 gives 95% CIs (2,000 tale-resampling bootstrap iterations) on every field's Fleiss' κ, in both conditions, alongside the calibration field `hero_nature` (story-level: `human/animal/plant/object/natural_force/spirit/other`, constant across a tale and almost always unambiguous from the text). Two things stand out. First, every CI at n=1,306 is narrow (width $\leq$0.06 for six of the eight fields, $\leq$0.08 for the remaining two, `cost_structure` and `hero_nature`), so the full-corpus estimates in Table 4 are precise, not noisy. Second, `cost_structure`'s n=50 CI, $[0.040, 0.243]$, does not contain its own n=1,306 estimate (0.328) – the discrepancy in Table 4 is not explained away by ordinary tale-resampling uncertainty at n=50. To check whether this is instead a *model-set* effect (recall κ_{50} uses five models, κ_{1306} only three), we recomputed κ for `cost_structure` using only the three full-corpus models restricted to the same 50 tales: $\kappa = 0.148$, CI $[-0.057, 0.373]$ – essentially identical to the five-model n=50 estimate (0.151) and nothing like the n=1,306 estimate. Sample size, not which models are included, is responsible for the shift.

Field	κ_{50} [95% CI]	κ_{1306} [95% CI]
`trigger_source`	0.32 [0.22, 0.41]	0.32 [0.29, 0.34]
`actor_type`	0.44 [0.33, 0.56]	0.45 [0.42, 0.48]
`actor_functional_role`	0.28 [0.19, 0.37]	0.32 [0.30, 0.35]
`cost_structure`	0.15 [0.04, 0.24]	0.33 [0.29, 0.37]
`outcome_type`	0.25 [0.15, 0.34]	0.35 [0.33, 0.38]
`narrative_function`	0.29 [0.21, 0.37]	0.31 [0.29, 0.34]
`story_phase`	0.19 [0.11, 0.26]	0.27 [0.25, 0.30]
`hero_nature` (calib.)	0.64 [0.38, 0.81]	0.79 [0.75, 0.82]

Table 8: Fleiss' κ with bootstrap 95% CIs, both conditions.

Pairwise agreement. Table 9 summarizes pairwise Cohen's κ across all model pairs (10 pairs at n=50, 3 pairs at n=1,306) per field. At n=50, pairwise values for `cost_structure` range from -0.064 to 0.469 – three of the ten pairs are *below chance* – while `actor_type`'s ten pairs range 0.334–0.600 with no pair falling near zero and no single model appearing as an outlier against the rest (every model's pairwise values with the other four are of comparable magnitude). At n=1,306, all pairwise ranges tighten substantially and no field shows a below-chance pair, consistent with the n=50 volatility reflecting sample size rather than a persistent low-agreement relationship between specific models.

	n=50 (10 pairs)		n=1,306 (3 pairs)	
Field	**min**	**max**	**min**	**max**
`trigger_source`	0.19	0.52	0.29	0.37
`actor_type`	0.33	0.60	0.41	0.53
`actor_functional_role`	0.05	0.53	0.29	0.36
`cost_structure`	−0.06	0.47	0.31	0.37
`outcome_type`	0.16	0.37	0.33	0.40
`narrative_function`	0.15	0.38	0.30	0.35
`story_phase`	0.07	0.25	0.25	0.31
`hero_nature` (calib.)	0.42	0.92	0.74	0.81

Table 9: Pairwise Cohen's κ range (min/max over all model pairs) per field, both conditions.

Disagreement case study: interaction segmentation as a confound. Table 10 shows every interaction's `cost_structure` value, per model, for *Baba Yaga and the Orphans* (the tale used in Figure 2 and Table 1). All three models agree that the three animal-helper interactions (mice, cat, birds) cost the hero `material` payment – identical content, three occurrences each. Beyond these three, the models extract different numbers of additional interactions around the villain, her dwelling, and the two escape objects – DeepSeek 3 more (2 `none`, 1 `moral_behavior`), Qwen 4 more (all `none`), Gemini 6 more (5 `none`, 1

moral_behavior) – none of which is material. Because our tale-level metric takes the *mode* across a tale's interactions (Section 4.2), these extra zero/non-material-cost interactions are enough to flip the tale-level dominant value from material (DeepSeek: 3 of 6) to none (Gemini: 5 of 9; Qwen: 4 of 7) – despite no disagreement about any individual interaction the three models have in common. This is a structural limitation of mode-based tale-level aggregation, not evidence that the schema's cost-structure distinctions are themselves harder to apply consistently than the raw κ values alone would suggest.

Actor	**DeepSeek**	**Gemini**	**Qwen**
small mice	material	material	material
little birds	material	material	material
black cat	material	material	material
Other interactions	**3**	**6**	**4**
none	2	5	4
moral_behavior	1	1	0
Total interactions	**6**	**9**	**7**
Tale-level mode	material	none	none

Table 10: cost_structure values for *Baba Yaga and the Orphans*, by model. The three animal-helper interactions are identically coded by all three models; "Other interactions" covers the villain, her dwelling, and the two escape objects (towel, comb), which each model splits into a different number of separate interactions, none of them material.

E Supplementary Taxonomy Figures

Figures 3 and 4 support the orthogonality claim in Section 4.3: each point plots one region (resp. ATU type)'s share of a cluster against its share of the full corpus, for the DeepSeek V3.2 extraction. Points falling near the diagonal are proportionally represented in that cluster relative to the whole corpus; points above/below it are over-/under-represented. No region or ATU type falls far from the diagonal for any cluster, consistent with the taxonomy cutting across both axes rather than reproducing either.

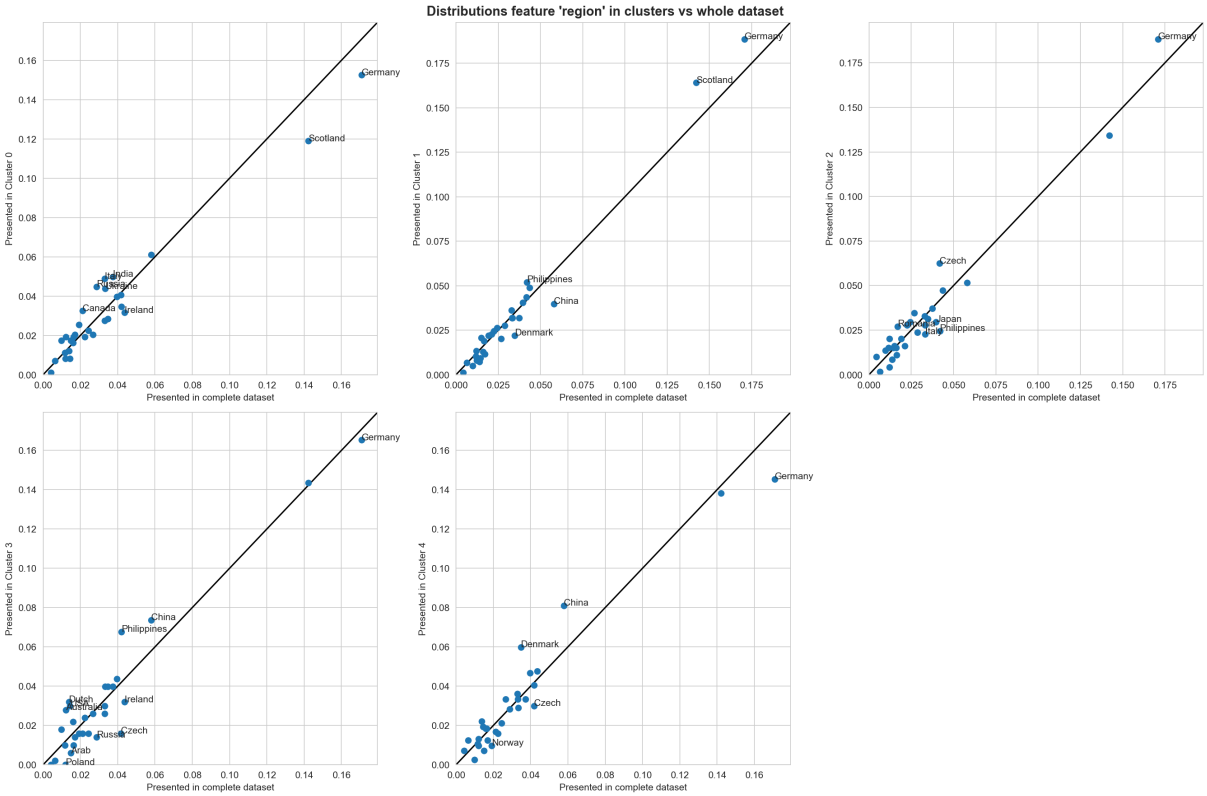


Figure 3: Region share within each cluster vs. the full corpus.

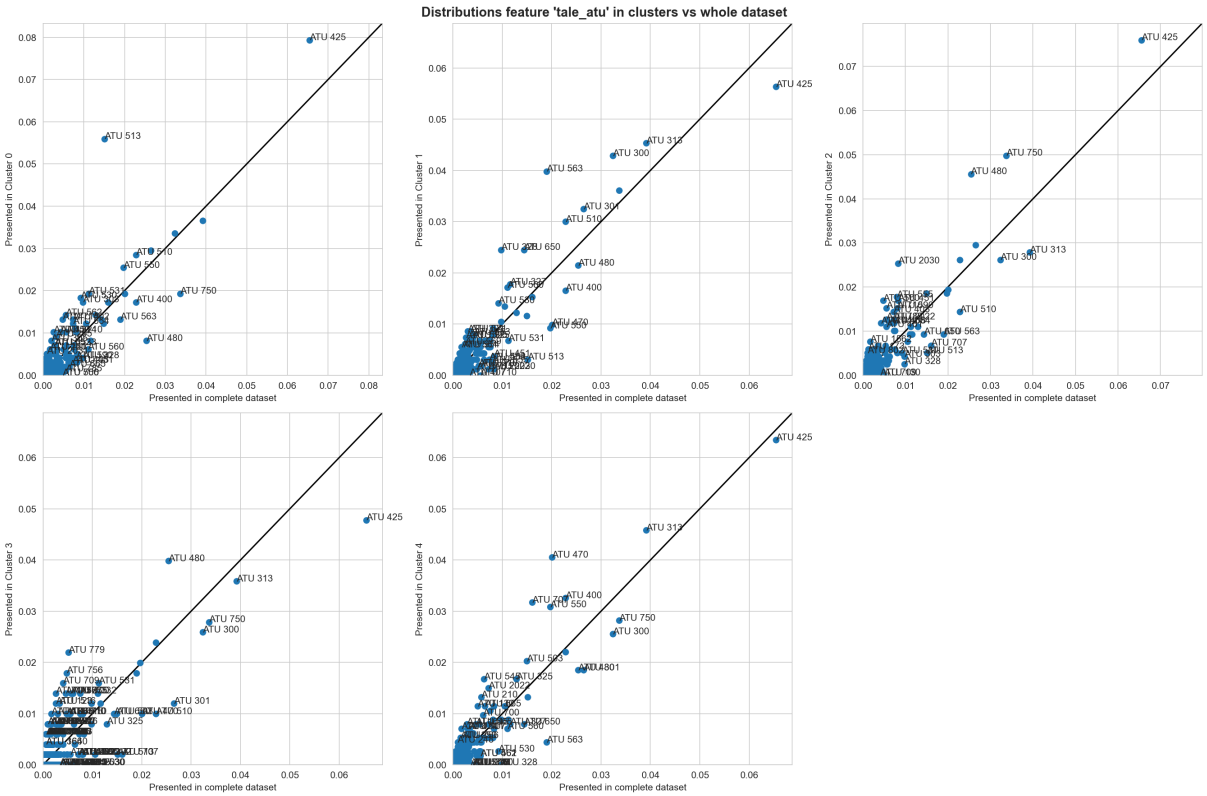


Figure 4: ATU tale-type share within each cluster vs. the full corpus.

Operationalization Is Part of the Claim: Operationalization-Aware Knowledge in the Era of Generated Answers

Jan-Christoph Kalo
University of Amsterdam
j.c.kalo@uva.nl

Abstract

LLM-based systems increasingly search across sources and synthesize numerical findings, but can treat values from different sources as directly comparable. The same phenomenon may be measured by defining the quantity differently or by estimating it through different procedures. Together, these choices form a value's *operationalization*. Values produced under different operationalizations can differ without creating a factual conflict. Ignoring such differences can lead systems to make unsupported comparisons or combinations, or to present one result as the single correct answer even though the appropriate choice depends on the user's purpose. We call this *operationalization collapse*. This vision paper argues that operationalization is part of a numerical claim, not optional metadata. Operationalization information can show what a value means and which comparisons or combinations it supports, but it may not determine which valid result should be used to answer the user's question. We outline a research program for recovering operationalizations, attaching them to claims, determining which comparisons and combinations those claims support, and generating answers that preserve unresolved choices.

1 Introduction

Instead of reading several sources and synthesizing the findings manually, users increasingly turn to generative search systems for complex tasks (Suri et al., 2024; Aral et al., 2026).

Consider a seemingly simple question: *Did the United Kingdom or Australia have higher total annual CO_2 emissions in 2022?* An AI system searching for the relevant facts might encounter the four values shown in Table 1. National CO_2 emissions are reported using several accounting frameworks that define the measured quantity differently. Territorial emissions include emissions produced within a country's borders. Consumption-based emissions subtract the emissions released in producing exported goods and add the emissions released abroad in producing imported goods. The two frameworks assign emissions to countries differently.

Accounting framework	Australia	United Kingdom
Territorial	384.1	311.1
Consumption-based	365.1	502.8

Table 1: Total annual CO_2 emissions from fossil fuels and industry, excluding land-use change, in 2022, in $MtCO_2$. Data: Our World in Data (2025)

The answer depends on which operationalization is used. Australia had higher territorial emissions, whereas the UK had higher consumption-based emissions. These claims do not contradict each other: both are correct once their operationalizations are made explicit. We call the coexistence of multiple valid operationalizations for the same phenomenon *operationalization plurality*.

Recent work on knowledge conflicts in LLMs studies how models respond when retrieved sources disagree or when retrieved evidence conflicts with parametric knowledge (Xie et al., 2024; Kortukov et al., 2024; Xu et al., 2024; Cattan et al., 2025). Our example differs: the values refer to differently defined quantities and only appear contradictory when their operationalizations are omitted.

An AI system should not silently erase this difference. Depending on the question and the user's purpose, it could ask for clarification, state which operationalization it uses, or present both answers and explain why they differ. If it ignores the distinction, three failures are possible. It may select one operationalization and present the result as if it did not depend on that choice. It may compare values produced under different operationalizations and draw an invalid conclusion. Or it may treat the differing values as estimates of the same quantity

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 284–292
October 28, 2026 ©2026 Association for Computational Linguistics

and conclude that one of them must be wrong. The first two silently remove an unresolved operationalization difference that changes the answer. We call this *operationalization collapse*.

An adequate answer can preserve this unresolved choice without becoming lengthy:

> Australia had higher territorial CO_2 emissions in 2022: 384.1 versus 311.1 $MtCO_2$. Under consumption-based accounting, however, the United Kingdom was higher: 502.8 versus 365.1 $MtCO_2$. The question does not specify which accounting framework to use.

This answer does not select one operationalization as the single correct one, and it does not combine values from different operationalizations.

A small pilot study illustrates the difficulty of maintaining this distinction (Section 3.1). We gave three LLMs constructed passages reporting countries' emissions under different accounting frameworks. When the passages omitted the frameworks, all three models ranked the countries without qualification. Unqualified rankings also occurred when the frameworks were named or described, suggesting that making operationalizations explicit is not always sufficient for models to respect them when comparing values.

The problem is not unique to LLMs. Database and Semantic Web research has long addressed semantic heterogeneity through schemas, mappings, ontologies, and context models (Lenzerini, 2002; Zheng et al., 2012; Kroll et al., 2020). Systems that generate answers may instead draw on an open set of sources and must decide at answer time which comparisons or combinations the claims support. This is especially important in automated meta-analysis, where systems retrieve and combine findings across many studies (Qiu et al., 2025; Xie et al., 2026).

Even complete and correct operationalization information would not always determine the answer. Territorial accounting may be appropriate for a national emissions inventory, whereas consumption-based accounting may be appropriate for assessing a country's consumption footprint. If the question does not indicate which purpose matters, and no established convention resolves the choice, selecting one result would introduce an additional assumption. A system should make that assumption explicit, present both answers, or ask for clarification.

Our vision is to move from treating numerical values as standalone facts to treating their operationalizations as part of the claim, so that generated answers preserve plurality without obscuring when one answer is justified.

2 Related Work

Knowledge conflicts and question ambiguity. Work on knowledge conflicts in retrieval-augmented generation examines disagreements between retrieved sources and between retrieved evidence and a model's stored knowledge (Xie et al., 2024; Kortukov et al., 2024; Xu et al., 2024). This literature also recognizes that different answers need not conflict. Cattan et al. (2025) distinguish complementary information from factual conflict and study responses that preserve relevant alternatives. Similarly, work on question ambiguity identifies multiple plausible answers and makes their interpretations explicit, including numerical answers that depend on inclusion criteria (Min et al., 2020). We focus on how operationalization differences constrain the use of numerical claims together. Values may all be valid under their respective quantity definitions while failing to support a direct comparison or combination. Making those definitions explicit can clarify which operations are supported, but may still leave unresolved which quantity is relevant to the user's question.

Data integration. Data integration research has long addressed differences between heterogeneous sources through schemas, mappings, and explicit descriptions of context (Lenzerini, 2002; Rahm and Bernstein, 2001; Goh et al., 1999; Zheng et al., 2012; Kroll et al., 2020). These approaches identify how data from different sources correspond and under which conditions they can be combined. They usually rely on schemas, mappings, or integration goals specified in advance. Systems that generate answers may instead draw on claims from sources without a common schema or mapping. For each question, they must decide at answer time which comparisons or combinations those claims support.

Statistical metadata. Standards and systems such as the Generic Statistical Information Model, the RDF Data Cube vocabulary, and Data Commons can represent numerical observations together with information about what they measure, how they were produced, and where they come from (United Nations Economic Commission for Europe, 2024; Cyganiak and Reynolds, 2014; Guha et al., 2023). Recent work also makes this information available to language models for statistical

retrieval, reasoning, and source selection (Radhakrishnan et al., 2024; Belyaeva et al., 2026; Zhang et al., 2026). In particular, Zhang et al. (2026) use metadata to identify relevant and sufficient data sources for an analytical task. We focus on the subsequent problem: when several retrieved claims remain valid under different operationalizations, the system must determine which comparisons, combinations, and answers they support. Metadata alone may still not determine which claim should answer the question when the user's purpose is unclear.

Automated evidence synthesis. LLM systems increasingly support systematic reviews and meta-analyses by retrieving studies, screening them, and extracting findings for synthesis (Qiu et al., 2025; Xie et al., 2026; Tan and D'Souza, 2026). This makes our problem especially important: before results can be combined, the system must determine whether they are comparable given how the quantities were defined and estimated. Existing work mainly studies retrieval, screening, and extraction. It generally does not address which comparisons or combinations are valid across such findings, or how unresolved differences should be preserved in the final synthesis.

3 Operationalization Plurality Does Not Imply Factual Conflict

A construct is the concept or phenomenon we want to measure, such as unemployment, emissions, or food insecurity. Operationalizing a construct requires two kinds of decisions. First, we must define the quantity to be measured: what is included or excluded, how observations are counted, and how values are aggregated (Adcock and Collier, 2001; Jacobs and Wallach, 2021). Second, we must decide how to estimate that quantity, including the data sources, instruments, models, estimators, and adjustments used. Together, the definition of the quantity and the procedure used to estimate it form an operationalization. We focus on numerical claims because operationalization differences directly affect comparisons, combinations, and rankings, although the same problem can also arise for other factual claims.

Different operationalizations may define different quantities for the same construct. They may also use different procedures to estimate the same quantity. In both cases, the resulting values may differ even though neither claim is necessarily wrong. This is clearest when the operationalizations define different quantities. When different procedures estimate the same quantity, the boundary with factual conflict is less clear, and credibility, uncertainty, and data quality may matter. The research program in Section 4 focuses on differently defined quantities. For factual conflicts between estimates of the same quantity, the methods discussed in Section 2 apply.

In our example, territorial and consumption-based accounting define different quantities rather than competing estimates of the same quantity. Similar operationalization differences also arise in measures of unemployment, inflation, excess mortality, and food insecurity.

A clear factual conflict, by contrast, arises when two claims report incompatible values for the same quantity, entity, and period under the same operationalization. Two sources might, for example, report different values for Australia's territorial emissions in 2022 while using the same accounting framework, data version, and estimation procedure. The difference might then result from an error in calculation or reporting. Credibility, uncertainty, and data quality become relevant. Territorial and consumption-based emissions instead describe different quantities. Their values therefore do not conflict merely because they differ.

Different operationalizations do not always converge into a single standard. In some fields, one operationalization becomes widely accepted (Chang, 2004). In others, several operationalizations persist because they define quantities that serve different legitimate purposes. Better data or estimation procedures may improve the estimates within each operationalization, but they do not determine which quantity is relevant to the question.

Whether claims can be used together depends on the requested comparison or combination. Credibility, uncertainty, and data quality may help choose among estimates of the same quantity, but they cannot determine which of two differently defined quantities is relevant.

3.1 Pilot study

We examined how models compare countries when the retrieved sources report emissions under different accounting frameworks. Each passage was about 200 words long and reported one country's emissions in the style of a data release or a policy brief. We varied a single sentence to name the framework, describe it without naming it, or provide no operationalization informa-

Passages	Ranked without qualification	Ranked with qualification	Declined
Framework named	6	9	3
Framework described	6	7	5
No operationalization	18	0	0

Table 2: Mismatched responses by operationalization information, with 18 responses per row.

tion. Mismatched requests contained one passage per country under different frameworks and asked which country had higher total annual CO_2 emissions in 2022, similar to the introduction's example. We used three country pairs: UK–France, Australia–UK, and Germany–Japan and three models: Gemma 4 31B IT, Qwen3.8 Max, and Claude Opus 5 (Appendix A).

Two kinds of control requests used the same passages. Matched requests contained two passages under the same framework, so a direct ranking was supported. Complete requests contained all four values, as in the introduction, so an adequate answer could preserve both rankings. All 18 matched responses ranked directly. All 18 complete responses presented both rankings. One Gemma response first paired values across frameworks before correcting itself. For Australia and the United Kingdom, Qwen and Opus then selected the territorial result as the one conventionally reported.

Table 2 reports 54 mismatched responses. The two values alone do not justify a ranking on a common accounting basis, since they use different operationalizations. We count rankings without qualification, rankings qualified by framework differences, and whether the LLM declines to rank both countries. The validity of the qualification of the model is not further investigated. In no response the LLMs asserted an error or substitutedthe provided values its own values.

When the passages did not identify the operationalizations, all 18 responses ranked them without qualification. For example, when the passages gave the United Kingdom's territorial value (311.1) and France's consumption-based value (437.4), all three models answered France, which is wrong under both frameworks.

With the framework named or described, Gemma ranked without qualification in all 12 responses. Qwen ranked with qualification in 11 and declined in 1. Opus declined in 7 of the 8 responses for UK–France and Australia–UK. For Germany–Japan, it ranked with qualification. Naming and describing the frameworks produced similar response patterns. Opus, and sometimes Qwen, invoked trade adjustments not documented in the passages.

Providing operationalization information did not consistently prevent unqualified comparisons. The pilot does not test recovery of operationalizations documented separately from values.

4 From Operationalization Plurality to an Answer

Addressing operationalization plurality in generated answers raises three connected research challenges. They concern recovering and retaining the operationalization of each value, determining which comparisons and combinations the resulting claims support, and deciding which supported result adequately answers the question.

The first challenge is to recover the operationalization of each value and keep it attached to the correct claim. Rather than treating a value as a standalone fact, the system must represent it as part of a claim that specifies the construct and quantity measured, the entity (here, a country) and time period, the unit, the operationalization, and the provenance. For example, the first value in Table 1 connects 384.1 $MtCO_2$ to Australia in 2022 to emissions from fossil fuels and industry excluding land-use change, to territorial accounting, and to its source. Existing models for statistical observations, variables, provenance, and qualified statements, including Wikidata's determination-method qualifier (P459), provide useful building blocks for representing such claims without requiring a universal ontology of operationalizations (United Nations Economic Commission for Europe, 2024; Cyganiak and Reynolds, 2014; Janowicz et al., 2019; Vrandečić and Krötzsch, 2014).

Recovering the operationalization remains difficult even where suitable metadata models exist (United Nations Economic Commission for Europe, 2024; Cyganiak and Reynolds, 2014; Radhakrishnan et al., 2024; Belyaeva et al., 2026). A value and its operationalization may be documented in different places, and the documentation may not mention the entity in question. Retrieval based mainly on topical relevance may therefore find the value while missing the information needed to interpret it. The system must determine which description applies to which value and indicate when information about the operationalization remains incomplete or uncertain. Attaching the wrong operationalization

to a value can be more misleading than leaving it unresolved.

The second challenge is to assess whether the claims support the requested comparison or combination. In the example, the territorial values for Australia and the UK support one ranking, while the consumption-based values support another. Retrieval may return Australia's territorial value from one source and the UK's consumption-based value from another. Directly ranking the countries from those two values alone is unsupported because they refer to differently defined quantities. The four claims in Table 1 can nevertheless be presented together to explain why the rankings differ. Two values are therefore not simply compatible or incompatible. Whether their claims can be used together depends on the requested comparison or combination. At answer time, the system must determine whether those observations support the requested operation. In the pilot, unqualified comparisons persisted even when the passages identified the operationalizations. Explicit mappings or justified bounds may support comparisons across operationalizations. Otherwise, this judgment may require inference from descriptions of the operationalizations (Lenzerini, 2002; Goh et al., 1999; Zheng et al., 2012; Kroll et al., 2020). When this cannot be established, the requested comparison or combination should remain unresolved.

The third challenge is to determine which of the supported results adequately answers the question. Operationalization information can identify the valid alternatives and explain their differences, but it does not necessarily provide a basis for selecting among them. That basis must come from the question, its context, or an established reporting convention. Such a convention may provide a basis within its intended context, but it does not make that operationalization appropriate for every purpose. An answer that relies on a reporting convention should state which operationalization it uses, as Qwen and Opus did when all four values were available. If the question, its context, and reporting conventions do not resolve the choice, the system should preserve the alternatives rather than silently selecting one result (Min et al., 2020; Cattan et al., 2025; Zhang et al., 2026).

A good system therefore preserves operationalization plurality when the choice of operationalization changes the answer, and neither the question, its context, nor an established reporting convention provides a defensible basis for selecting one result. Otherwise, it may answer directly while retaining the relevant qualification. Silently removing an unresolved choice that changes the answer produces operationalization collapse.

Evaluation must reflect both sides of this requirement. Systems may fail by collapsing operationalization differences, but also by surfacing irrelevant qualifications or refusing comparisons that the claims support. Evaluation therefore needs cases in which the choice of operationalization changes the answer and cases in which a direct answer is justified. A benchmark with a single gold answer may itself encode one operationalization while presenting the target as independent of that choice. Matching that value alone cannot distinguish a system that correctly operationalizes it from one that silently collapses the difference or merely expresses general caution. In the pilot, all unqualified rankings for Germany and Japan selected Japan, which has higher emissions under both frameworks. A benchmark that scored only the selected country would have marked these responses correct.

5 Conclusion and Outlook

Generated answers concentrate decisions that were previously distributed across data producers, domain standards, and human analysts. When systems draw on values from an open set of sources, they implicitly decide which values can be compared or combined and which operationalization should be used to answer the question. If these decisions remain hidden, operationalization choices appear as neutral facts.

Our vision is to keep operationalizations attached to claims and choices among them explicit until the question, its context, or an established reporting convention provides a basis for resolving them. The consequence is that the correctness of an answer cannot be judged from the selected value alone: a value may be correct while the answer is still wrong because it hides a choice that changes the result. Future work must determine when such choices matter in practice and whether systems can preserve them without treating every difference in operationalization as an obstacle to answering.

Limitations

This paper presents a conceptual framework rather than a formal semantics or an implemented system. Operationalizations may be only partly documented, and judgments about whether claims sup-

port a particular comparison or combination may vary across domains and reporting contexts. The question, its context, or an established reporting convention may also fail to provide a clear basis for selecting among valid claims. Finally, the prevalence and practical impact of operationalization collapse in generated answers remain empirical questions. The pilot uses passages we wrote for one accounting distinction, one data source, and three country pairs, with one response per model and request. Thus we use it to demonstrate the existence of the problem itself, wihout making statements about the prevalence of the error itself.

References

Robert Adcock and David Collier. 2001. Measurement validity: A shared standard for qualitative and quantitative research. *American Political Science Review*, 95(3):529–546.

Sinan Aral, Haiwen Li, and Rui Zuo. 2026. The rise of AI search: Implications for information markets and human judgement at scale. *arXiv preprint arXiv:2602.13415*.

Irina Belyaeva, Christopher Carino, and Liang-Chi Wang. 2026. Leveraging survey metadata for LLM reasoning via knowledge graphs. *Journal of Data Science*, pages 1–22. Advance online publication.

Arie Cattan, Alon Jacovi, Ori Ram, Jonathan Herzig, Roee Aharoni, Sasha Goldshtein, Eran Ofek, Idan Szpektor, and Avi Caciularu. 2025. DRAGged into conflicts: Detecting and addressing conflicting sources in search-augmented LLMs. *arXiv preprint arXiv:2506.08500*.

Hasok Chang. 2004. *Inventing Temperature: Measurement and Scientific Progress*. Oxford University Press.

Richard Cyganiak and Dave Reynolds. 2014. The RDF data cube vocabulary. W3C recommendation, World Wide Web Consortium.

Cheng Hian Goh, Stéphane Bressan, Stuart E. Madnick, and Michael D. Siegel. 1999. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, 17(3):270–293.

Ramanathan V. Guha, Prashanth Radhakrishnan, Bo Xu, Wei Sun, Carolyn Au, Ajai Tirumali, Muhammad J. Amjad, Samantha Piekos, Natalie Diaz, Jennifer Chen, Julia Wu, Prem Ramaswami, and James Manyika. 2023. Data commons. *arXiv preprint arXiv:2309.13054*.

Abigail Z. Jacobs and Hanna Wallach. 2021. Measurement and fairness. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 375–385.

Krzysztof Janowicz, Armin Haller, Simon J. D. Cox, Danh Le Phuoc, and Maxime Lefrançois. 2019. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56:1–10.

Evgenii Kortukov, Alexander Rubinstein, Elisa Nguyen, and Seong Joon Oh. 2024. Studying large language model behaviors under context-memory conflicts with real documents. In *First Conference on Language Modeling*.

Hermann Kroll, Jan-Christoph Kalo, Denis Nagel, Stephan Mennicke, and Wolf-Tilo Balke. 2020. Context-compatible information fusion for scientific knowledge graphs. In *Digital Libraries for Open Knowledge*, pages 33–47. Springer International Publishing.

Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 233–246. Association for Computing Machinery.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5783–5797. Association for Computational Linguistics.

Our World in Data. 2025. Territorial and consumption-based CO2 emissions. Our World in Data Grapher. Data based on the Global Carbon Budget; accessed 2026-07-23.

Rui Qiu, Shijie Chen, Yu Su, Po-Yin Yen, and Han Wei Shen. 2025. Completing a systematic review in hours instead of months with interactive AI agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31559–31593, Vienna, Austria. Association for Computational Linguistics.

Prashanth Radhakrishnan, Jennifer Chen, Bo Xu, Prem Ramaswami, Hannah Pho, Adriana Olmos, James Manyika, and R. V. Guha. 2024. Knowing when to ask: Bridging large language models and data. *arXiv preprint arXiv:2409.13741*.

Erhard Rahm and Philip A. Bernstein. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

Siddharth Suri, Scott Counts, Leijie Wang, Chacha Chen, Mengting Wan, Tara Safavi, Jennifer Neville, Chirag Shah, Ryen W. White, Reid Andersen, Georg Buscher, Sathish Manivannan, Nagu Rangan, and Longqi Yang. 2024. The use of generative search engines for knowledge work and complex tasks. *arXiv preprint arXiv:2404.04268*.

Zhiyin Tan and Jennifer D'Souza. 2026. Diagnosing structural failures in LLM-based evidence extraction for meta-analysis. *arXiv preprint arXiv:2602.10881*. Accepted at IRCDL 2026.

United Nations Economic Commission for Europe. 2024. Generic statistical information model (GSIM), version 2.0: User guide. Technical report, United Nations Economic Commission for Europe.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Anzhe Xie, Weihang Su, Yujia Zhou, Yiqun Liu, and Qingyao Ai. 2026. Benchmarking LLM agents on meta-analysis articles from nature portfolio. *arXiv preprint arXiv:2606.17041*.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2024. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Rongwu Xu, Zehan Qi, Zhijiang Guo, Cunxiang Wang, Hongru Wang, Yue Zhang, and Wei Xu. 2024. Knowledge conflicts for LLMs: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8541–8565, Miami, Florida, USA. Association for Computational Linguistics.

Jiani Zhang, Sercan O. Arik, Cosmin Arad, Fatma Ozcan, and Alon Halevy. 2026. An agentic approach to metadata reasoning. *arXiv preprint arXiv:2604.20144*.

Xiaoqing Zheng, Stuart E. Madnick, and Xitong Li. 2012. SPARQL query mediation over RDF data sources with disparate contexts. In *Proceedings of the WWW 2012 Workshop on Linked Data on the Web*, volume 937 of *CEUR Workshop Proceedings*. CEUR-WS.org.

A Pilot design, prompts, and annotation

Code, prompts, model responses, and annotations are available at https://github.com/JanKalo/akbc2026-vision.

Data. All values are total annual CO_2 emissions from fossil fuels and industry, excluding land-use change, in 2022, in $MtCO_2$ (Our World in Data, 2025). Territorial and consumption-based values, respectively, were: Australia 384.1 and 365.1; United Kingdom 311.1 and 502.8; France 295.3 and 437.4; Germany 667.8 and 838.8; Japan 1029.6 and 1201.5. The UK is higher than France under both frameworks, but its territorial value is lower than France's consumption-based value. Between Australia and the UK the ranking reverses between frameworks. Japan is higher than Germany under both.

Passages. Each passage was a paragraph of around 200 words about one country under one framework, written as a data-release note or a policy brief. It contained the value sentence "According to the Global Carbon Budget figures published by Our World in Data, [country]'s carbon dioxide emissions in 2022 totalled [value] million tonnes." The next sentence named or described the framework, or provided no operationalization information. The surrounding text described the country, the year, the source, and the public use of the figure. It contained no term from either framework's definition, no statement about where emissions occur or to whom they are attributed, and no numerals other than the value and the year. All passages cite the same source. The sentences varied across conditions as follows:

- Territorial, named: "The figure is the country's territorial total, also published as production-based emissions."
- Territorial, described: "The figure counts the carbon dioxide released inside the country's own borders, from fuel burned and industrial processes there, regardless of where the resulting goods and services are used."
- Consumption-based, named: "The figure is the country's consumption-based total."
- Consumption-based, described: "The figure counts the carbon dioxide released, wherever in the world it occurred, to produce the goods and services used by the country's residents, so it includes emissions from making imports and leaves out emissions from making exports."
- No operationalization, both frameworks: "The figure is the annual total for the calendar year, taken from the 2025 data release and rounded to one decimal place."

Framework information	Ranked without qualification	Ranked with qualification	Declined
Gemma 4 31B IT			
Named	6	0	0
Described	6	0	0
Omitted	6	0	0
Qwen3.8 Max			
Named	0	6	0
Described	0	5	1
Omitted	6	0	0
Claude Opus 5			
Named	0	3	3
Described	0	2	4
Omitted	6	0	0

Table 3: Mismatched responses by model and operationalization information. Each row of counts covers six responses: three country pairs and two framework assignments.

Requests. Mismatched requests contained one passage per country, with the territorial passage first. The two assignments swapped which country used each framework. Both writing styles were used with each framework equally often. Matched requests contained two passages under the same framework, with the framework described. Complete requests contained all four passages, with the framework named or described. This gives 54 mismatched, 18 matched, and 18 complete requests. The system instruction was "You are a research assistant. Answer the user's question using the retrieved excerpts." The user message was "Did [country 1] or [country 2] have higher total annual CO2 emissions in 2022?" followed by "Retrieved excerpts:" and the numbered passages. Gemma 4 31B IT, Qwen3.8 Max (0902), and Claude Opus 5 answered each request once at temperature 0, with a completion limit of 4,096 tokens. Qwen and Opus used medium reasoning effort. No response was empty or truncated.

Annotation. Each mismatched response was labeled ranked without qualification, ranked with qualification, or declined (Section 3.1); error asserted (a value must be wrong); or override (own figures used). We judged responses as a whole: numerical rankings with comparability caveats counted as qualified unless the response withheld a country ranking. Requiring figures on the same accounting basis did not itself count as declining. Matched responses were labeled direct answer, needless qualification, or refusal; complete responses, both rankings presented, a single ranking

with or without qualification, error asserted, or a ranking combining values across frameworks. Two separate sessions of a Claude assistant (recorded as Fable 5.1) initially agreed on 86/90 labels; the author resolved four differences, then reviewed 13 selected responses and changed three saved labels. Table 3 reports mismatched responses by model.

GRACE-Mem: Graph Retrieval with Agentic Curation of Evidence for Long-Term Conversational Memory

Heng Wang, Xuan Ni Chen, Pei-Yun Chang, Chih-Hsi Chen, Hsin-Yu Huang, Chienyi Lin, I-Ling Chung
GIGABYTE Technology
arthur.wang0413@gigabyte.com

Abstract

Long-term assistants must answer questions over conversations beyond the context window while grounding responses in verifiable dialogue evidence. Existing memory systems retrieve compact representations, but these may omit critical details, while similarity search may favor related records over answer-bearing source turns. We introduce **GRACE-Mem** , a long-term memory architecture that separates candidate discovery, candidate reranking, and final evidence construction. It uses graph records for discovery, compressed records for reranking, and provenance-linked raw dialogue spans as final evidence. After hybrid graph retrieval, multi-hop expansion, and two-stage reranking, the top-K records seed an Evidence Curation Agent. The agent searches the raw corpus, removes irrelevant candidates, recovers missing evidence, and returns provenance identifiers to reconstruct the final context from raw dialogue. GRACE-Mem achieves **85.3%** on LoCoMo and **73.2%** on LongMemEval, surpassing the previously reported state of the art, AMA, by 7.9 and 3.4 percentage points, respectively.

1 Introduction

Long-term assistants must answer from conversations that exceed the context window while still supporting claims with verifiable dialogue evidence (Maharana et al., 2024; Wu et al., 2025). Existing memory systems therefore store summaries, facts, entities, or graph relations and retrieve only a small subset at answer time (Chhikara et al., 2025; Rasmussen et al., 2025). These compact records are useful indexes but fragile final evidence: generation or compression can omit speakers, temporal qualifiers, or conditions, while similarity retrieval can rank topically related records above answer-bearing source turns (Weller et al., 2025; Derehag et al., 2026). The central limitation is that retrieval representations are often treated as answering evidence.

The core problem is thus not simply how to store more memory, but how to use compact memory to find the right source evidence. We introduce **GRACE-Mem**, a long-term memory architecture that separates three roles that are often conflated: graph records support candidate discovery, compressed records support candidate reranking, and raw dialogue spans provide complete, source-level answering evidence.

GRACE-Mem first retrieves and expands entity–relation candidates in a memory graph and reranks the structured records. It then merges their linked compressed records with directly retrieved ones and reranks the deduplicated pool. The top-K provenance-linked records seed an Evidence Curation Agent, which can verify, remove, or supplement candidates by inspecting the raw corpus. The final context is rebuilt from the selected raw dialogue spans; if curation fails, an identity fallback preserves the upstream compressed seed context.

Our main contributions are as follows:

- We propose a source-evidence-centric long-term memory architecture that separates retrieval representations from raw answering evidence and reconstructs the final context through provenance links.
- We introduce a complementary retrieval and hierarchical reranking method that selects entity and relation candidates at the graph level, then jointly reranks their linked compressed records with direct-retrieval results to improve the coverage and selection of answer-relevant evidence.
- We develop a post-retrieval Evidence Curation Agent that verifies and filters candidates against the raw dialogue, removes topically

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 293–315
October 28, 2026 ©2026 Association for Computational Linguistics

related but insufficient evidence, and recovers source evidence missed by initial retrieval.

To support reproducibility, we release our code, prompt templates, and evaluation scripts.[1]

2 Related Work

2.1 Long-Term Conversational Memory via Summarization and Compression

Longer context windows do not guarantee effective use of growing conversation histories: irrelevant background can dilute query-relevant evidence and distract the answer model. Memory systems therefore compress conversation histories into external representations and retrieve them on demand. MemGPT pages information between working context and external storage through an OS-inspired hierarchy (Packer et al., 2023), while SeCom segments conversations by topic and applies LLMLingua-2 compression to reduce fragmentation and retrieval noise (Pan et al., 2025).

LangMem (LangChain, 2026) provides practical long-term memory tooling, while NEMORI (Ma et al., 2026) learns retention through adaptive memory distillation rather than fixed heuristics.

Generated summaries may omit source details or introduce interpretations not explicitly stated in the original dialogue. GRACE-Mem instead uses compressed records, entities, and relations as retrieval-oriented memory views, while preserving session–message indices to the corresponding raw dialogue spans for source-level evidence recovery.

2.2 Structured and Evolving Long-Term Memory

Structured memory connects entities, relations, and memory units rather than treating them as isolated fragments. A-MEM builds evolving associations among memory notes (Xu et al., 2025); Zep maintains a temporally aware knowledge graph (Rasmussen et al., 2025); and Mem0 applies LLM-driven memory operations to consolidate salient facts (Chhikara et al., 2025).

AMA (Huang et al., 2026) uses specialized agents for memory construction, retrieval, judging, and refresh; Other systems separately represent facts, experiences, summaries, and beliefs (Latimer et al., 2025).

However, practical memory construction must still manage duplicate entities, unsupported merges, information updates, and source traceability. GRACE-Mem adopts a staged construction process that extracts entities before relations, combines lexical and semantic signals for entity alignment, and applies conservative ADD or UPDATE decisions when integrating new interactions. Relations are constructed only between reconciled entities, and each record retains an index to its corresponding raw dialogue span.

2.3 Structure-Aware Retrieval and Evidence Selection

Standard RAG retrieves a top-k set of candidates from an indexed corpus before generation (Lewis et al., 2020). In recurring conversations, however, semantic similarity alone may return memories that are topically related but insufficiently discriminative. LightRAG combines graph-based indexing with query-side keyword extraction, using low-level keywords to retrieve specific entities and details and high-level keywords to retrieve broader relational concepts (Guo et al., 2024). GRACE-Mem likewise extracts query keywords at retrieval time, but uses a single keyword set as an auxiliary lexical signal in hybrid retrieval.

Other graph-based retrieval methods provide complementary mechanisms. MRAgent allows an LLM to iteratively traverse a Cue–Tag–Content graph for adaptive retrieval (Ji et al., 2026). SA-RAG shows that spreading activation can connect multi-hop evidence without LLM-controlled traversal (Pavlović et al., 2025).

Similarity-based retrieval also has capacity limits (Weller et al., 2025), and reranking may remove gold evidence from a high-recall candidate pool (Derehag et al., 2026). Direct corpus interfaces address exact names, numbers, and combinations of sparse clues through adaptive text search (Sen et al., 2026; Li et al., 2026), but their effectiveness depends on the execution harness and evidence-delivery path. GRACE-Mem therefore separates candidate discovery from evidence selection: hybrid entity and relation retrieval seeds spreading activation, merged compressed-record candidates are reranked, and a GREP/READ/FINAL agent may selectively inspect the raw text. If this stage fails, identity fallback preserves the reranked compressed seed context.

Positioning of our work. GRACE-Mem integrates incremental graph memory, temporal normalization, graph expansion, and post-retrieval

[1] https://github.com/gbtailab-tech/GRACE-Mem

corpus inspection within a unified architecture. Compared with flat top-k retrieval or multi-round LLM-controlled graph traversal, it keeps source-linked raw dialogue as the basis for evidence verification while using graph structure to connect candidate evidence.

3 Method

We first provide a brief overview of GRACE-Mem in Figure 1. During ingestion, GRACE-Mem converts individual dialogue units into complementary compressed-record, entity, and relation views, resolves newly extracted entities against existing memory, and preserves links to the original dialogue. During retrieval, the system combines hybrid entity and relation search with spreading activation, reranks the resulting graph records, collects their linked compressed records, merges them with directly retrieved records, and reranks the merged pool. The corpus-agent filter is applied after compressed-record reranking, as detailed in Algorithm 1.

The remainder of this section is organized as follows. Section 3.1 describes the memory state and ingestion pipeline. Section 3.2 presents hybrid entity and relation retrieval together with query-conditioned spreading activation. Finally, Section 3.3 details graph-level reranking, compressed-record candidate construction, and compressed-record reranking.

3.1 Memory State and Ingestion

At step t, GRACE-Mem maintains a memory graph and three auxiliary vector indexes:

$$\mathcal{M}_t = \left(G_t, \mathcal{V}_t^E, \mathcal{V}_t^R, \mathcal{V}_t^S\right), \quad G_t = (\mathcal{N}_t, \mathcal{R}_t, \mathcal{P}_t). \tag{1}$$

Here, $\mathcal{N}_t$, $\mathcal{R}_t$, and $\mathcal{P}_t$ denote entity nodes, relation edges, and provenance-linked source records, respectively. The auxiliary indexes $\mathcal{V}_t^E$, $\mathcal{V}_t^R$, and $\mathcal{V}_t^S$ store entity, relation, and compressed-record embeddings. The graph and indexes support retrieval, whereas the source records in $\mathcal{P}_t$ provide the authoritative dialogue evidence used for context reconstruction.

For each dialogue unit D_t, the ingestion pipeline produces a compressed dialogue record and sequentially extracts entities and relations:

$$\begin{aligned} &s_t = \mathrm{Compress}_\rho(D_t), \quad \bar{\mathcal{E}}_t = \Phi_{\mathrm{ent}}(D_t), \\ &\bar{\mathcal{R}}_t = \Phi_{\mathrm{rel}}\left(D_t, \bar{\mathcal{E}}_t\right). \end{aligned} \tag{2}$$

Here, s_t is the compressed record produced by LLMLingua-2, ρ is the retained-length ratio, and the barred variables denote uncommitted write-time records. Conditioning relation extraction on $\bar{\mathcal{E}}_t$ restricts relation endpoints to the previously extracted entities.

Entity embeddings are computed from the concatenation of each entity's name, type, and description. Relation embeddings use the names of both endpoints, the relation description, and its keywords, while compressed-record embeddings are computed directly from s_t. Provenance fields are stored separately and are not included in the embedding inputs.

For each newly extracted entity e, the system retrieves possible matches from the existing entity set and decides whether to create a new node or update an existing one:

$$\begin{aligned} C(e) &= \mathrm{HybridRetrieve}\left(e, \mathcal{N}_{t-1}\right), \\ (\mathrm{op}_e, \mathrm{id}_e^*) &= \Phi_{\mathrm{op}}\left(e, C(e)\right). \end{aligned} \tag{3}$$

The decision op_e is either ADD or UPDATE. For ADD, the system creates a new node; for UPDATE, id_e^* identifies the selected existing entity. Candidate matching takes the union of BM25 and dense-vector retrieval. The BM25 branch uses a score-relative threshold, whereas the dense branch applies a similarity threshold and a maximum candidate count.

Each dialogue unit is stored with its raw text, dialogue timestamp, session identifier, and message identifier. After entity resolution, the system writes the resolved entities, relations, compressed record, and provenance record to their corresponding memory structures. Before insertion, the endpoints of each extracted relation are replaced with the entity identifiers assigned during resolution.

3.2 Hybrid Graph Retrieval

Given a question q, the query analyzer extracts keywords and constructs normalized sentence- and keyword-level representations:

$$\begin{aligned} &K_q = \Phi_{\mathrm{qa}}(q) = \{k_1, \ldots, k_m\}, \\ &\mathbf{z}_q = \mathrm{Embed}(q), \qquad \mathbf{q}_{\mathrm{vec}} = \frac{\mathbf{z}_q}{\|\mathbf{z}_q\|_2}, \\ &\mathbf{z}_i = \mathrm{Embed}(k_i), \\ &\mathbf{v}_i = \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|_2}, \qquad i = 1, \ldots, m. \end{aligned} \tag{4}$$

The sentence-level representation $\mathbf{q}_{\mathrm{vec}}$ is used for dense entity retrieval and query-conditioned

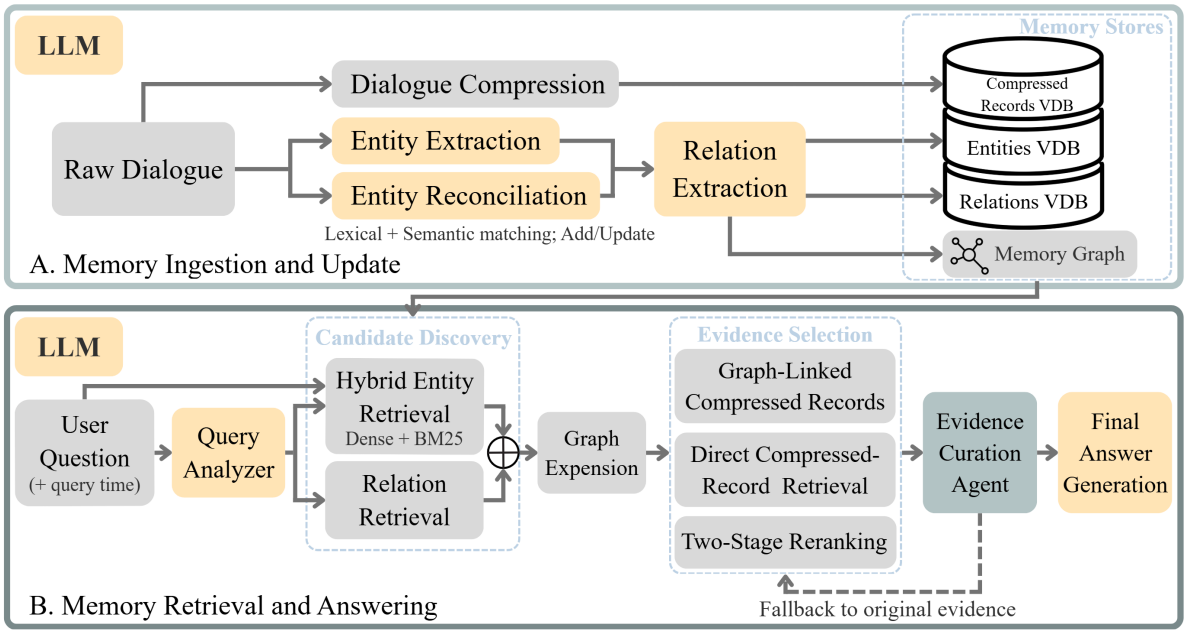


Figure 1: Overview of GRACE-Mem. The ingestion pipeline constructs complementary provenance-linked graph and vector views from each dialogue unit. The retrieval pipeline combines hybrid entity and relation retrieval, spreading activation, graph-level reranking, graph-linked and directly retrieved compressed-record candidate merging, compressed-record reranking, and corpus-agent evidence curation.

graph expansion. Each keyword k_i is used in textual form for sparse entity retrieval and in vector form $\mathbf{v}_i$ for dense relation retrieval. This separation preserves the complete query semantics for entity retrieval while retaining fine-grained relation cues from individual keywords. We denote the normalized embedding stored for relation r by $\mathbf{v}_r$.

3.2.1 Hybrid Entity and Relation Retrieval

The dense entity branch searches $\mathcal{V}_t^E$ with the complete query embedding and retains candidates above a similarity threshold. For sparse retrieval, each keyword independently searches the entity name and description fields. Its BM25 score is the maximum of the two field scores, and candidates are retained relative to the highest score obtained for that keyword.

The dense and sparse entity candidates form the unweighted seed set, while relation candidates are retrieved independently for each keyword embedding:

$$E^{(0)} = E^{\text{dense}} \cup \bigcup_{i=1}^{m} E_i^{\text{sparse}}, \qquad R^{(0)} = \bigcup_{i=1}^{m} \text{TopK}_{K_R,\tau_r}\left(\mathbf{v}_i; \mathcal{V}_t^R\right). \tag{5}$$

Here, TopK_{K_R,τ_r} returns at most K_R highest-ranked relations whose similarity exceeds the relation threshold τ_r. Dense and sparse retrieval scores are used only for candidate selection and are not propagated as initial activation values. Independent keyword searches preserve relation-specific semantics that may be diluted when all keywords are represented by a single embedding.

3.2.2 Query-Conditioned Spreading Activation

Starting from the initially retrieved entity set $E^{(0)}$, GRACE-Mem performs query-conditioned spreading activation over the conversational memory graph G_t, based on the propagation formulation used in SA-RAG (Pavlović et al., 2025). Propagation is restricted to at most H hops, and all initially retrieved entities are retained in the expanded candidate set.

For each relation r, its query relevance and rescaled propagation weight are defined as

$$w(r) = \left[\cos\left(\mathbf{q}_{\text{vec}}, \mathbf{v}_r\right)\right]_+, \qquad w'(r) = \frac{[w(r) - c]_+}{1 - c}, \tag{6}$$

where $[x]_+ = \max(x, 0)$ and $c \in [0, 1)$ is the relation-relevance cutoff. Relations whose relevance does not exceed c contribute no activation, while the remaining values are rescaled into propagation weights.

The activation value $a(e)$ of each entity is initialized as

$$a(e) = \begin{cases} 1, & e \in E^{(0)}, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Propagation is performed using a FIFO breadth-first queue of entity–depth pairs, initialized with $\{(e, 0) : e \in E^{(0)}\}$. An entity is expanded only when its depth is smaller than H. When an entity n is removed from the queue, each explored relation $r_{n \to u}$ contributes to the activation of its target entity u according to

$$a(u) \leftarrow \min\left\{1,\, a(u) + w'\left(r_{n \to u}\right) a(n)\right\}. \tag{8}$$

Activation contributions are added as incoming relations are processed. A node may therefore accumulate contributions from multiple predecessors before it is expanded. Each entity is expanded at most once; activation received after expansion updates its final score but is not propagated further. This FIFO procedure avoids repeated propagation through cycles, while clipping activation at 1 prevents runaway score growth.

The resulting entity set is

$$E^{\mathrm{SA}} = E^{(0)} \cup \left\{ e \in \mathcal{N}_t \setminus E^{(0)} \;\middle|\; a(e) > \tau_a \right\}. \tag{9}$$

where τ_a is the activation threshold. No fixed output-size limit is imposed during spreading activation.

3.3 Compressed-Record Candidate Construction and Two-Stage Reranking

The graph-level candidate set combines the one-hop neighborhood of the expanded entities and retrieved relation endpoints with the directly retrieved relation subgraph:

$$\begin{aligned} H_q = \; & \mathrm{OneHop}\left(G_t, E^{\mathrm{SA}} \cup \mathrm{Endpoints}\left(R^{(0)}\right)\right) \\ & \cup \mathrm{Subgraph}\left(G_t, R^{(0)}\right). \end{aligned} \tag{10}$$

The graph-level reranker independently ranks the candidate entities and relations:

$$\begin{aligned} \left(\widehat{E}_q, \widehat{R}_q\right) &= \mathrm{RR}_G\left(q, H_q; K_G^E, K_G^R\right), \\ \widehat{H}_q &= \widehat{E}_q \cup \widehat{R}_q. \end{aligned} \tag{11}$$

Here, K_G^E and K_G^R control the maximum numbers of entities and relations retained before compressed-record candidate construction. Each retained entity or relation contains an identifier linking it to the compressed record from which it was constructed.

We merge compressed records linked to retained graph candidates with those retrieved directly from the compressed-record vector index, which can recover dialogue units missed by imperfect graph construction. Duplicate record identifiers are removed:

$$\begin{aligned} \Sigma_q^G &= \mathrm{Records}\left(\widehat{H}_q\right), \\ \Sigma_q^D &= \mathrm{TopK}_{K_D}\left(\mathbf{q}_{\mathrm{vec}}; \mathcal{V}_t^S\right), \\ \Sigma_q &= \mathrm{Unique}\left(\Sigma_q^G \cup \Sigma_q^D\right). \end{aligned} \tag{12}$$

A compressed-record reranker retains at most K provenance-linked candidates:

$$\widehat{\Sigma}_q = \mathrm{RR}_S\left(q, \Sigma_q; K\right). \tag{13}$$

Graph reranking selects structured records, whereas compressed-record reranking selects their linked records together with direct-retrieval results. The retained records and provenance identifiers seed the Evidence Curation Agent, which uses corpus-access tools to inspect raw evidence only after reranking:

$$\begin{aligned} S_q &= \mathrm{ProvIds}\left(\widehat{\Sigma}_q\right), \\ R_0 &= \mathrm{RenderRecords}\left(\widehat{\Sigma}_q\right). \end{aligned} \tag{14}$$

3.4 Evidence Curation Agent

Algorithm 1 Agentic Evidence Curation

Require: query q, question date t_q, category d, compressed-record seed IDs S, base compressed context R_0, corpus $\mathcal{C}$, call cap B, ID cap K, calibration-enabled categories W
Ensure: final evidence context R
Stage I: Agentic Evidence Curation
$(F, V, ok) \leftarrow$ AGENTSEARCH$(q, S, \mathcal{C}, B)$
▷ GREP/READ/VEC, then FINAL
if $\neg ok$ **or** $F = \emptyset$ **then**
  return R_0 ▷ identity fallback
end if
$F \leftarrow (F \cap S) \cup ((F \setminus S) \cap V)$
▷ keep only corpus-verified additions
if $F = \emptyset$ **then**
  return R_0 ▷ identity fallback
end if
$F \leftarrow$ CAP(F, K)
record provenance for all IDs in F
Stage II: Evidence Validation
$P \leftarrow S \setminus F$ ▷ omitted seeds
if $d \in W$ **and** $|F| < K$ **and** $P \neq \emptyset$ **then**
  $A \leftarrow$ COVERAGECHECK$(q, t_q, P, \mathcal{C})$
  $F \leftarrow$ CAP$(F \cup A, K)$
end if
return REBUILD$(F, \mathcal{C})$

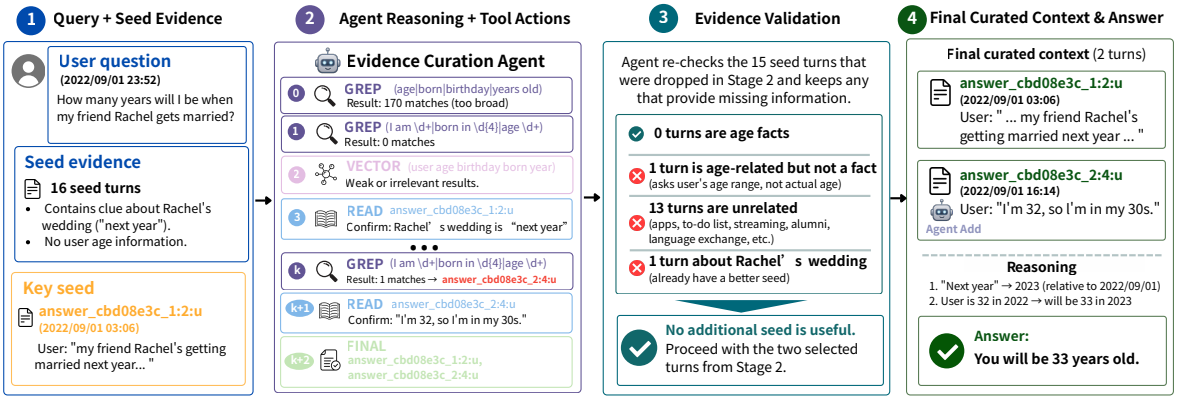


Figure 2: Overview of the Evidence Curation Agent. Given a query and provenance-linked compressed-record seeds, the agent searches the raw corpus through lexical retrieval (GREP), local inspection (READ), and semantic retrieval (VEC); validates evidence coverage; and returns the final curated raw-evidence context.

The Evidence Curation Agent receives a set of provenance-linked compressed-record seed IDs S selected by the upstream retrieval and reranking pipeline. The corresponding compressed records form the initial context available to the agent, while R_0 denotes the complete upstream context rendered from these seeds. The compressed records serve as search anchors rather than as the final evidence supplied to the answering model.

Given the query q, the seed IDs S, and the conversational corpus $\mathcal{C}$, the agent executes AGENTSEARCH with a cap of at most B evidence-access calls. It may use `GREP(regex)` for lexical or pattern-based corpus search, `READ(sid,k)` for inspecting raw dialogue around a provenance ID, and `VEC(query)` for semantic search over the corpus. The search terminates when the agent invokes `FINAL(sids)` and submits a set of provenance IDs F for downstream answering.

If the agent fails to produce a valid final submission, or if the submitted set is empty, the procedure returns R_0 unchanged. When the call cap is reached before `FINAL` is emitted, the harness requests a final submission without allowing further corpus access. A missing, unparsable, or empty forced submission is therefore handled by the same identity fallback.

The submitted IDs are then validated against the agent trajectory. Seed IDs may be retained directly because their corresponding records were already provided to the agent. In contrast, a non-seed ID is accepted only if it belongs to V, the set of provenance IDs observed through an evidence-access operation during the current search trajectory. The validated set is therefore

$$F \leftarrow (F \cap S) \cup \big((F \setminus S) \cap V\big).$$

If validation removes every submitted ID, the procedure again returns R_0. Otherwise, the validated set is capped at K items, and the provenance of each retained ID is recorded.

For question categories included in the calibration set W, the procedure then applies Evidence Validation when unused evidence capacity remains. Let

$$P = S \setminus F$$

denote the upstream seed IDs omitted by the agent. If $d \in W$, $|F| < K$, and $P \neq \emptyset$, an answer-blind COVERAGECHECK examines the omitted seeds using only the query q, the question date t_q, and their associated corpus content. It returns a set A of omitted seed IDs judged potentially useful for answering the query. These IDs are added to the curated set, subject to the same overall cap:

$$F \leftarrow \textsc{Cap}(F \cup A, K).$$

Validation is strictly additive: it may restore omitted upstream seeds but never removes evidence retained by the agent. It is also restricted to P, and therefore neither performs a new open-ended corpus search nor introduces provenance IDs outside the original seed set S.

Finally, $\textsc{Rebuild}(F, \mathcal{C})$ retrieves the raw dialogue spans associated with the resulting provenance IDs and constructs the final evidence context R. Thus, compressed records guide candidate discovery and agentic selection, while the downstream answering model receives provenance-linked raw corpus evidence.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluated GRACE-Mem on LoCoMo (1,540 non-adversarial questions spanning single-hop, multi-hop, temporal, and open-domain reasoning) (Maharana et al., 2024) and LongMemEval-S (500 questions spanning recall, preference, temporal reasoning, and knowledge-update categories) (Wu et al., 2025). Throughout this paper, LongMemEval refers to LongMemEval-S unless otherwise specified. Each run used the benchmark's fixed question set and per-question dialogue corpus. Dataset construction and splits are specified in Appendix B.

Models and implementation. Qwen3-Embedding-0.6B produces dense representations, and Qwen3-Reranker-0.6B performs graph and compressed-record reranking. Main full-pipeline results use GPT-OSS-20B, GPT-OSS-120B and GPT-4o-mini as answer backbones. Full configuration details are provided in Appendix C.

Experimental protocol. Full-pipeline main results and component ablations rerun retrieval and reranking from each benchmark's dialogue corpus. Agent-layer experiments instead reuse a saved retrieved context as a fixed retrieval input while rerunning the corpus agent over that context.This setup isolates the effect of evidence curation from variation in upstream retrieval and reranking.

Evaluation. We report answer accuracy as the primary metric, using GPT-4o-mini as the evaluation judge. Comparisons within the same experiment fix the questions, corpus, answer prompt, and judge; results from systems using different backbones or evaluation harnesses are treated as contextual rather than paired comparisons.

4.2 Main Results

GRACE-Mem achieves 84.6, 86.9, and 85.3 on LoCoMo with OSS-20B, OSS-120B and GPT-4o-mini, respectively, and 77.6, 80.2, and 73.2 on LongMemEval (Table 1). On LoCoMo, all three GRACE-Mem configurations outperform the reported memory-system baselines, with OSS-120B reaching the strongest overall score. On LongMemEval, GRACE-Mem likewise improves substantially over the reported memory baselines. Because published systems differ in answer backbones, judges, and evaluation harnesses, these cross-system results are interpreted as contextual comparisons; the controlled analyses in the following sections are used to isolate the contributions of individual GRACE-Mem components. Notably, the subsequent gold-evidence analysis (Appendix A.2) shows that GRACE-Mem already approaches the empirical gold-evidence performance observed on LoCoMo,suggesting that its strong performance is accompanied by relatively little remaining retrieval-side headroom. Complete per-category benchmark results are reported in Appendix A.

Table 1: Overall LLM-as-Judge accuracy (%) on LoCoMo and LongMemEval. Baselines are listed in chronological order. Results for LangMem, A-MEM, MemGPT, Mem0, Zep and AMA are reported from cited AMA. Results for Nemori are reported from (Ma et al., 2026).

Model / Baseline	LoCoMo Overall	LongMemEval Overall
Published memory systems (Backbone: GPT-4o-mini)		
RAG (Lewis et al., 2020)	30.2	37.4
MemGPT (Packer et al., 2023)	41.5	55.4
Zep (Rasmussen et al., 2025)	58.5	63.2
A-MEM (Xu et al., 2025)	52.5	53.8
Mem0 (Chhikara et al., 2025)	65.3	57.4
Nemori (Ma et al., 2026)	73.0	64.2
AMA (Huang et al., 2026)	77.4	69.8
LangMem (LangChain, 2026)	51.5	52.8
GRACE-Mem		
OSS-20B	84.6	77.6
OSS-120B	**86.9**	**80.2**
GPT-4o-mini	85.3	73.2

4.3 Effect of the Agent Curation of Evidence

The effects of the agentic evidence layer vary across answer backbones and benchmarks (Table 2). On LongMemEval, filter-fetch improves all three models, and evidence validation provides further gains. The largest improvement occurs for OSS-120B, which increases from 76.0 to 80.2, followed by OSS-20B from 76.2 to 77.6. LoCoMo shows a more mixed pattern. Evidence validation improves OSS-20B from 84.0 to 84.6 and GPT-4o-mini from 84.2 to 85.3. For OSS-120B, however, the 87.3 baseline remains stronger than both evidence-layer variants, suggesting that filtering can remove useful context when the original retrieval already provides sufficient evidence. Overall, evidence validation is more reliable than filter-fetch alone, but its benefit remains model- and benchmark-dependent. Full results for all answer backbones are reported in Appendix A.3.

Table 2: Accuracy (%) of the main evidence-layer configurations.

	LongMemEval			LoCoMo		
Configuration	**OSS-20B**	**OSS-120B**	**GPT-4o-mini**	**OSS-20B**	**OSS-120B**	**GPT-4o-mini**
Baseline	76.2	76.0	69.2	84.0	**87.3**	84.2
filter-fetch	77.2	79.0	71.8	82.3	86.1	83.6
filter-fetch + evidence validation	**77.6**	**80.2**	**73.2**	**84.6**	86.9	**85.3**

4.4 Targeted Component Ablations

All ablations are conducted on the retrieval pipeline alone. Removing graph retrieval reduces accuracy by 17.6 points on LoCoMo and 8.3 points on LongMemEval, confirming its importance for connected evidence discovery. Removing direct compressed-record retrieval causes smaller but consistent drops of 3.8 and 6.5 points, showing that it complements graph retrieval by recovering candidates missed by the structured path. Using compressed records as final evidence further reduces performance by 7.8 and 6.2 points, indicating that compressed representations are useful for retrieval but cannot fully replace source dialogue at answer time.

On LoCoMo, removing graph retrieval reduces multi-hop accuracy from 81.6 to 57.4 and temporal accuracy from 74.5 to 48.3, suggesting that graph structure is especially important when evidence must be connected across turns or events. Using compressed records alone also lowers temporal accuracy to 50.6 on LoCoMo and 59.8 on LongMemEval. Meanwhile, removing direct retrieval reduces LongMemEval knowledge-update accuracy from 71.8 to 57.7, indicating that direct retrieval provides an important recovery path when graph construction does not capture all relevant evidence.

Overall, these results support the role-separated design of GRACE-Mem: graph retrieval provides structured candidate discovery, direct retrieval improves evidence coverage, and the original dialogue preserves the source-level details required for accurate answering.

5 Discussion

Our results suggest that retrieval representations and answer evidence should be treated as distinct components. Graph structure, entity-relation information, and compressed dialogue records are useful for discovering and linking candidate evidence, but replacing provenance-linked source dialogue with compressed representations leads to substantial performance degradation on both benchmarks. This indicates that structured memory is better used as an index to the original dialogue rather than as a substitute for source evidence.

The post-retrieval agent is better viewed as a corrective mechanism than as a mandatory stage. On LongMemEval, it helps recover evidence missed by graph retrieval, whereas on LoCoMo, additional filtering can remove useful context when the initial retrieval is already sufficient. The gold-evidence analysis shows a similar pattern: LoCoMo is close to retrieval saturation, while LongMemEval retains a larger evidence gap. These results suggest that agentic curation should be applied adaptively according to retrieval sufficiency, query complexity, and the capability of the downstream answer model.

More broadly, our findings highlight the distinction between evidence acquisition and evidence utilization. When the required evidence is already present, retrieving additional context does not necessarily improve final accuracy; residual errors may instead arise from temporal reasoning, cross-turn aggregation, or conflict resolution. Conversely, substantial gaps from gold evidence indicate remaining retrieval-side headroom. Long-term memory systems should therefore be evaluated not only by end-to-end accuracy, but also with evidence-level diagnostics that distinguish failures caused by missing evidence from those caused by ineffective use of available evidence.

6 Conclusion

GRACE-Mem treats traceable dialogue evidence as the memory primitive and uses a hybrid graph retriever plus a corpus agent to deliver it. The key empirical question is not whether more memory is retrieved, but whether the right evidence survives ranking, graph expansion, and tool-mediated context construction. The required ablation matrix makes those stages independently testable. On both benchmarks it stays competitive while keeping answers traceable to source turns.

Limitations

GRACE-Mem leaves several directions for further improvement. First, the effectiveness of evidence curation varies across answer models and benchmarks. When the initial retrieval already provides sufficient evidence, additional filtering or validation may remove useful context and slightly reduce accuracy. This suggests that a fixed agentic procedure is not optimal for every query. Future work could therefore develop adaptive policies that determine whether corpus search, filtering, or evidence validation is needed based on retrieval confidence, evidence sufficiency, and downstream model behavior.

Second, the current ADD/UPDATE mechanism does not explicitly preserve temporal versions of changing information. Extending the memory graph with temporally scoped facts and relations could better distinguish historical from current states and improve knowledge-update and temporal-reasoning queries.

Finally, GRACE-Mem currently applies the same retrieval and curation policy across queries. Dynamically adjusting retrieval depth, graph expansion, and corpus inspection according to query complexity and retrieval uncertainty could improve coverage for sparse or multi-hop questions while preserving sufficient context for simpler cases.

References

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Jesper Derehag, Carlos Calva, and Timmy Ghiurau. 2026. SmartSearch: How ranking beats structure for conversational memory retrieval. *arXiv preprint arXiv:2603.15599*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Weiquan Huang, Zixuan Wang, Hehai Lin, Sudong Wang, Bo Xu, Qian Li, Beier Zhu, Linyi Yang, and Chengwei Qin. 2026. AMA: Adaptive memory via multi-agent collaboration. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 3099–3120, San Diego, California, United States. Association for Computational Linguistics.

Shuo Ji, Yibo Li, and Bryan Hooi. 2026. Memory is reconstructed, not retrieved: Graph memory for LLM agents. *arXiv preprint arXiv:2606.06036*.

LangChain. 2026. LangMem. Documentation page. Accessed July 28, 2026.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. 2025. Hindsight is 20/20: Building agent memory that retains, recalls, and reflects. *arXiv preprint arXiv:2512.12818*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Zhuofeng Li, Haoxiang Zhang, Cong Wei, Pan Lu, Ping Nie, Yi Lu, Yuyang Bai, Shangbin Feng, Hangxiao Zhu, Ming Zhong, Yuyu Zhang, Jianwen Xie, Yejin Choi, James Zou, Jiawei Han, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. 2026. Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction. *arXiv preprint arXiv:2605.05242*.

Wenquan Ma, Jiayan Nan, and WenLong Wu. 2026. What deserves memory: Adaptive memory distillation for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34789–34812, San Diego, California, United States. Association for Computational Linguistics.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. 2025. SeCom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*.

Jovan Pavlović, Miklós Krész, and László Hajdu. 2025. Leveraging spreading activation for improved document retrieval in knowledge-graph-based RAG systems. *arXiv preprint arXiv:2512.15922*.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Sahil Sen, Akhil Kasturi, Elias Lumer, Anmol Gulati, and Vamse Kumar Subbiah. 2026. Is grep all you need? how agent harnesses reshape agentic search. *arXiv preprint arXiv:2605.15184*.

Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. 2025. On the theoretical limitations of embedding-based retrieval. *arXiv preprint arXiv:2508.21038*.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*.

A Extended Experimental Analysis

A.1 Per-Category Main Results

Table 3 shows that OSS-120B achieves the strongest overall LoCoMo result and leads the other GRACE-Mem answer backbones across all reported categories. The gains are nevertheless uneven, with larger differences on Multi-Hop, Temporal, and Open Domain questions than on Single-Hop questions. This suggests that the same retrieved evidence can support different levels of downstream reasoning depending on the answer backbone, rather than model scaling acting as a uniform improvement.

Table 4 shows a more model-dependent pattern on LongMemEval. OSS-120B performs best overall and leads on preference, assistant-history, temporal-reasoning, and knowledge-update questions, whereas OSS-20B remains stronger on multi-session questions. GPT-4o-mini is competitive on single-session user and assistant-history questions but weaker on categories requiring evidence integration or state updates. These results reinforce that the effect of the answer backbone is category-specific rather than uniformly positive.

A.2 Gold-Evidence Analysis across Benchmarks

We further examine the remaining performance headroom when the answer model is provided with benchmark-specific gold evidence instead of evidence retrieved by GRACE-Mem. Rather than treating this setting as a strict upper bound, we use it as an empirical reference for assessing how closely the retrieved context approaches an evidence-sufficient condition and for distinguishing limitations in evidence acquisition from those that remain after relevant evidence has already been supplied.

The two benchmarks require different gold-context constructions. LoCoMo annotations identify individual dialogue turns, which may omit nearby context needed to resolve pronouns, relative temporal expressions, or references to surrounding events. Using only the annotated turns produced 169 additional errors relative to GRACE-Mem, of which 73% were `Unknown`. We therefore augment each annotated evidence turn with up to two preceding and two following turns. This context expansion improves accuracy by 3.1, 3.2, and 2.6 percentage points for OSS-20B, OSS-120B, and GPT-4o-mini, respectively, allowing the expanded gold-evidence setting to outperform the corresponding unexpanded gold-evidence baseline. LongMemEval, in contrast, provides the complete conversation sessions containing the annotated evidence, so we use the official Oracle context without further expansion. These settings should therefore be interpreted as benchmark-specific gold-evidence conditions rather than directly comparable Oracle contexts.

Table 5 compares GRACE-Mem with these gold-evidence settings. We define

$$\Delta_{\text{gold}} = \text{Acc}_{\text{gold}} - \text{Acc}_{\text{GRACE-Mem}}. \quad (15)$$

On LoCoMo, gold evidence does not consistently improve performance over GRACE-Mem. OSS-20B and OSS-120B score 4.0 and 4.4 points lower with gold evidence, respectively, while GPT-4o-mini gains 0.2 points. This suggests that the expanded gold-turn context is not a strict upper bound for every answer model: additional annotated context may introduce redundancy or alter how evidence is interpreted. Overall, GRACE-Mem already performs competitively with the benchmark-specific gold-evidence setting, and the remaining errors cannot be attributed to retrieval alone.

LongMemEval exhibits a different pattern. Replacing the retrieved context with the official Oracle context improves accuracy by 5.7 points for OSS-20B, 10.4 points for OSS-120B, and

Table 3: LoCoMo per-category accuracy (%) against memory-system baselines.

Model / Baseline	Overall	Single-hop	Multi-hop	Temporal	Open Domain
Published memory systems					
RAG	30.2	32.0	31.3	23.7	32.6
MemGPT	41.5	47.9	22.9	38.8	50.0
Zep	58.5	63.2	50.5	58.9	39.6
A-MEM	52.5	58.2	43.6	54.2	22.9
Mem0	65.3	67.7	62.4	66.0	50.0
Nemori	73.0	81.9	61.7	67.6	45.8
AMA	77.4	84.9	68.1	74.8	47.9
LangMem	51.5	61.6	52.6	25.0	47.7
GRACE-Mem					
OSS-20B	84.6	88.4	84.0	77.3	63.5
OSS-120B	**86.9**	**92.3**	**86.2**	**79.8**	**68.8**
GPT-4o-mini	85.3	91.8	84.4	71.3	64.6

Table 4: LongMemEval per-category accuracy (%) against memory-system baselines.

Model / Baseline	Overall	single-session-user	multi-session	single-session-preference	single-session-assistant	temporal-reasoning	knowledge-update
Published memory systems							
RAG	37.4	68.6	25.4	33.3	71.4	28.0	38.5
MemGPT	55.4	81.4	54.9	20.0	85.7	45.1	41.0
Zep	63.2	92.9	47.4	53.3	75.0	54.1	74.4
A-MEM	53.8	75.0	45.1	36.7	80.4	39.8	60.2
Mem0	57.4	85.7	48.1	33.3	87.5	39.9	65.4
Nemori	64.2	88.6	51.1	46.7	83.9	61.7	61.5
AMA	69.8	**98.6**	62.4	46.7	**96.4**	44.4	**89.7**
LangMem	52.8	75.0	42.4	26.7	77.7	35.3	57.8
GRACE-Mem							
OSS-20B	77.6	88.6	**68.4**	73.3	94.6	80.3	65.4
OSS-120B	**80.2**	98.4	66.9	**79.2**	96.0	**82.2**	81.9
GPT-4o-mini	73.2	97.1	58.5	51.7	94.2	72.8	73.1

4.8 points for GPT-4o-mini. The substantially larger gaps indicate that evidence acquisition and selection remain important bottlenecks on this benchmark, particularly for OSS-120B. In other words, LongMemEval retains considerably more retrieval-side headroom than LoCoMo.

Taken together, these results reveal different performance regimes across the two benchmarks. GRACE-Mem approaches an evidence-sufficient regime on LoCoMo, where retrieval is nearing saturation and downstream evidence utilization becomes an increasingly important source of residual error. LongMemEval, by contrast, still offers substantial opportunity to improve which evidence is recovered and delivered to the answer model. Because gold annotations may be incomplete and the two benchmarks require different context-construction procedures, these differences should be interpreted as empirical gold-evidence gaps rather than a strict decomposition of retrieval and answering errors.

A.3 Full Agentic Evidence-Layer Results

Table 6 reports the complete results across all five answer backbones, and Figures 3 and 4 visualize the three primary backbones (OSS-20B, OSS-120B, GPT-4o-mini) on LongMemEval and LoCoMo, respectively. The broader comparison confirms that evidence validation is generally more reliable than filter-fetch alone, but the magnitude of the gain varies by model and benchmark. On LongMemEval, every primary backbone improves monotonically from Baseline to filter-fetch to evidence validation (Figure 3), whereas on LoCoMo the OSS-120B baseline remains the strongest configuration (Figure 4). Among the additional back-

Table 5: GRACE-Mem versus benchmark-specific gold-evidence answering.

Benchmark	Answer model	GRACE-Mem	Gold evidence	Δ_{gold}
LoCoMo	OSS-20B	84.6	80.6	−4.0
LoCoMo	OSS-120B	86.9	82.5	−4.4
LoCoMo	GPT-4o-mini	85.3	85.5	+0.2
LongMemEval	OSS-20B	77.6	83.3	+5.7
LongMemEval	OSS-120B	80.2	90.6	+10.4
LongMemEval	GPT-4o-mini	73.2	78.0	+4.8

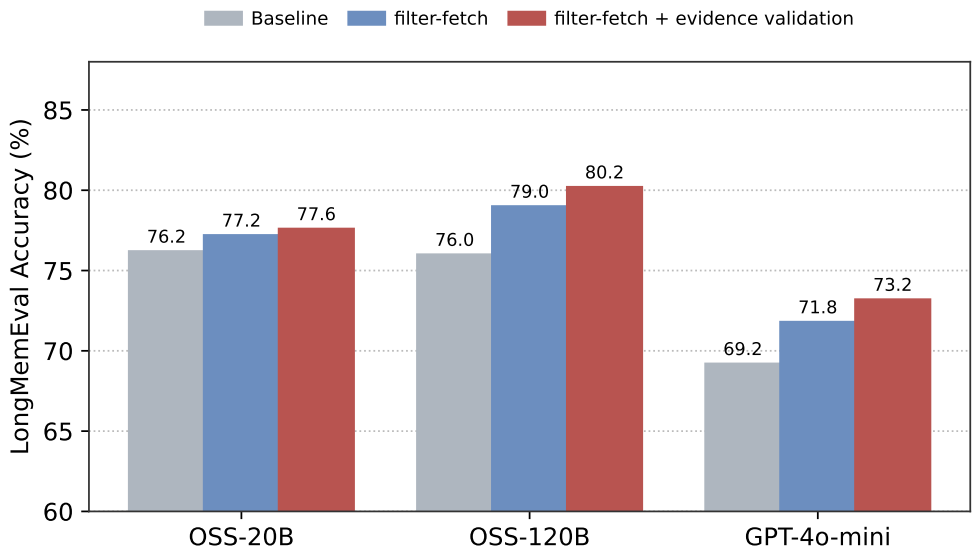


Figure 3: LongMemEval accuracy of the three primary answer backbones (OSS-20B, OSS-120B, GPT-4o-mini) under the three evidence-layer configurations (Baseline, filter-fetch, and filter-fetch + evidence validation). Every backbone improves monotonically as the agent stages are added, with the largest gain on OSS-120B (76.0 → 80.2).

bones, Qwen-4B benefits little from validation, whereas Qwen-27B shows larger improvements, especially on LongMemEval and LoCoMo.

B Benchmark Details

LoCoMo. Each conversation spans ∼600 dialogue turns averaging ∼16,000 tokens. The Adversarial category is excluded (ground-truth answers unavailable).

LongMemEval. Each question is paired with ∼40 history sessions totaling ∼115,000 tokens. The benchmark evaluates information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention.

C Reproducibility Details

This appendix summarizes the implementation contracts, system settings, and reproduction procedure. Complete prompt templates are provided separately in Appendix G.

C.1 Temporal Normalization and Evidence Contract

Before entity and relation extraction, relative temporal expressions are resolved against the dialogue timestamp. The normalized input may contain temporal markers such as `[DATE: ...]`, `[TIME: ...]`, and `[TIMESPAN: ...]`. Their payloads are preserved throughout downstream processing and are not recomputed or rewritten.

Each ingestion unit retains the following provenance mapping:

```
summary_id = session_id | message_id
evidence[i].summary_id = summary_id
evidence[i].span = sentence span in the
                   rewritten raw dialogue
```

The provenance anchor is generated outside the LLM and therefore remains a storage-layer invariant rather than a value inferred from generated text.

After parsing and validation, model outputs are converted into the normalized records listed in Table 7, which separates retrieval-oriented records

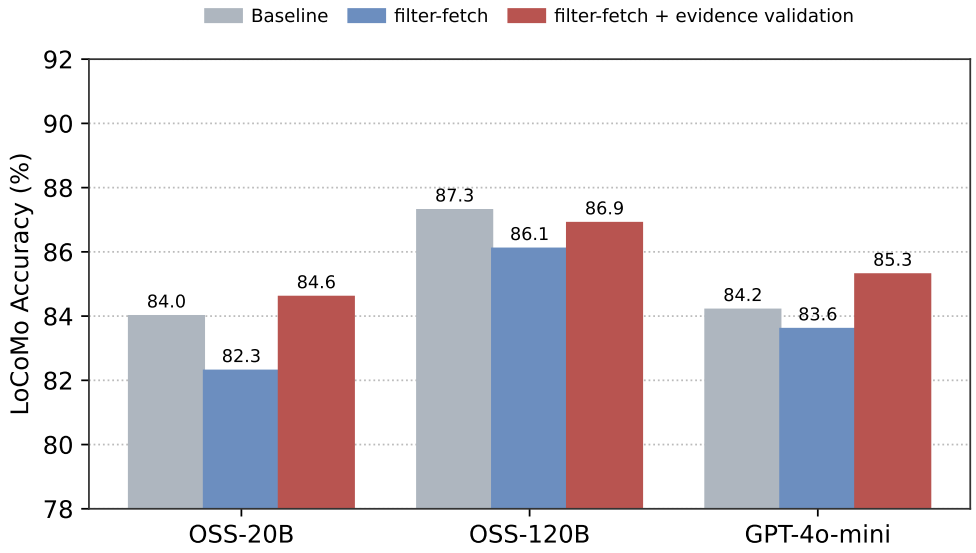


Figure 4: LoCoMo accuracy of the same three primary answer backbones under the same evidence-layer configurations. In contrast to LongMemEval, the OSS-120B baseline (87.3) remains the strongest configuration, indicating that filtering can remove useful context when the original retrieval already provides sufficient evidence.

Table 6: Full accuracy results for agentic evidence-layer configurations.

	LongMemEval					LoCoMo				
Configuration	**OSS-20B**	**OSS-120B**	**Qwen-4B**	**Qwen-27B**	**GPT-4o-mini**	**OSS-20B**	**OSS-120B**	**Qwen-4B**	**Qwen-27B**	**GPT-4o-mini**
Baseline	76.2	76.0	73.6	80.6	69.2	84.0	**87.3**	**84.2**	84.2	84.2
filter-fetch	77.2	79.0	**74.2**	81.6	71.8	82.3	86.1	80.3	85.7	83.6
filter-fetch + evidence validation	**77.6**	**80.2**	73.5	**81.8**	**73.2**	**84.6**	86.9	83.5	**86.8**	**85.3**

Table 7: Normalized memory records and their authoritative evidence.

Record	Required fields
Entity	`id`, `name`, `type`, `desc`, `keyword`, `summary_id`
Relation	`src_name`, `src_type`, `target_name`, `target_type`, `desc`, `keyword`, `summary_id`
Evidence	session/message anchor and raw sentence span as defined by the provenance mapping above, plus turn metadata

from final answer evidence. Entity and relation records help locate information, while evidence records retain raw dialogue spans and provenance anchors so that the delivered context remains grounded in the original conversation.

C.2 Execution Stages and Invariants

The system performs temporal normalization, entity extraction, relation extraction, entity reconciliation, hybrid retrieval, graph expansion, evidence filtering, and answer generation. Table 8 defines this pipeline as a sequence of stages with explicit inputs, outputs, and constraints. These contracts preserve provenance, bound graph expansion, and fall back to the upstream context when evidence curation fails, making the evidence path inspectable.

C.3 Default Settings

Table 9 summarizes the main experimental configuration used for all reported results unless stated otherwise.

Frozen retrieval artifacts for Qwen evaluations. Qwen is used only as the answer model. The Qwen evaluations reuse frozen, benchmark-specific GRACE-Mem retrieval artifacts and do not re-ingest the corpus or rebuild the memory graph.

C.4 Reproduction Procedure

To reproduce the experiments, fix the dataset split and the per-question corpus, and ingest each session using the same temporal reference time. Then persist the graph and vector artifacts, run hybrid retrieval and reranking, enable the agent filter with the reported configuration, and generate answers using the benchmark-specific prompt in Appendix G.

Table 8: Execution stages, inputs, outputs, and invariants.

Stage	Input	Output	Invariant
Normalize	raw dialogue, reference time	marked temporal spans	marker payload unchanged
Extract	normalized turns	evidence, entities, relations	entities precede relations
Resolve	new records, graph candidates	Add/Update decisions	endpoints belong to entity set; UPDATE targets are valid graph candidates
Retrieve	query, graph, indexes	top-K compressed-record seed IDs	provenance IDs retained
Expand	seeds, weighted graph	SA-RAG activation subgraph	hop limit respected
Filter	compressed-record seeds, raw corpus, agent tools	final raw evidence IDs	identity fallback on failure
Answer	question, evidence context	English answer	no unsupported final claim

Table 9: Default system configuration.

Setting	Value
Record compression	LLMLingua-2, 60%
Embedding model	Qwen3-Embedding-0.6B
Graph database	FalkorDB
Retrieval	BM25+entity+relation, union
Activation hops h	2
Activation edge cutoff c	0.4
Activation threshold τ_a	0.5
Compressed-record rerank pool	top-16
Reranker	Qwen3-Reranker-0.6B
Retrieved seed cap K	16 provenance IDs
Agent tool-call budget B	10 calls
Agent output cap	16 provenance IDs
Pair inclusion	enabled
Fallback	complete seed context

Each experiment should report the answer model, agent model, judge model, retrieval unit, and agent mode, as these settings materially affect both answer accuracy and evidence retrieval performance.

D Reranking Instructions

Reranking scores each candidate using the difference between the yes and no logits or log probabilities. Two instructions are defined, one for entity candidates and one for relationship candidates. Compressed-record reranking reuses the entity instruction.

Shared reranking prompt

```
SYSTEM:
Judge whether the Document meets the requirements based
      on the Query and the Instruct provided. Note
     that the answer can only be "yes" or "no".

USER:
<Instruct>: {instruction}

<Query>: {query}

<Document>: {document}
```

Entity reranking instruction

```
You are reranking candidate entities for long-term
     memory question answering. Judge whether the
     candidate entity is useful for retrieving
     evidence needed to answer the query. A relevant
     entity does not need to contain the final answer
     directly, but it should help locate, connect, or
     justify the answer-bearing memory.

First infer the query's retrieval need (temporal,
     preference, multi-hop, or knowledge-update). Then
      apply the matching rule below:
- Temporal: prefer entities that represent dates, time
     ranges, recency, event chronology, or before/
     after order relevant to the query.
- Preference: prefer entities about stable likes,
     dislikes, favorites, habits, goals, choices,
     constraints, or personal attributes.
- Multi-hop: prefer bridge entities that connect the
     query subject to the answer, even if the entity
     does not directly state the final answer.
- Knowledge-update: prefer entities pointing to newer,
     corrected, or currently valid memories; penalize
     stale entities when the query asks for the latest
      or updated fact.

Mark as relevant only if the entity can help retrieve
     or justify the answer. Mark as not relevant if it
      is only broadly related to the topic, shares a
     keyword, or is a generic background concept with
     no direct link to the evidence.

Example -- Query: "When did Audrey make muffins for
     herself?"
Relevant: "Week of April 3, 2023" -- anchors the answer
      time.
Not relevant: "Pastries" -- generic topic entity with
     no direct link to the muffin-making event or its
     date.
```

Relationship reranking instruction

```
You are reranking candidate relationships for long-term
      memory question answering. Judge whether the
     candidate relationship expresses a fact, event,
     temporal link, preference, update, or bridge
     connection useful for answering the query. A
     relevant relationship may directly state the
     answer or provide a necessary reasoning step
     toward it.
```

```
First infer the query's retrieval need (temporal,
      preference, multi-hop, or knowledge-update). Then
       apply the matching rule below:
- Temporal: prefer relationships that state when
      something happened, link an event to a date,
      describe before/after order, recency, duration,
      or event chronology.
- Preference: prefer relationships that state likes,
      dislikes, favorites, habits, goals, choices,
      constraints, or stable personal tendencies.
- Multi-hop: prefer relationships that connect two
      useful entities in the reasoning chain (subject-
      event, event-date, person-preference, or bridge
      relations).
- Knowledge-update: prefer relationships that indicate
      newer, corrected, replaced, or currently valid
      information; penalize stale or superseded facts
      when the query asks for the latest state.

Mark as relevant only if the relationship helps answer,
       retrieve evidence for, or justify the answer.
      Mark as not relevant if it only shares broad
      topic words or expresses a generic association
      that does not narrow down the answer.

Example -- Query: "When did Audrey make muffins for
      herself?"
Relevant: a relationship linking the muffin/pastry
      event to "Week of April 3, 2023" -- supports the
      temporal answer.
Not relevant: a relationship about pastries in general
      with no connection to the specific event or its
      date.
```

Instruction-free control prompt. An earlier version of the reranker scored every candidate with a single generic template that asked only whether the document was useful, without a task instruction and without distinguishing candidate types:

Generic reranking control prompt

```
Query: {query}

Document: {document}

Question: Is the document useful to answer the query?
      Answer "Yes" or "No".
```

From a generic scorer to a memory-evidence scorer. The instruction-free control template leaves *useful* undefined for long-term memory question answering over a knowledge graph, so the reranker tends to conflate topic-level relatedness with answer-bearing relevance: a candidate that merely shares a person, an event, or a keyword with the query can still score highly even when it cannot support the answer. The instruction-conditioned prompt restores that format and adds a task instruction, split by candidate type, that first infers the query's retrieval need (temporal, preference, multi-hop, or knowledge update) and then prefers candidates pointing to the target person, event, time, preference, attribute, or multi-hop bridge relation over candidates that only share broad topic words. The design intent is to steer the reranker toward answer-bearing evidence rather than topically related distractors; as the measurements below show, its dominant measured effect is a large gain in gold-evidence coverage.

Table 10: Generic reranking control prompt vs. instruction-conditioned reranking prompt on LongMemEval. The instruction-conditioned prompt nearly doubles gold-evidence coverage (recall) and lifts overall accuracy.

Metric / Category	Generic	Instruction	Δ
Retrieval vs. gold evidence			
Gold coverage (recall)	34.5	63.4	+28.9
All-gold-hit rate	35.3	59.9	+24.6
Answer accuracy			
Overall	**70.7**	**73.7**	**+3.0**

Retrieval and end-to-end effect. Table 10 compares a run using the generic reranking control prompt with one using the instruction-conditioned prompt on LongMemEval, reporting gold-evidence coverage and downstream answer accuracy. The instruction-conditioned prompt substantially improves retrieval coverage: gold-evidence recall rises from 34.5% to 63.4%, an increase of 28.9 points. This improvement also leads to better end-to-end performance, with overall answer accuracy increasing by 3.0 points, from 70.7% to 73.7%. These results indicate that the instruction-conditioned prompt improves downstream accuracy primarily by recovering a larger fraction of the answer-bearing evidence. The numbers reflect this controlled comparison and are not the headline system configuration reported in the main results tables.

E Component Ablation Results

We conduct component ablations on the retrieval pipeline before the agentic evidence-curation layer. The reference configuration retains all graph-retrieval, memory-representation, reranking, and temporal-processing components, but excludes both the Evidence Curation Agent and Evidence Validation. We refer to this configuration as the full retrieval pipeline. Complete category-level results are reported in Section E.4.

E.1 Graph-Structured Retrieval

This group evaluates the contribution of the graph retrieval channel and its main internal components. In the full retrieval pipeline, GRACE-Mem retrieves entity and relation candidates using dense

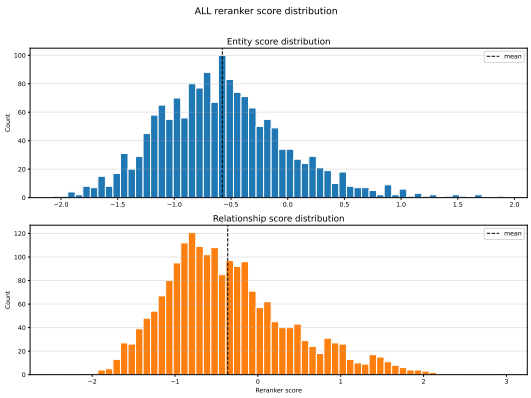


Figure 5: Reranker score distribution under the Generic reranking control prompt, shown separately for entity candidates (top) and relationship candidates (bottom). The dashed line marks the mean. Scores collapse into a narrow unimodal band spanning roughly two score units, leaving little margin between retained and discarded candidates.

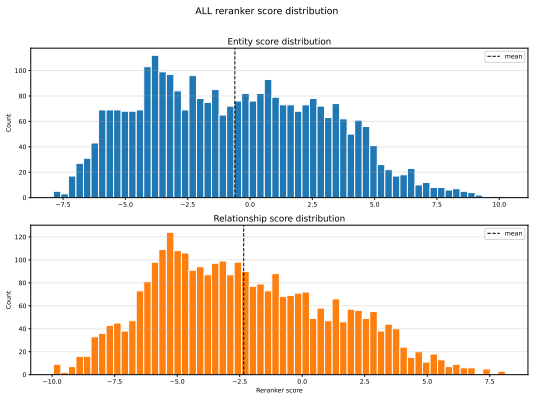


Figure 6: Reranker score distribution under the instruction-conditioned prompt, for the same query batch and the same candidate types as Figure 5, and plotted on the same vertical scale. The score range widens by roughly an order of magnitude (to about ± 10), giving a much larger margin between rejected and endorsed candidates.

and lexical signals, expands the retrieved entities through spreading activation, constructs a graph candidate set, and reranks its entities and relations before collecting their linked compressed records. These graph-linked records are subsequently merged with records obtained through direct dense retrieval.

We construct four ablation variants to isolate the role of each stage. **No graph retrieval** removes the entire graph channel, including entity and relation retrieval, spreading activation, graph candidate construction, and graph-element reranking, leaving only direct dense retrieval over the compressed-record index. **No lexical entity retrieval** disables the BM25-based entity retrieval branch while retaining dense entity retrieval and the remaining graph pipeline, testing whether exact lexical matching provides information beyond semantic similarity. **No spreading activation** retains the initially retrieved graph candidates but disables propagation-based multi-hop expansion, isolating the contribution of graph connectivity to recovering indirectly related evidence. **No graph-level reranking** removes the graph-level entity and relation reranker before compressed-record candidate construction, testing whether the initially retrieved and expanded graph candidates require an additional relevance-selection stage.

Table 11: Graph-structured retrieval ablations on LoCoMo and LongMemEval with OSS-20B.

Configuration	LoCoMo	LongMemEval
Full retrieval pipeline	84.0	76.2
No graph retrieval	66.4 (-17.6)	67.9 (-8.3)
No lexical entity retrieval	80.2 (-3.8)	72.2 (-4.0)
No spreading activation	81.2 (-2.8)	73.0 (-3.2)
No graph-level reranking	81.1 (-2.9)	71.3 (-4.8)

Among the graph-side ablations, removing the entire graph channel produces the largest overall degradation on both benchmarks, reducing OSS-20B accuracy by 8.3 points on LongMemEval and 17.6 points on LoCoMo. The larger LoCoMo drop is most visible in multi-hop and temporal questions, whose accuracy falls from 81.6 to 57.4 and from 74.5 to 48.3, respectively. Removing lexical entity retrieval, spreading activation, or graph-element reranking also lowers accuracy on both benchmarks, with drops ranging from 2.8 to 4.8 points.

E.2 Memory and Evidence Representation

This group examines how memory units are constructed, how candidate records are retrieved, and which representation is ultimately delivered to the answer model. The full retrieval pipeline uses role-separated dialogue units during memory construction, combines graph-linked candidates with direct dense retrieval over the compressed-record index, and reconstructs the final answering context from provenance-linked raw dialogue spans.

We construct three ablation variants to isolate these design choices. **Unsplit turn-pair units** re-

places the role-separated dialogue units used by the full pipeline with a single combined unit containing both sides of each turn pair. This variant tests whether separating speaker-specific content into finer-grained memory units improves retrieval precision and reduces interference from unrelated information within the paired turn. Because this unit construction was evaluated only for LongMemEval, no corresponding LoCoMo result is reported.

No direct compressed-record retrieval removes the dense retrieval path over the compressed-record vector index while retaining graph-based retrieval and the compressed records linked to the selected graph candidates. This variant tests whether direct record-level retrieval complements graph construction by recovering relevant dialogue units that are omitted because of imperfect entity extraction, relation extraction, or graph connectivity.

Compressed records only retains the complete retrieval and reranking pipeline but supplies the selected compressed records directly to the answer model instead of reconstructing the final context from their provenance-linked raw dialogue spans. This variant isolates the contribution of source-level dialogue evidence and tests whether compressed representations alone preserve the speaker, temporal, and contextual details required for answering.

Table 12: Memory and evidence representation ablations on LoCoMo and LongMemEval with OSS-20B.

Configuration	LoCoMo	LongMemEval
Full retrieval pipeline	84.0	76.2
Unsplit turn-pair units	–	69.5 (−6.7)
No direct compressed-record retrieval	80.2 (−3.8)	69.7 (−6.5)
Compressed records only	76.2 (−7.8)	70.0 (−6.2)

On LongMemEval, replacing role-separated evidence units with unsplit turn pairs reduces accuracy by 6.7 points, indicating that finer-grained evidence units support more effective retrieval. We report this ablation only on LongMemEval because LoCoMo uses dialogue-turn units rather than the role-separated units used in LongMemEval, making the same comparison not directly comparable. Direct compressed-record retrieval also contributes across both benchmarks: removing this path lowers accuracy by 6.5 points on LongMemEval and 3.8 points on LoCoMo. Restricting the final context to compressed records causes the largest degradation in this group, reducing accuracy by 6.2 points on LongMemEval and 7.8 points on LoCoMo.

As shown in Table 14 and 15, the decline is particularly pronounced for LongMemEval temporal questions and for LoCoMo multi-hop questions. These results show that compressed-record retrieval is useful for candidate discovery, but compressed-only context does not preserve sufficient detail for temporal and multi-hop reasoning.

E.3 Retrieval and Temporal Design Choices

This group evaluates three design choices within the retrieval pipeline: query-side keyword extraction, temporal normalization of dialogue content, and compressed-record reranking. The main graph retrieval and evidence representation architecture is retained in all variants.

We construct three ablation variants to isolate the contribution of each component. **No keyword extraction** disables query-side keyword generation. Entity retrieval therefore relies on the complete query embedding without keyword-based lexical entity matching, while relation retrieval no longer receives the fine-grained keyword embeddings used to identify relation-specific cues. This variant tests whether extracted keywords provide retrieval signals that are complementary to the full-query representation.

No temporal normalization disables the normalization of relative temporal expressions in dialogue records while leaving the remaining ingestion and retrieval pipeline unchanged. Expressions such as “yesterday,” “last month,” or “two weeks ago” are therefore retained in their original form rather than being grounded to the corresponding dialogue date. This variant tests whether temporally normalized memory representations improve the retrieval and interpretation of time-sensitive evidence across sessions.

No compressed-record reranking removes the final reranking stage over the merged pool of graph-linked and directly retrieved compressed records. Instead, the candidate records produced by the preceding retrieval stages are passed forward without query-conditioned record-level selection. This variant tests whether reranking is necessary to prioritize answer-relevant records and remove candidates that are structurally or semantically related to the query but insufficient for answering it.

Table 13: Retrieval and temporal design ablations on LoCoMo and LongMemEval with OSS-20B.

Configuration	LoCoMo	LongMemEval
Full retrieval pipeline	84.0	76.2
No keyword extraction	79.9 (−4.1)	73.0 (−3.2)
No temporal rewriting	81.0 (−3.0)	75.4 (−0.8)
No compressed-record reranking	79.3 (−4.7)	74.2 (−2.0)

Keyword extraction and compressed-record reranking contribute consistently across both benchmarks. Removing keyword extraction reduces accuracy by 4.1 points on LoCoMo and 3.2 points on LongMemEval, while removing compressed-record reranking causes drops of 4.7 and 2.0 points, respectively.

E.4 Joint Interpretation

Tables 14 and 15 report the complete category-level ablation results. For LongMemEval, User denotes single-session user, Multi-sess. denotes multi-session reasoning, Pref. denotes single-session preference, Assistant denotes single-session assistant, Temporal denotes temporal reasoning, and KU denotes knowledge update.

Across the three ablation groups, graph retrieval, direct compressed-record retrieval, and provenance-linked evidence provide complementary rather than interchangeable functions. The graph path connects related entities and events, direct compressed-record retrieval recovers content outside the selected graph seeds, and fine-grained evidence supplies traceable source material to the answer model.

The category-level results also reveal non-monotone effects. Some component removals improve individual categories despite reducing overall accuracy. For example, removing temporal rewriting increases LongMemEval preference accuracy from 60.0 to 73.3, while removing spreading activation increases multi-session accuracy from 61.7 to 63.9. We therefore interpret the ablations as evidence of complementary overall contributions rather than uniform improvements across every category.

F LLM-as-Judge Evaluation Protocol

All newly generated GRACE-Mem results and controlled ablations in this paper are evaluated by an LLM judge (GPT-4o-mini, temperature 0) that compares the model answer to the gold answer and returns a binary CORRECT/WRONG label. This appendix documents the two components of that protocol: the single-pass judge and an abstention-specific rubric.

Single-pass judge. For a normal question, the judge receives the question, the gold answer, and the generated answer under a semantic-equivalence rubric: an answer is CORRECT if it refers to the same fact or entity as the gold answer, regardless of surface form. Time questions are evaluated by the same semantic-equivalence criterion—a relative phrasing ("last Tuesday") or a reformatted date ("May 7th" vs. "7 May") counts as correct if it denotes the same date. To reduce spurious mismatches on relative-date gold answers, relative temporal expressions in the gold answer (e.g. "the week before 25 May 2023", "the Sunday before X", or "a few days before X") are deterministically expanded to an absolute date or interval and passed to the judge as a normalization hint before grading. The judge emits a one-sentence rationale followed by the label in JSON. Each question is graded once.

LongMemEval abstention (`_abs`) questions. For LongMemEval only, a subset of questions is *unanswerable*: the asked-about fact was never stated, so the only correct behavior is to abstain (the gold answer is a phrase such as "the information provided is not enough" / "did not mention", or a count of 0). The generic rubric—which only asks "does the response contain the gold answer"—systematically mislabels clean abstentions as wrong, so these questions are routed to a dedicated rubric. Membership is taken *authoritatively* from the dataset's `_abs` tag when available, and otherwise inferred from abstention markers in the gold text. Under the abstention rubric the judge sets CORRECT when the response cleanly declines (states it has no record / insufficient information, or answers "0"/"none" when the event never happened), and WRONG when the response fabricates any concrete number, date, duration, name, or ordering—*including* the common failure of being led by a similar-but-different distractor mentioned earlier and answering about that instead. Non-abstention questions continue to use the unchanged default rubric.

G Prompt Templates

This appendix presents the core prompt templates used in the reported experiments. We include the behavioral contracts and output formats that

Table 14: Category-level component ablations on LongMemEval with OSS-20B.

Configuration	Overall	User	Multi-sess.	Pref.	Assistant	Temporal	KU
Full retrieval pipeline	76.2	88.6	61.7	60.0	85.7	86.4	71.8
Graph-Structured Retrieval							
No graph retrieval	67.9	88.6	57.1	63.3	87.5	59.1	57.1
No lexical entity retrieval	72.2	91.2	61.4	66.7	91.1	72.7	61.5
No spreading activation	73.0	90.0	63.9	66.7	91.1	71.2	65.4
No graph-level reranking	71.3	90.0	58.6	63.3	89.3	72.0	65.4
Memory and Evidence Representation							
Unsplit turn-pair units	69.5	87.1	63.2	60.0	89.3	65.2	61.5
No direct compressed-record retrieval	69.7	87.1	56.4	70.0	89.3	72.7	57.7
Compressed records only	70.0	89.5	60.9	60.0	89.3	59.8	78.2
Retrieval and Temporal Design Choices							
No keyword extraction	73.0	90.0	63.2	66.7	91.1	73.5	62.8
No temporal rewriting	75.4	94.3	62.4	73.3	87.5	76.5	70.5
No compressed-record reranking	74.2	88.6	62.4	80.0	91.1	75.0	65.4

materially affect the system. Complete verbatim prompts, additional few-shot examples, inactive variants, and implementation wrappers are provided in the accompanying code repository.

G.1 Entity Extraction Prompt

Entity extraction operates on temporally normalized raw dialogue; the normalization is deterministic and does not use an additional LLM prompt.

Prompt for entity extraction

```
GOAL:
Given the conversation, identify all core entities for
      knowledge-graph construction.

REFERENCE TIME:
{dialogue_datetime}

RULES:
- Output entities only.
- Use English for all outputs.
- entity_type must be one of: Person, Event, Date, Time,
       Timespan, Location, Organization, Product,
      Service, Activity, Topic, Concept.
- Each entity must have a concise and factual
      description.
- Create Event entities for important actions, plans,
      decisions, changes, milestones, and incidents.
- Marker tags are authoritative:
  [DATE: X]     -> Date entity named X
  [TIME: X]     -> Time entity named X
  [TIMESPAN: X] -> Timespan entity named X
- Do not recompute, rename, normalize, or paraphrase
      marker payloads.
- An Event and its temporal anchor must be represented
      separately.
- Do not infer an exact date from an unmarked fuzzy
      expression.

OUTPUT:
("entity" <|> entity_name <|> entity_type <|>
      entity_description)

Return one flat list separated by <|RECORD|> and
      terminate with <|COMPLETE|>.

INPUT:
Raw conversation:
{raw_conversation}
```

G.2 Relationship Extraction Prompt

Relationship extraction is performed in a second stage and may only use entities produced by the preceding entity-extraction stage.

Prompt for relationship extraction

```
GOAL:
Identify relationships among the provided entities.

REFERENCE TIME:
{dialogue_datetime}

RULES:
- Output relationships only; do not create entities.
- source_entity and target_entity must exactly match a
      provided entity_name.
- If a required entity is missing, omit the
      relationship.
- Do not recompute, rename, or paraphrase temporal
      anchors.
- Prefer relations useful for WHO, WHAT, WHEN, WHERE,
      and HOW.
- Prefer:
  Event -> Date/Time/Timespan for temporal links
  Event -> Location for spatial links
- relationship_keywords must contain 2--4 lowercase
      English keywords.

OUTPUT:
("relationship" <|> source_entity <|> target_entity <|>
       relationship_description <|>
      relationship_keywords)

Return one flat list separated by <|RECORD|> and
      terminate with <|COMPLETE|>.

INPUT:
```

Table 15: Category-level component ablations on LoCoMo with OSS-20B.

Configuration	Overall	Single-hop	Multi-hop	Open-domain	Temporal
Full retrieval pipeline	84.0	90.1	81.6	68.8	74.5
Graph-Structured Retrieval					
No graph retrieval	66.4	77.6	57.4	55.2	48.3
No lexical entity retrieval	80.2	86.3	76.1	60.2	73.8
No spreading activation	81.2	86.9	78.7	60.4	74.5
No graph-level reranking	81.1	86.3	79.4	62.5	74.5
Memory and Evidence Representation					
No direct compressed-record retrieval	80.2	85.8	77.7	59.4	73.8
Compressed records only	76.2	87.4	77.7	60.4	50.6
Retrieval and Temporal Design Choices					
No keyword extraction	79.9	86.2	76.2	55.2	73.8
No temporal rewriting	81.0	87.4	76.6	59.4	74.5
No compressed-record reranking	79.3	83.8	76.2	62.5	75.1

```
Raw conversation:
{raw_conversation}

Provided entities:
{entities_text}
```

G.3 Entity Reconciliation Prompt

Entity reconciliation determines whether each newly extracted entity should be added as a new graph node or used to update an existing candidate node. The merged description is produced by the same call rather than by a separate consolidation pass.

Prompt for entity reconciliation

```
TASK:
Reconcile each newly extracted entity against the
      existing candidate entities.

DECISION POLICY:
- Choose UPDATE only when the new entity and a
      candidate clearly denote the same real-world
      object, person, event, or concept.
- Type differences alone do not imply separation.
- Similarity score is only a hint, not proof.
- Related but distinct entities must remain separate.
- When uncertain, choose ADD.

SUBJECT GUARD:
Events whose primary subjects are different named
      people must remain separate, even when their
      names and descriptions are similar.

DESCRIPTION MERGE:
For UPDATE, merged_description must:
- preserve facts from the existing description;
- preserve facts from the new description;
- include new temporal, role, or version information;
- remain concise and factual.

OUTPUT:
===BEGIN===
input_name || input_type || action ||
      target_existing_id_or_NULL || canonical_name ||
      canonical_type || merged_description
===END===

CONSTRAINTS:
- UPDATE target IDs must be copied from the valid
      candidate IDs.
- ADD must use NULL as the target ID.
- merged_description must not be empty.
```

The two few-shot examples below are reproduced verbatim from the prompt, with line breaks inserted to fit the column. The first shows that a differing type label does not by itself imply a distinct entity; the second shows the subject guard.

Entity reconciliation few-shot example: UPDATE

```
[INPUT]
name=OpenAI | type=organization | desc=AI research and
      deployment company; created GPT models.
Candidates:
- id=org_OpenAI | name=OpenAI | type=org | score=0.91 |
       desc=An AI company known for foundation models
      and an API platform.
Valid target_existing_id choices: ["org_OpenAI"]

EXPECTED OUTPUT (between markers):
===BEGIN===
OpenAI||organization||UPDATE||org_OpenAI||OpenAI||
      organization||An AI research and deployment
      company known for foundation models and an API
      platform; creator of GPT models.
===END===
```

Entity reconciliation few-shot example: conservative ADD

```
[INPUT]
name=check-up with doctor | type=event | desc=Sam
      visited the doctor for a routine check-up a few
      days before May 24.
Candidates:
- id=event_checkup_001 | name=check-up with doctor |
      type=event | score=0.83 | desc=Evan's son had a
```

```
      doctor check-up in early August due to recurring
      stomach issues.
Valid target_existing_id choices: ["event_checkup_001"]

EXPECTED OUTPUT (between markers):
===BEGIN===
check-up with doctor||event||ADD||NULL||check-up with
      doctor||event||Sam visited the doctor for a
      routine check-up a few days before May 24.
===END===
```

G.4 Keyword Extraction Prompt

Prompt for retrieval keyword extraction

```
You extract retrieval keywords for memory question
      answering.

Return JSON only with exactly two keys:
{
  "high_level_keywords": [
    abstract intent or reasoning type
  ],
  "low_level_keywords": [
    concrete entities, events, dates, and lexical
      anchors
  ]
}

RULES:
- Do not hallucinate missing facts or the final answer.
- low_level_keywords must not be empty.
- Preserve names, organizations, activities, objects,
      dates, time expressions, and event phrases.
- Ignore first-person pronouns and retain the actual
      retrieval target.
- For temporal, counting, duration, or comparison
      questions, include the reasoning intent in
      high_level_keywords.

QUERY: {query}
```

G.5 Agent Filter Prompt

The agent performs evidence selection rather than answer generation, validating retrieved candidates and recovering relevant turns missing from the initial set. The `VECTOR` tool block is injected only when a compressed-record vector index is available for the question; otherwise the tool is not shown to the agent.

System prompt for the evidence-selection agent

```
You are an evidence-selection agent for a long-term
      memory question-answering system.

You receive a QUESTION and candidate evidence turns,
      each with a provenance sid. Candidates may
      contain distractors, and relevant turns may be
      missing.

Use the available tools to verify candidates, search
      the raw conversation corpus, and return the final
       evidence sids.

TOOLS:
  GREP <regex>
    Case-insensitive search over all raw conversation
      turns. Dates, names, numbers, and distinctive
      phrases are searchable.

  READ <sid> [k]
    Return the raw turns surrounding sid. The default
      window is 2.

  VECTOR <query>
    Semantic search over compressed conversation
      records. Present only when a
    compressed-record index exists for the question.
      Results are unverified and
    must be checked with READ or GREP before inclusion.

  FINAL <sid> <sid> ...
    Return the selected evidence sids.

RULES:
- Emit exactly one command as the final line of each
      response.
- Copy every sid exactly as displayed.
- Do not repeat a search that produced zero results.
- Verify suspicious candidates with READ.
- Remove distractors and preserve every turn that
      supports the answer.
- Search for missing evidence when the initial
      candidates are incomplete.
- For counting questions, collect every distinct
      relevant mention.
- For updated facts, retain the dated evidence needed
      to identify the most recent value.
- Use no more than {max_calls} tool calls.
- Stop and return FINAL as soon as the evidence is
      sufficient.
```

G.5.1 User-turn template

The selected hint is inserted directly, without a field label. The question-date line is present only when the question date is known.

User message for the evidence-selection agent

```
QUESTION: {question}

QUESTION DATE:
{question_date}
{category_hint}

CANDIDATE EVIDENCE TURNS: {candidates}

Verify the candidates, search for missing evidence, and
       return the final evidence sids.
```

G.6 Answer Generation Prompts

All reported LongMemEval settings, including the agent-filter configuration, use the same default answer prompt: the agent filter changes the retrieved context rather than the answering instruction. LoCoMo uses a different template in which the instruction is carried by the user turn.

In the template below, the date block is appended only when the question date is known.

Default LongMemEval answer prompt

```
SYSTEM:
You are a concise and accurate assistant.

Use the Retrieved Context. If the context is
      insufficient, general knowledge may be used, but
      retrieved facts should be preferred.

Answer directly.
```

```
Current Date/Time: {question_date}
Note: When answering temporal questions (e.g., "how
      long ago", "how many months"), calculate based on
       this date.

---Retrieved Context---
{context}
-----------------------

USER:
Question: {question}

Answer:
```

LoCoMo answer template

```
SYSTEM:
---Retrieved Context---
{context}
-----------------------

USER:
Please answer based on the retrieved knowledge-graph
      context above. Be concise and accurate.

Question: {question}

Answer:
```

G.7 Evaluation Judge Prompts

These prompts are used only for automatic evaluation and do not affect retrieval or answer generation.

G.7.1 LongMemEval judge

LongMemEval evaluation template

```
Evaluate whether the generated answer contains the
      correct answer to the question.

QUESTION: {question}

REFERENCE ANSWER:
{gold_answer}

GENERATED ANSWER:
{predicted_answer}

CATEGORY-SPECIFIC POLICY:
{category_rubric}

Return JSON:
{
  "reasoning": "one-sentence explanation",
  "correct": true or false
}
```

LongMemEval abstention (_abs) judge. Questions marked _abs ask for information that was never provided. They use the following dedicated template rather than the category-specific template above. The template deliberately distinguishes an answer to the asked-about item from an explicitly labelled comparison with a different, similar item: a concrete value is disqualifying only when it is offered as the answer to the asked-about item.

Table 16: LongMemEval category-specific judging criteria.

Category	Additional criterion
Temporal reasoning	Minor off-by-one duration errors are accepted.
Knowledge update	The latest required value determines correctness, even if older information is also mentioned.
Preference	Correct use of the user's personal information is sufficient; not every reference point must be repeated.
Other categories	The generated answer must contain or be semantically equivalent to the required answer.

LongMemEval abstention evaluation template

```
Evaluate an ABSTENTION question: the asked-about
      information was never
provided, so the only correct behavior is to abstain.

QUESTION:
{question}

REFERENCE ANSWER:
{gold_answer}

GENERATED ANSWER:
{predicted_answer}

Set \texttt{correct=true} when the response clearly
      states that it has no
record / insufficient information / cannot answer, or
      correctly answers
``0'' or ``none'' when the event never occurred. A
      response remains correct
when it abstains on the asked-about item while
      explicitly identifying a
different, similar item as a contrast; any value must
      be clearly attributed to
that different item, not presented as the requested
      answer. It is also correct
to offer conditional help (for example, to calculate
      the result if the user
provides missing input).

Set \texttt{correct=false} when the response gives or
      borrows a definite
number, date, duration, name, or ordering as the answer
       to the asked-about
item, including an unlabelled substitution of a similar-
      but-different
distractor.

Return JSON:
{
  "reasoning": "one-sentence explanation",
  "correct": true or false
}
```

LongMemEval judging schedule. Each item receives a single GPT-4o-mini judgment at temperature 0. Non-_abs items are evaluated using their category-specific templates, whereas _abs items use the abstention template described above.

G.7.2 LoCoMo judge

LoCoMo standard evaluation template

```
SYSTEM:
You are an expert grader that determines whether an
      answer matches a gold-standard answer.

USER:
Label the generated answer as CORRECT or WRONG.

Question:
{question}

Gold answer:
{gold_answer}

Generated answer:
{response}

Be generous about semantically equivalent answers and
      equivalent date formats. Return JSON only:

{"label": "CORRECT"}
or
{"label": "WRONG"}
```

Explaining Textual Entailment with Lexical Entailments: Using LLMs to Supply Lexical Relations for Formal Proofs

Jorryt de Jong[1] **Stefan Morača**[1] **Ettore Cesari**[1] **Lasha Abzianidze**[2]
[1]Utrecht University, the Netherlands
[2]Institute for Language Sciences, Utrecht University, the Netherlands
{j.e.j.dejong,s.moraca,e.cesari}@students.uu.nl
Correspondence: l.abzianidze@uu.nl

Abstract

Large Language Models (LLMs) are highly capable of natural language reasoning and appear to store a great deal of lexical knowledge, but it is still unclear how much of this knowledge they actually use when reasoning, and whether they use it in the right way. On the other hand, logic-based Natural Language Inference (NLI) systems provide transparent and formally grounded reasoning, but they need to be supplied with rich lexical knowledge to prove inferences beyond purely logical ones. In this paper, we evaluate whether LLMs can identify all lexical knowledge needed to solve NLI problems and how much this knowledge contributes to proof search in a logic-based NLI system. Our research focuses exclusively on structured lexical entailments (e.g., **chinchilla ⊑ small animal**) as a proxy for structured explanations for NLI problems with an entailment label. First, we curate a dataset for a new task of explaining sentential entailments with a set of lexical entailments. The dataset is used to intrinsically evaluate LLMs on generating structured lexical explanations. Then, we use NLI as an extrinsic evaluation in a simple neuro-symbolic setting, assessing whether LLMs can supply sufficient lexical relations to LangPro, a natural-logic theorem prover for natural language. The results show that the proposed task remains challenging even for hosted proprietary LLMs, and that their contribution to theorem proving is moderate: generated relations are often only partially sound and may be tailored to the specific NLI problem rather than representing generally valid lexical knowledge.

1 Introduction

Recent LLM progress has shifted much of NLP research toward downstream tasks. However, Natural Language Inference (NLI, Dagan et al., 2013), a generic reasoning task, remains popular because it allows evaluation of models on specific reasoning phenomena. The NLI task challenges models to predict whether a hypothesis sentence is entailed by, contradicted by, or is neutral with respect to a premise text. While the frontier models have saturated common NLI benchmarks, we still do not know how sound their reasoning is when arriving at correct inference labels.

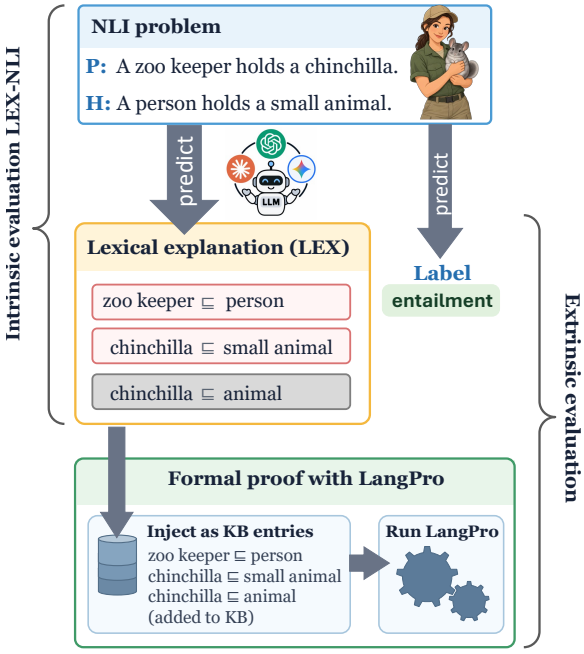


Figure 1: Overview of the proposed intrinsic and extrinsic evaluation. Given an NLI problem, an LLM is jointly evaluated on the predicted inference label and a structured lexical explanation (LEX) against human-curated multi-reference LEX sets. Extrinsically, the generated lexical relations are supplied to the LangPro theorem prover, which determines whether they suffice to construct a formal proof.

To study how LLMs reason in NLI more closely, researchers have built simplified synthetic NLI-style datasets (that usually require multi-step reasoning) and pursued two related lines of work: (i) train and evaluate LLMs on multi-step reasoning in natural language (Clark et al., 2020; Zhong et al., 2022; Saparov and He, 2023; Han et al., 2024),

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 316–333
October 28, 2026 ©2026 Association for Computational Linguistics

and (ii) use LLMs to encode sentence meaning in a formal logic formula that is later fed to rule-based reasoners, such as first-order logic theorem provers (Olausson et al., 2023; Pan et al., 2023; Ryu et al., 2025). However, this suggests that the problems from the original NLI datasets[1] are too challenging for (i) and (ii), and such valuable datasets are unfortunately left out of these research lines.

To partially formalize NLI reasoning, we introduce an explainable task in which models produce structured lexical relations supporting the inference label. We also evaluate a neuro-symbolic pipeline distinct from (ii). Specifically, LLMs generate lexical entailments—not logical formulas—to help the natural logic prover LangPro (Abzianidze, 2017a) address scarce KB resources.[2] This aligns with work treating LLMs as knowledge bases (KB) (Petroni et al., 2019; AlKhamissi et al., 2022). Our evaluation setup is illustrated in Figure 1.

Our contributions are threefold:

(1) We curate a test set of entailment problems together with the corresponding lexical entailments that are necessary and sufficient for solving them;

(2) We evaluate multiple LLMs against the test set on the LEX-NLI task, identifying relevant lexical entailments for NLI problems, and analyze the results with respect to model family and size;

(3) We select high-performing LLMs and use them as KB generators for LangPro while the latter is evaluated on its ability to prove NLI problems. Together, these results demonstrate an interpretable integration of LLMs and a symbolic reasoner.

2 Related Work

NLI is one of the most widely studied NLP tasks. The release of e-SNLI (Camburu et al., 2018) initiated a research line on explainable NLI by augmenting NLI problems with human-elicited free-text explanations and span highlights. However, the lack of precise and adequate metrics for automatically comparing generated and reference explanations hinders progress in free-text explainable NLI.

Several alternatives to free-text explanations have therefore been explored. The first line of work directly stems from e-SNLI and uses its span highlights for training and/or evaluation of explainable NLI models (DeYoung et al., 2020; Zhao and Vydiswaran, 2021; Carton et al., 2022; Stacey et al., 2022; Ghasemi Madani and Minervini, 2023). However, the span highlights, which Camburu et al. (2018) used to facilitate free-text explanation collection, cannot be regarded as lexical knowledge and are too noisy for a gold standard. The second line of work delegates reasoning to rule-based systems, while LLMs translate sentences into meaning representations, often first-order logic (Olausson et al., 2023; Pan et al., 2023; Ryu et al., 2025; Noble et al., 2025). The third line of work constructs synthetic NLI-style datasets in which the reasoning process is represented as a tree, graph, or sequence of formally validated inference steps (Clark et al., 2020; Zhong et al., 2022; Saparov and He, 2023; Dalvi et al., 2021; Han et al., 2024). The last two lines of work rely on simpler language or more restricted reasoning patterns than those found in standard NLI benchmarks such as SNLI (Bowman et al., 2015) and SICK (Marelli et al., 2014).

In contrast, we use existing entailment problems from SNLI and SICK augmented with structured lexical explanations, formatted as curated multiple-reference relation sets. We restrict the task to entailment rather than to synthetic sentences or simple reasoning patterns, as in the related work.

State-of-the-art rule-based NLI systems (Yanaka et al., 2018; Abzianidze, 2020) suffer from lexical knowledge sparsity despite using resources such as WordNet (Fellbaum, 1998) and ConceptNet (Speer et al., 2017). Consequently, several studies used neural models to supply missing lexical/phrasal knowledge. Yoshikawa et al. (2019) applied the ComplEx KB completion model (Trouillon et al., 2016) to generate lexical relations for a theorem prover. Tomihari and Yanaka (2023) used vision-language embeddings to assess phrase-level entailments selected by a logic-based NLI system. For Japanese logic-based NLI, Kobayashi et al. (2026) developed a lexical-entailment classifier inspired by hyperbolic embeddings (Ganea et al., 2018). Chen et al. (2021) complemented WordNet and ConceptNet with phrasal paraphrases derived from a pretrained language model, while Noble et al. (2025) used an LLM to generate first-order-logic axioms corresponding to lexical relations for a theorem prover. Aly et al. (2023) and Strong et al. (2024) on natural-logic-based fact verification use instruction-tuned LLMs to predict short-phrase relations for a version of natural logic.

Our work instead formulates lexical KB prediction as an independent explainable task. Reference

[1] For instance, RTE (Dagan et al., 2006), SICK (Marelli et al., 2014), and SNLI (Bowman et al., 2015).

[2] Unlike ours, the works in (i) and (ii) do not use LLMs to produce lexical knowledge because their synthetic datasets require little covert lexical knowledge.

lexical relations are not tied to third-party systems or their representation language. We judge them by whether they explain the NLI inference, not by whether a given reasoning system can import them and successfully use them in its proofs.

3 Structured Explanation with Lexical Entailments

We use lexical entailments, such as **guitar** $\sqsubseteq$ **musical instrument**, as structured NLI explanations. We first motivate our focus on entailment problems and then show how lexical entailments can serve as a key component of the underlying reasoning. Finally, we define the kinds of words and short phrases that may occur in lexical entailments.

Earlier formulations of Recognizing Textual Entailment (Dagan et al., 2006) distinguished between entailment and non-entailment, whereas contemporary NLI datasets typically divide non-entailment into contradiction and neutral. The distinction between contradiction and neutral, however, is not always straightforward.[3] Moreover, non-entailment cases are generally more difficult to explain using only word- or phrase-level relations. A neutral label usually results from the absence of certain semantic relations, and contradictions sometimes depend only on negation, which doesn't require relations. In contrast, word- or phrase-level relations frequently represent knowledge needed to explain entailment NLI problems, e.g., "a zoo keeper holds a chinchilla" entailing "a person holds a small animal" is explained with **zoo keeper** $\sqsubseteq$ **person** and **chinchilla** $\sqsubseteq$ **small animal** entailment relations. We therefore currently focus only on entailment problems and lexical entailment relations between words and multi-word lexical items.

A lexical entailment is a directed binary relation, e.g., **guitar** $\sqsubseteq$ **musical instrument**, where the first argument is semantically more specific than or equal to the second. It is *structured* because it represents the knowledge in an explicit format. This makes the explanations easier to compare across annotators and models and allows them to be directly supplied to a symbolic reasoning system.[4]

We use the term *lexical* broadly enough to include both single- and multi-word concepts. Lexical items are essentially similar to those found in lexical databases such as WordNet, but extending beyond their incomplete coverage. Single-word relations include **cap** $\sqsubseteq$ **hat** and **obese** $\sqsubseteq$ **fat**, whereas multi-word relations include **female swimmer** $\sqsubseteq$ **woman** and **little boy** $\sqsubseteq$ **child**. Multi-word expressions are included when using only one of their components would produce an invalid or insufficient relation. For instance, **swimmer** $\sqsubseteq$ **woman** is false, whereas **female swimmer** $\sqsubseteq$ **woman** is acceptable. Similarly, **chinchilla** $\sqsubseteq$ **animal** is insufficient for a hypothesis referring to a "small animal", as the more informative relation **chinchilla** $\sqsubseteq$ **small animal** is required.

For consistency and comparability, expressions in a relation are normalized to their lemmatized forms and exclude determiners, auxiliary verbs, the infinitival marker "to", and prepositional phrases. We also exclude relations that are merely obtained by dropping modifiers: **tall man** $\sqsubseteq$ **man** is uninformative because entailment follows from the compositionality by discarding *tall*.

We call a set of lexical entailments that serves as a lexical basis of explanations for an NLI problem a LEX set, or LEX. Some entailment problems don't require lexical entailments. When the hypothesis follows through syntactic and compositional reasoning, the LEX set is empty. For example, "A zoo keeper holds a chinchilla" entailing "A chinchilla is held" doesn't require lexical relations but is licensed by passive alternation and subcategorization of "hold", which omits the agent.

LEX sets are not intended to constitute complete formal proofs of entailment. Instead, they serve as structured proxies for the lexical knowledge that licenses an entailment. When non-empty, they represent the lexical component that a complete formal or structured explanation must account for.

4 Curating a Test Set

To evaluate models on explainable NLI from a lexical perspective, we curate a set of NLI problems paired with reference LEX sets. We constructed a manually annotated test set of entailment problems drawn from SICK and SNLI, as these datasets heavily depend on lexical knowledge. All four authors served as both data curators and annotators.

[3] This was already discussed by Marelli et al. (2014). Consider a premise-hypothesis pair from SICK: "A couple is not looking at a map" and "A couple is looking at a map". The inference label depends on the interpretation of indefinite NPs: if "a couple" and "a map" co-refer across sentences, then the NLI problem is contradiction, otherwise neutral.

[4] In contrast, a free-text explanation doesn't make the comparison straightforward: detecting equality between "guitar means musical instrument" and "a guitar is a type of a musical instrument" is not trivial.

NLI problem with three LEX annotations and an optional 4th alternative LEX			
P: A man in a bright yellow shirt juggles while riding a unicycle. **H:** A man performs a juggling act on a unicycle.	∅	∅	∅
P: A young boy wearing safety swimming gear and goggles is in a pool. **H:** A kid uses eye protection while in the water.	**goggles ⊑ eye protection;** **pool ⊑ water; young boy ⊑ kid**	**goggles ⊑ eye protection;** **pool ⊑ water; young boy ⊑ kid**	**goggles ⊑ eye protection;** **pool ⊑ water; young boy ⊑ kid**
P: A woman observes a rusted antique car. **H:** A woman looks at an old car.	**antique ⊑ old;** **observe ⊑ look**	**antique car ⊑ old car;** **observe ⊑ look**	**antique ⊑ old;** **observe ⊑ look at**	**antique car ⊑ old car;** **observe ⊑ look at**
P: A person rides his bicycle in the sand beside the ocean. **H:** A person is on a beach.	**sand ⊑ beach**	∅	∅
P: A black and white dog is carrying a huge stick on the green grass. **H:** A black and white dog with a large branch is running in the field.	**huge stick ⊑ large branch**	**huge ⊑ large**	**green grass ⊑ field;** **huge ⊑ large**	**grass ⊑ field;** **huge ⊑ large;** **stick ⊑ branch**

Table 1: Examples of agreement and variation among three independently produced LEX annotations. An additional 4th LEX, if introduced during adjudication, is marked with a dashed borderline. ∅ stands for an empty LEX. The first two problems have unanimous agreement in LEX annotations.

Each annotator initially selected 70-100 entailment problems from the two datasets and assigned a LEX set to each selected problem. During this selection process, the annotators aimed for approximately 20-30% of the problems to have an empty LEX. This ensured that the test set contained both problems requiring non-trivial lexical knowledge and problems whose entailment could be explained through syntactic and compositional reasoning alone. All annotations were produced according to the guidelines presented in Appendix B. The selected NLI problems were subsequently pooled into a single collection. Each problem was then independently annotated by two additional annotators, besides its original annotator. Consequently, every problem received three independently produced LEX annotations. This procedure allowed us to assess both the reproducibility of the task and the extent to which different annotators identify the same lexical knowledge as explanatory.

We considered the annotations to be in unanimous agreement when all three annotators assigned the same LEX set. During the first phase, out of the 365 annotated problems, 161 (44%) received unanimous LEX annotations. The remaining cases exhibited variation in the selected relations, their granularity, or the amount of lexical knowledge considered necessary for explaining the inference. These disagreements do not always indicate annotation errors, since more than one set of lexical entailments may provide an acceptable explanation for the same NLI problem (see Table 1).

We therefore conducted an adjudication phase to distinguish annotation mistakes from legitimate explanatory variation. As a result, we obtained unanimous agreement for 223 (62%) problems (see Table 2).[5] When a disagreement could be resolved into a single adjudicated annotation, one annotator proposed the revised LEX. The revised annotation was subsequently checked by the annotators whose original LEX sets had been modified. This process ensured that changes were not introduced unilaterally and that the final annotation reflected agreement among the affected annotators.

When disagreement couldn't be reduced to a single canonical LEX (in total 139), several LEX variants were retained as a reference answer for the same problem. In several instances, adjudica-

[5]Three SNLI problems were excluded from the final dataset: one contained an incomplete sentence, and two had entailment labels that were judged unreliable.

Dataset statistics			
Total NLI problems			362
SNLI problems / SICK problems			280 / 82
Problems w/ unique LEX (unanimous agreement)			223 (61.6%)
Problems with an alternative (4th) LEX			22 (6.1%)
Total LEX (incl. the added 4th LEX)			1108
LEX size distribution: 0, 1, 2, 3+			29 / 51 / 14 / 6%
Annotator performance	n	Exact Match	micro-F1
Annotator-1 vs. rest	271	75.7	81.5
Annotator-2 vs. rest	271	84.5	86.0
Annotator-3 vs. rest	271	80.8	85.9
Annotator-4 vs. rest	273	82.8	86.2
Mean	–	80.9	84.9

Table 2: Statistics of the final dataset and annotator performance after the adjudication. Annotator-N vs rest is based on the problems annotated by Annotator-N.

tion also resulted in the creation of a fourth LEX variant that captured an acceptable explanation not represented by any of the three original annotations. This outcome indicates that variation is, to some extent, inherent to the task rather than being solely a consequence of inconsistent annotation. The examples from the final dataset are given in Table 1.

The resulting test set attaches one to four reference LEX sets (possibly empty LEX) to each NLI problem. 20% of the LEX annotations contain more than one relation, while 29% are empty. Overall annotator performance is 80.9% for exact match and 84.9% for micro-F1, computed as the mean of the four annotator-level scores. For each annotator, we calculate the scores by comparing the annotator's LEX to the other two or three LEXs (depending on the presence of the 4th LEX).

5 Lexical Entailments for Formal Proofs

The lexical entailments introduced in Section 3 can be converted directly into KB entries for logic-based NLI systems. For example, **chinchilla** $\sqsubseteq$ **small animal** can be represented as a subsumption relation stating that every chinchilla is a small animal. Such relations complement the logical rules of a prover with the lexical knowledge required to connect predicates occurring in the premise and hypothesis. A LEX set can therefore be viewed not only as a structured explanation of an inference, but also as a small problem-specific KB that may lead to a formal proof.

One logic-based NLI system that can use such knowledge is LangPro, a natural language theorem prover based on natural logic and semantic tableaux (Abzianidze, 2017a,b).[6] In addition to its logical rules, LangPro relies on a lexical KB containing relations such as synonymy, hypernymy, and semantic incompatibility. A large part of this knowledge is obtained from WordNet. However, WordNet does not contain every relation required by NLI problems. Consequently, a tableau may remain open even when LangPro's linguistic analysis and proof rules are otherwise sufficient to derive the intended inference. For instance, our running NLI example requires relations corresponding to **zoo keeper** $\sqsubseteq$ **person** and **chinchilla** $\sqsubseteq$ **small animal**, neither of which is available in WordNet.[7]

We use LLMs to supplement LangPro with lexical knowledge that is missing from WordNet. Rather than asking an LLM to predict the NLI label, construct the proof itself, or translate sentences into logic formulas, we use it only to propose relevant lexical knowledge. After receiving lexical KB from an LLM, LangPro remains solely responsible for determining whether the supplied KB is sufficient to find a proof. Thus, LangPro provides an extrinsic test of whether LLMs generate lexical entailments relevant for formal proof construction.

6 Experiments

We conduct two groups of experiments: evaluating LLMs intrinsically on the new task of predicting LEX sets and extrinsically on how much those predictions help LangPro prove NLI problems.[8]

6.1 Predicting Lexical Explanations

We evaluate eight instruction-following LLMs from four model families: GPT-5.4 and GPT-5.4 Mini, Claude Sonnet 4.5 and Claude Haiku 4.5, Gemini 3.5 Flash and Gemini 3.1 Flash-Lite, GPT-OSS-20B, and Gemma 3 4B. The selection combines higher-capability hosted models with smaller, cheaper alternatives from the same families, allowing us to examine whether LEX prediction requires a frontier model or can be handled by cheaper models. It also reduces dependence on a single developer. Additionally, GPT-OSS-20B and Gemma 3 4B provide open-weight comparison points.

Each model is evaluated five times using the same prompt. Only relations extracted from raw model responses are evaluated: no automatic post-processing was used to correct word forms or discard relations based on the vocabulary of an NLI problem. LLM inference details are provided in Appendix A. The prompts follow the annotation guidelines (see Appendix B). Prompt selection details are in Appendix C.

Predictions are evaluated against all reference LEX sets, and the best match determines the final

[6] LangPro analyzes the linguistic structures of the premise and hypothesis and converts them into logical forms that remain relatively close to the original sentences. To test whether the premise entails the hypothesis, it attempts to construct a tableau from the premise together with the negation of the hypothesis. The prover repeatedly applies logical, lexical, and monotonicity-based inference rules, and the entailment is established when every branch of the resulting tableau is closed by a contradiction. The resulting tableau constitutes a formal proof whose individual inference steps can be inspected.

[7] Feeding LangPro with lexical relations is straightforward because multi-word expressions such as **zoo keeper** and **small animal** are already terms in its higher-order logic, without need to represent them as, for example, **small**$(x) \wedge$ **animal**(x).

[8] The dataset and codebase are available from https://github.com/jorrytdejong/kbprojection.

score. We use two scores to evaluate models on labeling entailment problems and generating LEX sets. *Exact Match* (EM) counts a predicted LEX set correct when it is equal to one of the reference LEX sets. Because EM ignores the size of LEX, we additionally use *micro-F1* for finer-grained matching.[9]

Mismatch between lexical entailment relations can be roughly divided into normalization, structure, and semantic errors. *Normalization errors* concern wrong lemmatization of words in the relations. *Structure errors* include cases where two correctly extracted arguments are placed in the wrong order in a relation, i.e., in a swapped order for the entailment relation. *Semantic errors* cover mistakes that do not belong to the two aforementioned classes. To assess how much models struggle with normalization, we use *lemmatized F1* that eliminates this type of error by lemmatizing LEX sets before matching.[10] We also use *argument-unordered F1*, which ignores relation direction during matching, so that **guy** $\sqsubseteq$ **man** matches **man** $\sqsubseteq$ **guy**.

6.2 Supplying Knowledge to LangPro

For the extrinsic evaluation, we use an LLM to supply lexical entailments as a KB to LangPro while it searches for proofs of entailment problems. 1,000 randomly sampled entailment problems from the SNLI training set are used for this evaluation. We compare five ways of assisting LangPro with lexical KBs: only WordNet, only LLM-generated knowledge, and WordNet with LLM-generated knowledge, with the latter two settings evaluated both in one-shot and agentic use of LLMs.

In the one-shot setting, the LLM generates a KB once and LangPro attempts a proof using the KB. In the agentic setting, a failed proof triggers one refinement cycle. A critic prompt receives the premise-hypothesis pair together with LangPro's proof outcome, tableau closure information, and the attempted KB, and diagnoses possible missing or inappropriate relations. A subsequent LLM call uses this diagnosis to generate a revised KB, after which LangPro attempts the proof again. The experiment therefore tests whether feedback from the symbolic prover, followed by a critical reassessment of the generated KB, can improve the usefulness of LLM-generated lexical knowledge on a second attempt.

7 Results

7.1 Predicting Lexical Explanations

The evaluation results of the eight models on the curated 362 problems are given in Table 3. The evaluation is averaged over five runs, and the scores (with micro-F1 regarded as primary) are computed based on the best reference LEX sets. Since the proposed explainable NLI task requires models to predict an inference label together with a LEX set, any LEX set paired with a non-entailment prediction is considered distinct from a gold empty LEX. However, if a model's prediction for a problem is not available due to an error in the API call, we ignore such problems from the model evaluation (indicated in the support column).

In addition to the eight LLMs, we report results for three baselines. Empty LEX always returns an empty LEX set and serves as the simplest baseline. WordNet & Rules returns relations of the form $s_p \sqsubseteq s_h$ that are found in WordNet, where s_p and s_h are single- or multi-word lemma spans that occur only in the premise and hypothesis, respectively.[11] WordNet & Oracle peeks at the reference LEX sets and selects all and only those relations that are validated by WordNet lookup. It therefore represents a baseline that is perfect in selection of arguments, but limited by WordNet coverage.

Model performance ranking generally follows a simple size-based pattern, except for Gemini 3.1 Flash-Lite which achieves better scores than Gemini 3.5 Flash. Interestingly, Gemini 3.1 Flash-Lite also substantially outperforms Claude Haiku 4.5 and GPT-5.4 Mini, from a comparable product tier. GPT-5.4 Mini, GPT-OSS-20B, and Gemma 3 4B scored substantially lower than the remaining models, with Gemma 3 4B performing worst overall and falling below the WordNet & Rules baseline on

[9] We opt for micro-F1 because it gives greater weight to larger LEX sets and is less influenced by the 29% of examples with empty LEX sets than macro-F1.

[10] We POS-tag the original premise and hypothesis and locate words of each argument in either sentence. The corresponding tags are mapped to WordNet's noun, verb, adjective, or adverb categories, and NLTK's WordNet lemmatizer is used to get the lemma for a word-tag pair. Tokens for which this procedure fails remain unchanged. We also apply lemmatization to the reference relations because they contain plural-only nouns such as "glasses", "pants", and "shades", and this might increase the chance of a match.

[11] WordNet & Rules represents a realistic rule-based baseline using the heuristic that the first and second components of a relation typically come from the premise and hypothesis, respectively, while words shared by the premise and hypothesis are rarely part of such relations. It uses spaCy (Honnibal et al., 2020) for lemmatization and POS tagging, with POS tags used to select the corresponding WordNet senses. During WordNet lookup, we use the transitive closure of hypernymy, entailment, and similarity relations.

Model	Support	Exact match	Micro-F1				Micro-Precision	Micro-Recall
			Raw ↓	Δ Lem.	Δ Un.	Δ Lem.+Un.		
Claude Sonnet 4.5	100.0	**58.2** (± 0.2)	**73.1** (± 0.2)	+0.9	+0.6	+1.5	**64.5** (± 0.2)	**84.2** (± 0.3)
Gemini 3.1 Flash-Lite	100.0	57.9 (± 0.4)	72.3 (± 0.5)	+0.4	+0.0	+0.4	64.4 (± 0.4)	82.6 (± 0.6)
Gemini 3.5 Flash	100.0	58.1 (± 0.4)	71.9 (± 0.4)	+1.0	+0.4	+1.4	63.6 (± 0.4)	82.6 (± 0.5)
■ WordNet & Oracle	100.0	64.1	70.5	–	–	–	100.0	54.4
GPT-5.4	100.0	57.5 (± 1.0)	69.3 (± 0.4)	+0.7	+0.4	+1.1	61.0 (± 0.4)	80.2 (± 0.7)
Claude Haiku 4.5	100.0	50.6 (± 0.0)	67.2 (± 0.0)	+1.6	+0.0	+1.6	57.1 (± 0.0)	81.8 (± 0.0)
GPT-5.4 Mini	100.0	38.7 (± 1.2)	56.0 (± 1.1)	+7.3	+0.3	+7.8	46.3 (± 0.9)	71.0 (± 1.5)
GPT-OSS-20B	99.8	44.0 (± 1.1)	55.0 (± 0.9)	+4.5	+0.2	+4.7	48.8 (± 0.7)	63.0 (± 1.3)
■ WordNet & Rules	100.0	50.0	51.0	–	–	–	70.2	40.0
Gemma 3 4B	98.1	14.1 (± 0.3)	30.6 (± 0.3)	+11.5	+0.1	+12.1	21.4 (± 0.2)	53.8 (± 0.3)
■ Empty LEX	100.0	37.0	0.0	–	–	–	100.0	0.0

Table 3: LLMs' results on the LEX-NLI task. Values are reported as the mean (± sample standard deviation) across five runs. Results are sorted by raw Micro-F1, computed without post-processing. Δ Lem., Δ Un., and Δ Lem.+Un. report gains over raw Micro-F1 under lemmatized, argument-order-agnostic (A ⊑ B matches B ⊑ A), and combined scoring, respectively. Support is the percentage of problems evaluated; exclusions are due to error responses from API calls. ■ denotes baseline and oracle systems.

both EM and micro-F1, as well as below the Empty LEX baseline on EM.

The substantially higher recall than precision suggests systematic overgeneration. Across the top five models, predicted LEX sets average 1.4 relations, compared with 1.0 for the reference LEX sets. Relative to the references, models produce fewer empty sets (15% vs 29%) and more sets of size 2 (25% vs 14%) and 3+ (11% vs 6%). Due to the models' tendency to overgenerate relations, WordNet & Oracle provides a strong baseline. This suggests that selecting the correct lexical spans is a major challenge of the task.

Table 3 also reports micro-F1 under lemmatized and/or argument-order-agnostic matching. Ignoring inflectional differences increases micro-F1 for all models, but the gain is small for the top five models ($\leq$1.6 points) and substantially larger for the remaining models ($\geq$4.5 points). This suggests that producing lemmatized arguments is not a major difficulty for the better-performing models.

Ignoring argument order has a much smaller effect ($\leq$0.6 points). We hypothesize that this reflects a combination of models distinguishing more specific from more general concepts and exploiting the heuristic that the specific concepts usually occur in the premise and the general ones in the hypothesis. Note that among the eight entailment examples in the nine-shot prompt, we deliberately included one example that violates this heuristic (Example 7 in Appendix B). However, because entailment NLI problems rarely include such adversarial cases, we included only one example in the prompt.

Overall, the results show substantial room for improvement on the LEX-NLI task. The best model trails annotator performance by 22.7 points in EM and 11.8 points in micro-F1. Even under the more model-friendly evaluation, where problems classified as non-entailment are excluded rather than penalized (see Table 5 in Appendix D), the best scores remain below average annotator performance, with gaps of 13.9 percentage points in EM and 7.1 in micro-F1.[12]

7.2 Using LEX to Prove Entailments

Table 4 reports the number of entailment proofs constructed by LangPro for 1,000 SNLI entailment problems. When using only WordNet as its lexical KB, LangPro proves 157 problems. We additionally evaluate four configurations in which the lexical KB is generated by an LLM either in one-shot mode (LLM_1) or agentic mode (LLM_a), with the generated KB optionally complemented by WordNet (LLM_1^{WN} and LLM_a^{WN}, respectively). Due to budget constraints, we conduct this experiment with five small models.

The extrinsic evaluation results show that the model ranking differs from that observed on the LEX-NLI task. While Gemini 3.1 Flash-Lite and Claude Haiku 4.5 remain the strongest models in the LLM_1 setting, they are outperformed by other models in the remaining three configurations.

The relatively small gains of LLM_1 over the LangPro+WordNet baseline (see Δ_{WN} values in

[12] Additionally, while prompt engineering was not extensive, the selection was biased toward the test set; see Appendix C.

Model	LLM_1 Δ_{WN}	LLM_a Δ_{WN} Δ_a	LLM_1^{WN} Δ_{WN}	LLM_a^{WN} Δ_{WN} Δ_a
Gemini 3.1 Flash-Lite	**165** +8	180 +23 +15	179 +22	193 +36 +14
Claude Haiku 4.5	**165** +8	176 +19 +11	179 +22	190 +33 +11
GPT-5.4 Mini	162 +5	183 +26 +21	178 +21	199 +42 +21
GPT-OSS-20B	158 +1	**194** +37 +36	174 +17	**212** +55 +38
Gemma 3 4B	163 +6	163 +6 +0	**182** +25	182 +25 +0

Table 4: LangPro proof coverage on 1,000 SNLI entailment problems when using various sources of the lexical KB. LLM_1 and LLM_a denote one-shot and agentic LLM-generated LEX, while LLM_1^{WN} and LLM_a^{WN} are the former two coupled with WordNet, respectively. Values indicate the number of problems proved as entailment. The LangPro+WordNet baseline proves 157 problems. The boxed Δ_{WN} values indicate gains over LangPro+WordNet, while Δ_a values indicate the additional gains from agentic inference over the corresponding non-agentic LLM.

LLM_1) do not appear to arise simply because the LLM-generated relations are largely subsumed by WordNet. When combined with WordNet, each model still generates relations enabling additional proofs, see the Δ_{WN} values in the LLM_1^{WN} and LLM_a^{WN} columns. In the one-shot setting with WordNet (LLM_1^{WN}), Gemma 3 4B performs best, with 182 proofs, whereas GPT-OSS-20B performs best in the agentic setting LLM_a^{WN}, with 212 proofs.

Agentic generation improves proof coverage for all models except Gemma 3 4B, indicating that Gemma cannot successfully revise the relations produced in its initial attempt. GPT-OSS-20B benefits most from agentic generation, improving over the corresponding one-shot configurations by 36 proofs without WordNet and 38 proofs with WordNet. It consequently achieves the best overall result, with 212 proofs in the LLM_a^{WN} configuration.

We also conducted an extrinsic evaluation on the LEX test set (see Table 6 in Appendix D). The performance gaps between models are substantially narrower than on LEX-NLI, indicating that the extrinsic evaluation is coarser-grained and captures a different aspect of model quality. The LEX test set also appears easier: proof coverage is typically around 1.5-1.9 times as high as on the 1,000-problem SNLI sample, while gains over the WordNet baseline are substantially larger. In the LLM_a^{WN} setting, all but one model exceed the 34% accuracy of the LangPro-LEX oracle, where LangPro is supplied with the union of the reference LEX sets. This suggests that, despite suboptimal intrinsic performance on LEX-NLI, LLM-generated lexical knowledge can have an extrinsic effect comparable to the curated reference LEX sets.

8 Analysis

8.1 Predicting Lexical Explanations

We examine the NLI problems and the predicted LEX sets to understand the type of mistakes models make. For 33 problems (9%), no model predicts any of the reference LEX sets exactly. Our analysis of these cases yields three main observations.

Overgeneration of variants: Models frequently overgenerate variants of the same semantic relation (in roughly half of the cases), most often differing only in inflection. Examples include **shades** $\sqsubseteq$ **glasses** alongside **shade** $\sqsubseteq$ **glass**, **cartoon** $\sqsubseteq$ **animated** alongside **cartoon** $\sqsubseteq$ **animate**, and **stereo equipment** $\sqsubseteq$ **audio equipment** alongside **stereo** $\sqsubseteq$ **audio**. Such overgeneration lowers precision and micro-F1 but does not affect recall. The extent of this penalty is reflected in the improvement obtained with lemmatized micro-F1 (Table 3). Additional prompt instructions could likely reduce variant overgeneration, although this would not improve recall.

Overgeneration of phrasal relations: LEX-NLI specifically targets lexical relations and excludes phrasal ones. For example, for the entailment snippets "man painting a picture" and "man is creating art", it expects the explanation {**paint** $\sqsubseteq$ **create**, **picture** $\sqsubseteq$ **art**}. However, no model predicted either relation, often opting instead for phrasal relations such as **paint** $\sqsubseteq$ **create art** or **paint picture** $\sqsubseteq$ **create art**, more natural in this case. Although the lexical-phrasal distinction is relatively clear from a syntactic perspective, the annotation guidelines, which were also used as the prompt, deliberately avoided syntax-heavy terminology.

Issues with references: Several correct predictions were unfairly penalized because the references contained wrong relations instead of correct

ones (**look out** $\sqsubseteq$ **view** instead of **look out over** $\sqsubseteq$ **view**), omitted valid ones (**make** $\sqsubseteq$ **cause** as in "make laugh" and "cause to be merry"), or lacked acceptable variants (having **decorated** $\sqsubseteq$ **nice**, but not **highly decorated** $\sqsubseteq$ **nice**).

8.2 Supplying Knowledge to LangPro

To assess the additional contribution of the agentic mode over one-shot generation, we manually analyzed the 15/14 additional problems solved by LLM_a/LLM_a^{WN} compared with LLM_1/LLM_1^{WN} for Gemini 3.1 Flash-Lite. The added relations for these problems are shown in Table 8 and Table 7, respectively, in Appendix D. Only five of the analyzed problems contain at least one relation we judged semantically reasonable in isolation. In the remaining cases, the additional relations help LangPro prove the problem either by compensating for errors introduced by the CCG parser or by exploiting weaknesses in LangPro's representation or proof procedure. For example, for the problem P: "Two skiers stand near a snowy mountain." and H: "Some humans standing.", LangPro fails to construct a proof even when LLM_1^{WN} provides **skier** $\sqsubseteq$ **human**. This is because "standing" in the hypothesis is analyzed as a noun with lemma "standing" by the NLP tools used in LangPro's syntactic pipeline. When LLM_a^{WN} additionally proposes **stand** $\sqsubseteq$ **standing** and **standing** $\sqsubseteq$ **stand**, LangPro succeeds; the resulting proof requires only **stand** $\sqsubseteq$ **standing** together with **skier** $\sqsubseteq$ **human**. Another type of tool error that these relations can compensate for involves confusing particles and prepositions. Such cases illustrate that the agentic mode can improve proof success by repairing parser-induced lexical mismatches when supplying reasonable lexical knowledge is not enough.

9 Conclusion & Future Work

To our knowledge, this is the first work that augments existing naturalistic NLI problems with structured lexical explanations of the knowledge required for their reasoning. Unlike prior lexical relation detection work (Vulić et al., 2017; Schmitt and Schütze, 2019; Shwartz and Dagan, 2016), among others, our task requires identifying only the relations needed to explain a particular inference. We operationalize these explanations as LEX sets and construct a manually annotated test set that retains multiple references when necessary.

Our intrinsic evaluation on LEX-NLI shows that LLMs can recover much of the required lexical knowledge, but the task remains challenging. Under the primary evaluation, the best-performing LLM reaches 73.1 micro-F1 and 58.2 EM (Table 3), trailing average annotator performance by 11.8 and 22.7 points, respectively. Under the more model-friendly evaluation, which excludes problems predicted as non-entailment, the best per-metric scores increase to 77.8 micro-F1 and 67.0 EM (Table 5), reducing the corresponding gaps to 7.1 and 13.9 points. Models particularly struggle with overgenerating relations, including inflectional variants, and with selecting minimal argument spans that constitute valid lexical entities.

Our extrinsic evaluation shows that LLM-generated KBs can enable LangPro, a natural-logic theorem prover, to find additional formal proofs. LLMs can supply lexical relations that are absent from WordNet, while the agentic setting, which revises an initial KB once, further improves proof coverage. However, much of this additional gain comes from semantically unsound relations that compensate for errors in the syntactic analysis of the input. On the LEX test set, the agentic setting even surpasses LangPro supplied with oracle reference relations, illustrating that higher proof coverage does not necessarily imply better lexical knowledge.

Overall, the results support a neuro-symbolic division of labor in which LLMs supply flexible lexical knowledge and symbolic systems perform explicit reasoning, while structured lexical evaluation provides an independent check on whether the supplied knowledge genuinely explains the inference.

As future work, we plan to scale up the LEX-NLI task to cover all SICK problems and a larger portion of SNLI. To make this feasible, we will combine manual validation with automatic candidate-LEX detection using LangPro supplied with LLM-generated lexical knowledge. As the analysis of model outputs in § 8.1 reveals, a few overgenerated relations are in fact valid; we therefore plan to use them to refine the reference annotations and improve their completeness as we develop the dataset. In the longer term, we aim to extend structured NLI explanations beyond entailment to other NLI labels and to richer explanatory structures that capture reasoning beyond lexical entailment as outlined in Abzianidze (2025).

Limitations

Our study is restricted to NLI problems with an entailment label. This choice allows us to study lexical entailments in a relatively well-defined setting, but leaves contradiction and neutral cases unexplored. Extending structured explanations to these labels is less straightforward, as their reasoning may depend on incompatibility, negation, missing information, or pragmatic assumptions rather than directed lexical entailments alone.

We also deliberately restrict explanations to lexical entailments over words and multi-word concepts. Such relations often constitute an important component of entailment reasoning, but they do not provide a complete account of the inference. LEX sets abstract away from syntactic reasoning, argument structure, quantification, coreference, event relations, and other compositional or world-knowledge phenomena. Our results should therefore be interpreted as evaluating one crucial structured component of explanation rather than complete explanations of NLI reasoning.

Finally, the annotation task itself also admits genuine explanatory variation, as evidenced by cases in which several LEX variants remained valid after adjudication. Although our multi-reference evaluation (with the 4th LEX added during the annotation adjudication phase) accounts for the retained alternatives, the reference set may not exhaust all semantically acceptable explanations.

Acknowledgments

We thank the three anonymous reviewers for their constructive feedback and helpful suggestions. This work was supported by the Dutch Research Council (NWO) through the project *Towards Explainable Natural Language Reasoning for Artificial Intelligence* (project number DXQIL95775). This paper grew out of a student project conducted within the course *Reasoning with Natural Language* (TLMV25020) at Utrecht University.

References

Lasha Abzianidze. 2017a. Langpro: Natural language theorem prover. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 115–120, Copenhagen, Denmark. Association for Computational Linguistics.

Lasha Abzianidze. 2017b. *A natural proof system for natural language*. Ph.D. thesis, Tilburg University.

Lasha Abzianidze. 2020. Learning as abduction: Trainable natural logic theorem prover for natural language inference. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 20–31, Barcelona, Spain (Online). Association for Computational Linguistics.

Lasha Abzianidze. 2025. Formal proofs as structured explanations: Proposing several tasks on explainable natural language inference. *Preprint*, arXiv:2311.08637.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *Preprint*, arXiv:2204.06031.

Rami Aly, Marek Strong, and Andreas Vlachos. 2023. QA-NatVer: Question answering for natural logic-based fact verification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8376–8391, Singapore. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: natural language inference with natural language explanations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9560–9572, Red Hook, NY, USA. Curran Associates Inc.

Samuel Carton, Surya Kanoria, and Chenhao Tan. 2022. What to learn, and how: Toward effective learning from rationales. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1075–1088, Dublin, Ireland. Association for Computational Linguistics.

Zeming Chen, Qiyue Gao, and Lawrence S. Moss. 2021. NeuralLog: Natural language inference with joint neural and logical reasoning. In *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 78–88, Online. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, Research in Computational Semantics, pages 177–190. Springer-Verlag.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. Mit Press.

Octavian Ganea, Gary Becigneul, and Thomas Hofmann. 2018. Hyperbolic entailment cones for learning hierarchical embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1646–1655. PMLR.

Mohammad Reza Ghasemi Madani and Pasquale Minervini. 2023. REFER: An end-to-end rationale extraction framework for explanation regularization. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 587–602, Singapore. Association for Computational Linguistics.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. FOLIO: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Honoka Kobayashi, Hinari Daido, and Daisuke Bekki. 2026. Neural DTS: Integrating hyperbolic classifiers into natural language inference systems. In *Proceedings of the 6th Workshop on Natural Language Meets Logic and Machine Learning (NALOMA)*, pages 9–18, Prague, Czechia. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Bill Noble, Rasmus Blanck, and Gijs Wijnholds. 2025. In the mood for inference: Logic-based natural language inference with large language models. In *Proceedings of the 5th Workshop on Natural Logic Meets Machine Learning (NALOMA)*, pages 33–47, Bochum, Germany. Association for Computational Linguistics.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176, Singapore. Association for Computational Linguistics.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Hyun Ryu, Gyeongman Kim, Hyemin S. Lee, and Eunho Yang. 2025. Divide and translate: Compositional first-order logic translation and verification for complex logical reasoning. In *The Thirteenth International Conference on Learning Representations*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Martin Schmitt and Hinrich Schütze. 2019. SherLIiC: A typed event-focused lexical inference benchmark for evaluating natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 902–914, Florence, Italy. Association for Computational Linguistics.

Vered Shwartz and Ido Dagan. 2016. Adding context to semantic data-driven paraphrasing. In *Proceedings*

of the Fifth Joint Conference on Lexical and Computational Semantics, pages 108–113, Berlin, Germany. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence*, pages 4444–4451.

Joe Stacey, Pasquale Minervini, Haim Dubossarsky, and Marek Rei. 2022. Logical reasoning with span-level predictions for interpretable and robust NLI models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3809–3823, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Marek Strong, Rami Aly, and Andreas Vlachos. 2024. Zero-shot fact verification via natural logic and large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 17021–17035, Miami, Florida, USA. Association for Computational Linguistics.

Akiyoshi Tomihari and Hitomi Yanaka. 2023. Logic-based inference with phrase abduction using vision-and-language models. *IEEE Access*, 11:45645–45656.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2071–2080. JMLR.org.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. 2017. HyperLex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835.

Hitomi Yanaka, Koji Mineshima, Pascual Martínez-Gómez, and Daisuke Bekki. 2018. Acquisition of phrase correspondences using natural deduction proofs. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 756–766, New Orleans, Louisiana. Association for Computational Linguistics.

Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji, and Daisuke Bekki. 2019. Combining axiom injection and knowledge base completion for efficient natural language inference. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Xinyan Zhao and V. G. Vinod Vydiswaran. 2021. Lirex: Augmenting language inference with relevant explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14532–14539.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2022. Analytical reasoning of text. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2306–2319, Seattle, United States. Association for Computational Linguistics.

A LLM Inference Setup

The models were accessed through the OpenRouter Chat Completions interface. The model identifiers record the versions requested from OpenRouter, but they are not pinned provider snapshots. The underlying snapshots were not retained in the response metadata. In the final model comparison, every model was evaluated on the test data five times using the same nine-example few-shot prompt (see Appendix B). Repetition was used to quantify variation caused by nondeterministic model and provider behaviour. A temperature of 0 was requested, but this did not guarantee identical outputs because temperature support was not uniform across the evaluated OpenRouter routes and the setting could be ignored by a provider. Requests were issued with a concurrency of four, a 120-second timeout, and at most two retries. No common `top_p`, random seed, reasoning-effort setting, stop sequence, or explicit output-token limit was supplied. The total budget of the experiments was under $80.

B Annotation Guidelines and LLM Prompt

The following instructions were used both as annotation guidelines for constructing the gold-standard structured explanations and as the prompt given to the evaluated LLMs. The placeholders `$PREMISE` and `$HYPOTHESIS` were replaced with the premise and hypothesis of each NLI problem.

Following § 3, the guidelines define a LEX as the smallest set of directed lexical entailment relations needed to explain how the hypothesis follows from the premise, when it does. The instructions require short, factually acceptable argument phrases and emphasize four properties: (a) *minimality* requires the model to omit relations unnecessary for the inference or that follow merely by discarding a modifier. (b) *lemmatization* requires canonical word forms without determiners, auxiliary verbs, or the infinitival marker *to*. (c) *relation direction* requires the first argument to license the second in the direction needed for the inference. (d) *output formatting* requires the prescribed format for non-empty relation sets.

Lexical-entailment annotation prompt

You are an expert in linguistic semantics and logic. You will receive a Natural Language Inference (NLI) problem in English, consisting of a premise sentence and a hypothesis sentence.
You will reason carefully and decide whether the premise entails the hypothesis. Entailment means that if the premise is true, then the hypothesis must also be true under ordinary English meaning and widely accepted background knowledge.
If the answer is `entailment`, output a structured explanation consisting of a set of lexical entailment relations over short phrases that explain why the hypothesis is entailed by the premise. A lexical entailment must be defined over short phrases that are lemmatized or normalized versions of expressions occurring in the premise and hypothesis. Use the format **phrase_1 ⊑ phrase_2**. The relation means that *phrase_1* is a type of, or semantically entails, *phrase_2*. Examples include **woman ⊑ person**, **dog ⊑ domestic animal**, **huge ⊑ very big**, and **run ⊑ move fast**.
Use lexical entailment relations only when they are needed to explain the entailment. If the entailment follows without any non-trivial lexical relation, output an empty set.

Relation-formatting rules

1. The meaning of a lexical entailment relation must be acceptable according to ordinary common-sense knowledge. For example, **woman ⊑ blond person** is not acceptable.
2. A lexical entailment may not express a trivial relation that can be obtained simply by discarding modifiers. For example, **blond woman ⊑ woman** is not acceptable.
3. A lexical entailment may not contain redundant words such as auxiliary verbs or the infinitival marker *to*. For example, **will walk ⊑ will move**, **is red ⊑ is colored**, and **to walk ⊑ to move** are not acceptable.
4. The words in each phrase must be lemmatized. For example, **dogs ⊑ domestic animals** is not acceptable.
5. Phrases may not contain determiners. For example, **a dog ⊑ a domestic animal** is not acceptable.
6. Phrases may not contain prepositional phrases. For example, **dog with spots ⊑ domestic animal** is not acceptable.

Example 1
Premise: *Young ladies are playing the guitar.*
Hypothesis: *A musical instrument is being played by girls.*
Correct output:
Answer: `entailment`
Relations: {**young lady ⊑ girl**, **guitar ⊑ musical instrument**}
Unwanted output:
Relations: {**young ladies ⊑ girls**, **the guitar ⊑ a musical instrument**}
Explanation: Phrases in relations may not contain determiners such as *a* and *the*. The words *ladies* and *girls* must be replaced by their lemmas, *lady* and *girl*, respectively.

Example 2
Premise: *A female swimmer getting out of the pool still dripping wet.*
Hypothesis: *A woman gets out of the pool.*
Correct output:
Answer: `entailment`
Relations: {**female swimmer ⊑ woman**}
Unwanted output:

Relations: {**swimmer** ⊑ **woman**}
Explanation: The proposed relation is factually incorrect because not every swimmer is a woman.

Example 3
Premise: *A young girl wearing a pink coat plays with a yellow toy.*
Hypothesis: *A kid is swinging a toy golf club.*
Correct output:
Answer: `non-entailment`

Example 4
Premise: *A black race car starts up in front of a crowd of people.*
Hypothesis: *A car is running.*
Correct output:
Answer: `entailment`
Relations: {**start up** ⊑ **run**}
Unwanted output:
Relations: {**starts up** ⊑ **is running**}
Explanation: The expressions *starts up* and *is running* are not lemmatized. The auxiliary verb *is* is also unnecessary.

Example 5
Premise: *A woman dressed in red clothing is dancing inside a crowd of people.*
Hypothesis: *A woman in red is dancing in a crowd.*
Correct output:
Answer: `entailment`
Relations: {}
Unwanted output:
Relations: {**in red clothing** ⊑ **in red**}
Explanation: The arguments of a relation may not contain prepositional phrases such as *in red clothing*.

Example 6
Premise: *A tall man with a cap is climbing a cord.*
Hypothesis: *The man in a hat is climbing a rope.*
Correct output:
Answer: `entailment`
Relations: {**cap** ⊑ **hat**, **cord** ⊑ **rope**}
Unwanted output:
Relations: {**cap** ⊑ **hat**, **tall man** ⊑ **man**}
Explanation: The relation **cord** ⊑ **rope** is missing. The relation **tall man** ⊑ **man** is trivial because it only discards the modifier *tall*.

Example 7
Premise: *No person is cooking.*
Hypothesis: *No cook is cooking in the kitchen.*
Correct output:
Answer: `entailment`
Relations: {**cook** ⊑ **person**}
Unwanted output:
Relations: {**person** ⊑ **cook**}
Explanation: The unwanted relation does not explain the inference because negation reverses the permitted direction of lexical substitution. It is also factually incorrect because not every person is a cook.

Example 8
Premise: *A person who is obese is holding a chinchilla.*
Hypothesis: *A fat person is holding a small animal.*
Correct output:
Answer: `entailment`
Relations: {**chinchilla** ⊑ **small animal**, **obese** ⊑ **fat**}
Unwanted output:
Relations: {**fat** ⊑ **obese**, **chinchilla** ⊑ **animal**}
Explanation: The arguments of **fat** ⊑ **obese** must be reversed to align the relation with the direction of the inference. The relation **chinchilla** ⊑ **animal** is insufficient because it does not account for the adjective *small*, which is crucial in the hypothesis.

Example 9
Premise: *A little boy is laughing and happily bouncing on a trampoline outside.*
Hypothesis: *The child is jumping outdoors.*
Correct output:
Answer: `entailment`
Relations: {**little boy** ⊑ **child**, **bounce** ⊑ **jump**, **outside** ⊑ **outdoors**}
Unwanted output:
Relations: {**boy** ⊑ **child**, **outdoors** ⊑ **outside**}
Explanation: The relation **bounce** ⊑ **jump** is missing. The relation **little boy** ⊑ **child** is preferred to **boy** ⊑ **child** because the former provides a more precise correspondence between the expressions in the premise and hypothesis. The arguments of **outdoors** ⊑ **outside** must be reversed to align the relation with the direction of the inference.

Additional calibration while preserving all rules above:

- Only output lexical entailment relations that are both factually acceptable and needed to explain the entailment.
- Do not output relations for entailments that follow without a non-trivial lexical bridge.
- Do not use event or social implications as lexical entailment unless the phrase relation is a direct paraphrase.
- Do not output modifier-dropping relations such as `old woman` → `woman` or `military men` → `men`.
- Do not output both an inflected form and a lemmatized form for the same relation.
- If the best relation would violate any existing formatting rule, omit it instead of approximating it.

Now process the following input while strictly following the above instructions and formatting.
Output exactly in one of the following formats:

```
Answer: entailment
Relations: { . . . }
or
Answer: non-entailment
```

Input:

```
Premise: $PREMISE
Hypothesis: $HYPOTHESIS
```

Correct output:

C Prompt Selection

Preliminary prompt-selection experiments compare four controlled variants on the unanimously agreed samples (223 out of 362) using four models: GPT-5.4 mini, Claude Haiku 4.5, Gemini 3.1 Flash-Lite, and GPT-OSS-20B. Prompt variants are constructed by adding sections to the same base prompt. The *zero-shot base* condition contains only the task definition. The *zero-shot constrained* con-

dition adds the relation-formatting rules from the annotation guidelines. The *few-shot base* condition further adds nine worked examples, including correct and unwanted outputs. Finally, the *few-shot precision* condition adds a calibration block discouraging speculative or unnecessary relations, modifier deletion, and duplicate surface and lemma forms. For each prompt-model combination, we calculate multi-reference micro-F1. We then compute the unweighted mean of the four model-level micro-F1 scores for each prompt condition: 43.8 for *zero-shot base*, 53.5 for *zero-shot constrained*, 65.2 for *few-shot base*, and 69.2 for *few-shot precision*. Based on these results, we select the *few-shot precision* prompt for the final experiments.

D Mixture of Figures and Tables

Model	Support	Exact match	Micro-F1				Micro-Precision	Micro-Recall
			Raw ↓	Δ Lem.	Δ Un.	Δ Lem.+Un.		
Gemini 3.1 Flash-Lite	89.0	65.1 (± 0.5)	**77.8** (± 0.6)	+0.4	+0.0	+0.4	**70.7** (± 0.5)	86.6 (± 0.7)
Claude Sonnet 4.5	91.7	63.5 (± 0.2)	76.9 (± 0.2)	+1.0	+0.6	+1.5	69.2 (± 0.2)	**86.7** (± 0.3)
GPT-5.4	85.7	**67.0** (± 0.9)	76.6 (± 0.2)	+0.8	+0.4	+1.2	68.8 (± 0.4)	86.4 (± 0.2)
Gemini 3.5 Flash	95.2	61.0 (± 0.5)	74.3 (± 0.5)	+1.1	+0.4	+1.5	66.0 (± 0.5)	85.0 (± 0.5)
Claude Haiku 4.5	89.0	56.8 (± 0.0)	72.1 (± 0.0)	+1.7	+0.0	+1.7	62.2 (± 0.0)	85.8 (± 0.0)
■ WordNet & Oracle	100.0	64.1	70.5	–	–	–	100.0	54.4
GPT-OSS-20B	77.5	56.7 (± 1.2)	66.9 (± 0.7)	+5.5	+0.2	+5.7	60.1 (± 0.8)	75.3 (± 0.9)
GPT-5.4 Mini	86.9	44.5 (± 1.1)	60.9 (± 0.8)	+7.9	+0.3	+8.4	50.8 (± 0.7)	75.8 (± 1.0)
■ WordNet & Rules	100.0	50.0	51.0	–	–	–	70.2	40.0
Gemma 3 4B	92.5	14.9 (± 0.4)	31.3 (± 0.3)	+11.8	+0.1	+12.4	21.9 (± 0.2)	55.0 (± 0.3)
■ Empty LEX	100.0	37.0	0.0	–	–	–	100.0	0.0

Table 5: LLMs' results on the LEX-NLI task. These scores are more favorable to the models, as examples assigned a non-entailment label are omitted from evaluation; in contrast, Table 3 penalizes predictions on those examples. Support is the percentage of problems evaluated; exclusions mostly result from models labeling problems as non-entailment. ■ denotes baseline and oracle systems.

Model	LLM_1 %		LLM_a %			LLM_1^{WN} %		LLM_a^{WN} %		
		Δ_{WN}		Δ_{WN}	Δ_a		Δ_{WN}		Δ_{WN}	Δ_a
Claude Sonnet 4.5	25.4	+7.2	32.0	+13.8	+6.6	30.1	+11.9	35.4	+17.1	+5.2
Gemini 3.1 Flash-Lite	24.9	+6.6	31.8	+13.5	+6.9	29.3	+11.0	34.8	+16.6	+5.5
Gemini 3.5 Flash	23.5	+5.2	**33.4**	+15.2	+9.9	28.2	+ 9.9	**37.0**	+18.8	+8.8
GPT-5.4	24.3	+6.1	30.7	+12.4	+6.4	29.3	+11.0	35.6	+17.4	+6.4
Claude Haiku 4.5	**25.7**	+7.5	31.2	+13.0	+5.5	**30.9**	+12.7	35.6	+17.4	+4.7
GPT-5.4 Mini	23.8	+5.5	29.3	+11.0	+5.5	29.8	+11.6	34.8	+16.6	+5.0
Gemma 3 4B	25.1	+6.9	25.1	+ 6.9	+0.0	30.7	+12.4	30.7	+12.4	+0.0

Table 6: LangPro proof coverage on the **test set** (of 362 entailment problems) when using various sources of the lexical KB. LLM and LLM_a denote one-shot and agentic LLM-generated LEX, while LLM_1^{WN} and LLM_a^{WN} are the former two coupled with WordNet, respectively. Values indicate the percentage of problems proved as entailment. The LangPro+WordNet baseline proves 66 problems (18.2%). The boxed Δ_{WN} values indicate gains over LangPro+WordNet, while Δ_a values indicate the additional gains from agentic inference over the corresponding non-agentic LLM. GPT-OSS-20B is omitted because the model host was unresponsive when the experiments were conducted.

Premise	Hypothesis	Added/changed relations	Judgment
Two skiers stand near a snowy mountain.	Some humans standing.	stand ⊑ standing; standing ⊑ stand	Poor: inflection only; no new lexical knowledge
A black dog is running along the beach.	a dog is running on the beach	black dog ⊑ dog; running along ⊑ running on; run along ⊑ run on; along ⊑ on	Poor: modifier dropping plus invalid general substitution
Safety officers standing outside.	Safety officers outdoors.	standing outside ⊑ outdoors; stand outside ⊑ outdoors	Poor: semantically plausible but non-minimal
A female is wearing a big purple flower in her hair and has some beads around her neck.	a girl with a flower in her hair	wear ⊑ with; have ⊑ with	Good: unusual pairs but specific to SNLI & SICK datasets
A man wearing a black hat, white shirt, red vest, black tie, and face makeup is swallowing a sword, while a man wearing a cowboy hat behind him is watching.	A man is performing.	swallow ⊑ perform	Poor: valid only in the staged context
A little girl is petting her white, fluffy cat.	The girl has a pet.	little girl ⊑ girl; petting ⊑ has; pet ⊑ have	Poor: modifier dropping plus false possession inference
Little kids are on an amusement park ride.	Children are on a ride.	little kids ⊑ children; little kid ⊑ children; amusement park ride ⊑ ride	Poor: unnormalized, non-minimal modifier dropping
Man in blue shirt playing water instrument in an orchestra.	The man is wearing blue	in ⊑ wearing; in ⊑ wear	Poor: valid only in a clothing construction
A man on an elevator looks at the mirror on the wall.	A person in the elevator looks at a mirror.	on ⊑ in; mirror on the wall ⊑ mirror	Poor: invalid preposition change plus modifier dropping
A child wearing a rainbow striped shirt is pushing a toy cart.	A child has a toy.	push ⊑ have	Poor: pushing does not entail possession
Young blond woman putting her foot into a water fountain	A person is dipping her foot into water.	put into ⊑ dip into; put ⊑ dip	Poor: valid only for the foot–water context
The blonde tennis player is wearing a white shirt and a green tennis skirt.	The tennis playing is wearing a green skirt.	tennis player ⊑ tennis playing; tennis player ⊑ tennis play; player ⊑ playing	Poor: participant–event category mismatch
Adjusting a tie for accuracy.	Fixing a tie.	adjusting ⊑ fixing; adjusting for accuracy ⊑ fixing; adjust for accuracy ⊑ fix	Poor: unnormalized and context-dependent
The bird's feet are grasping the window tightly.	there is a bird resting	grasp ⊑ rest	Poor: grasping does not entail resting

Table 7: Quality judgments for the 14 additional problems obtained by Gemini 3.1 Flash-Lite in $\mathrm{LLM}_a^{\mathrm{WN}}$ agentic refinement over $\mathrm{LLM}_1^{\mathrm{WN}}$. We judge 1 case good, 0 mixed, and 13 poor.

Premise	Hypothesis	Added/changed relations	Judgment
A black dog is walking through a stream of water.	An animal is walking through water.	black dog ⊑ animal; stream ⊑ water	Good: both relations are generally valid; black dog ⊑ animal is non-minimal but sound
Two skiers stand near a snowy mountain.	Some humans standing.	stand ⊑ standing; standing ⊑ stand	Poor: inflection only; no new lexical knowledge
A black dog is running along the beach.	a dog is running on the beach	black dog ⊑ dog; running along ⊑ running on; run along ⊑ run on; along ⊑ on	Poor: modifier dropping plus invalid general substitution
An Asian woman in a floral wedding dress poses on a bridge close to her bridesmaids.	An asian women is wearing a wedding dress.	woman ⊑ women; women ⊑ woman; in ⊑ wear; floral wedding dress ⊑ wedding dress; floral wed dress ⊑ wed dress	Poor: inflection, context dependence, modifier dropping, and malformed forms
A lady walking down stairs with a green poster pod on her back.	A woman walking down stairs with a green thing on her back.	lady ⊑ woman	Good: generally valid lexical entailment
Safety officers standing outside.	Safety officers outdoors.	standing outside ⊑ outdoors; stand outside ⊑ outdoors	Poor: semantically plausible but non-minimal
A man wearing a black hat, white shirt, red vest, black tie, and face makeup is swallowing a sword, while a man wearing a cowboy hat behind him is watching.	A man is performing.	swallowing ⊑ performing; swallow ⊑ perform	Poor: context-dependent and partly unlemmatized
A black dog runs in the grass, its tongue hanging out.	The dog is outdoors.	run ⊑ outdoors	Poor: running does not entail being outdoors
A little girl is petting her white, fluffy cat.	The girl has a pet.	petting ⊑ having; pet ⊑ have	Poor: petting does not entail possession
Little kids are on an amusement park ride.	Children are on a ride.	little kids ⊑ children; little kid ⊑ children; amusement park ride ⊑ ride	Poor: unnormalized, non-minimal modifier dropping
A child wearing a rainbow striped shirt is pushing a toy cart.	A child has a toy.	push ⊑ have	Poor: pushing does not entail possession
A guy kneeling down in front of a desk.	A man bending down in front of a piece of furniture.	desk ⊑ furniture	Good: valid normalized hypernym, but not new semantic knowledge over one-shot
An asian lady in a brown shirt with glasses is making some food.	A woman is cooking	lady ⊑ woman; making ⊑ cooking; make ⊑ cook	Mixed: one valid relation; two food-context-dependent mappings
Adjusting a tie for accuracy.	Fixing a tie.	adjusting ⊑ fixing; adjusting for accuracy ⊑ fixing; adjust for accuracy ⊑ fix	Poor: unnormalized and context-dependent
The bird's feet are grasping the window tightly.	there is a bird resting	grasp ⊑ rest	Poor: grasping does not entail resting

Table 8: Quality judgments for the 15 additional problems obtained by Gemini 3.1 Flash-Lite in LLM_a agentic refinement over LLM_1. We judge 3 cases good, 1 mixed, and 11 poor.

LMKG: a flexible method for ontology-aware Knowledge Graph Extraction from text

Gianmarco Pappacoda[1] **and Paolo Torroni**[1]

[1]DISI, University of Bologna
{gianmarco.pappacoda, p.torroni}@unibo.it

Abstract

Knowledge Graphs (KGs) are a practical representation of information that combines structural flexibility with expressiveness. With the recent advent of Large Language Models (LLMs), it has become possible to extract KGs from plain text. However, extracting facts from text is only one part of building a usable knowledge graph. A construction system must also reconcile entity mentions, control its relation vocabulary, integrate information across passages, and preserve a traceable connection between graph facts and their source. We present LMKG, a modular LLM-based pipeline that supports both open and ontology-constrained extraction. We also propose two construction strategies for long documents: *fold*, which accumulates a graph over successive chunks, and *enrich*, which combines whole-document and chunk-level extraction before graph refinement. We evaluate LMKG on three standard benchmarks: Text2KGBench, MINE-1, and HotpotQA. We find that our ontology conditioning substantially improves conformance to a given ontology in Text2KGBench. Our system consistently recovers facts, reaching the highest reported fact recovery of 90.80% in MINE-1. We observe degraded downstream question-answering performance on HotpotQA when supplying graphs rather than full passages.

1 Introduction

Most of the knowledge in the world is stored as unstructured text. News and scientific articles, technical documentation, and online social media texts all fall into this category. *Knowledge Graphs* (KGs) offer a different, structured representation of such information. The information in KGs usually follows an *ontology*: a schema of the types and relations allowed in the graph. This representation is auditable and presents textual facts in a structured form that can also be used by retrieval-augmented generation systems (Lewis et al., 2020). Nevertheless, most general-purpose knowledge graphs, such as DBpedia, YAGO, and Wikidata (Auer et al., 2007; Suchanek et al., 2007; Vrandečić and Krötzsch, 2014), are incomplete and difficult for humans to inspect without purpose-built software. By contrast, small, high-quality, hand-curated graphs are more focused on a specific task but remain expensive to construct.

With the rise of LLMs, it has become possible to tackle the problem of text-to-KG extraction without employing expensive hand-crafted ontologies. Classical closed Information Extraction (cIE) assumes a predefined inventory of entity and relation types, often derived from an existing KG or ontology. Such constraints produce a consistent representation, but make the extractor difficult to transfer to new domains and unable to express facts outside its schema. Open Information Extraction (OpenIE) instead derives entities and relations directly from text (Angeli et al., 2015). Recent language models make this setting particularly attractive because they can generate structured information without task-specific training data.

Flexibility alone, however, does not produce a usable graph. Open extractors may represent the same entity under several surface forms, generate many overlapping relation labels, or fail to reconcile facts extracted from different passages. As these variations accumulate, the resulting graph becomes sparse and fragmented. Moreover, generated entities and relations may no longer retain a clear connection to the passages that support them. These problems weaken the consistency, interpretability, and auditability that motivate KG construction in the first place.

We present LMKG, a modular language-model-based pipeline for both open and ontology-constrained KG construction. The pipeline separates entity extraction, entity canonicalization, relation extraction, evidence resolution, and evidence refinement. An optional ontology constrains entity

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 334–342
October 28, 2026 ©2026 Association for Computational Linguistics

extraction, relation extraction, and refinement without changing the remaining construction process. Every predicted entity and triple is also paired with a model-proposed source quote, which deterministic matching resolves to a character span when possible. This gives the resulting graph an explicit connection to its textual evidence.

Constructing a graph from a long document introduces an additional integration problem: local chunks provide manageable extraction contexts, but important entities and relations may cross chunk boundaries. Therefore, we also introduce two document-level strategies.

We evaluate LMKG through three questions:

RQ1: Ontology control. How does supplying an ontology affect extraction accuracy, conformance, and hallucination?

RQ2: Long-document construction. How do sequential and global-first construction strategies differ in fact recovery and graph size?

RQ3: Graph utility. Is the extracted knowledge graph useful for a downstream task when compared to the original text it was extracted from?

2 Related Work

From extraction to graph construction. OpenIE extracts relational tuples without a predefined inventory (Angeli et al., 2015), while supervised document-level systems such as DyGIE++ jointly model entities, relations, and events under task-specific annotations (Wadden et al., 2019). Generative IE serializes structured outputs: REBEL predicts relation triples over a broad inventory (Huguet Cabot and Navigli, 2021), and GenIE constrains decoding to a predefined KB schema (Josifoski et al., 2022).

LLM-based construction and canonicalization. EDC organizes open construction by extracting, defining, and canonicalizing a schema, then mapping open predicates into it (Zhang and Soh, 2024). KGGen separates entity and relation extraction from entity and edge resolution, then evaluates whether its graph retains source information with MINE-1 and its downstream usefulness with MINE-2 (Mo et al., 2025). GraphRAG constructs an entity graph and community summaries for corpus-level retrieval and query-focused summarization (Edge et al., 2024). COMBO distinguishes entity, relation, and ontology canonicalization in open KGs (Jiang et al., 2023). Wikontic instead uses Wikidata-aligned type and relation constraints together with entity normalization (Chepurova et al., 2026), and schema-adaptable KGC studies generalization when extraction schemas change (Ye et al., 2023). We develop LMKG as an intersection of these settings: one representation and program support both open and supplied-schema extraction, while canonicalization, refinement and knowledge graph construction strategies remain interchangeable.

Grounding, refinement, and evaluation. Iterative construction can improve coverage but also amplify unsupported edges. SAC-KG uses a generator–verifier–pruner loop for domain KG construction (Chen et al., 2024); RefChecker extracts claim triplets and verifies them against a reference (Hu et al., 2024). We propose a narrower mechanism that asks for a local quote with each prediction and resolves it to source offsets deterministically before source-conditioned KG refinement. This provides an anchor in the actual text for extracted facts.

3 Methodology

3.1 Knowledge Graph and Ontology Representations

A knowledge graph is a pair $G = (\mathcal{E}, \mathcal{R})$. The set $\mathcal{E}$ holds the entities, each with a name and a type from an entity-type vocabulary $\mathcal{T}_{\text{ent}}$. The set $\mathcal{R} \subseteq \mathcal{E} \times \mathcal{E}$ holds the relations. A relation $re \in \mathcal{R}$ is paired with a type from a relation-type vocabulary $\mathcal{T}_{\text{rel}}$. Each relation is therefore expressed either as a triple of the form $re = (s, r, o)$ or in the functional form $re = r(s, o)$ with $s, o \in \mathcal{E}$, $r \in \mathcal{T}_{\text{rel}}$. An ontology is the triple $\mathcal{O} = (\mathcal{T}_{\text{ent}}, \mathcal{T}_{\text{rel}}, \mathcal{T}_{\text{sig}})$, where the set of allowed signatures is defined as $\mathcal{T}_{\text{sig}} \subseteq \mathcal{T}_{\text{rel}} \times \mathcal{T}_{\text{ent}} \times \mathcal{T}_{\text{ent}}$. Under a given ontology $\mathcal{O}$, entity recognition should type every mention with a type from $\mathcal{T}_{\text{ent}}$, and relation extraction may use only relation types from $\mathcal{T}_{\text{rel}}$ and signatures allowed by $\mathcal{T}_{\text{sig}}$.

3.2 LMKG

We developed our system as a streamlined pipeline that uses a mixture of components: reusable DSPy modules for LLM calls (Khattab et al., 2024), deterministic evidence resolution, and adaptable chunking strategies. In our system, every entity and every triple additionally carries provenance (q, c, p): an evidence quote q, the chunk c it came from, and a character span p in the source document. The

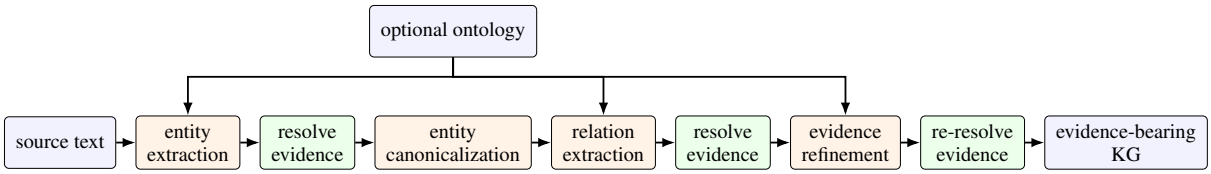


Figure 1: The LMKG extraction pipeline. Orange boxes contain language-model decisions; green boxes map proposed evidence quotes to source offsets deterministically. The optional ontology constrains the label vocabulary. Fold and enrich compose this pipeline differently for long documents (Figures 2 and 3).

model supplies q alongside the prediction it justifies. The deterministic resolver locates q in the source and records its character span p; when no match exists, p remains empty rather than receiving an LLM-guessed offset. Table 1 shows two triples with their quotes and spans. We define the pipeline as shown in Figure 1 as the following series of steps:

Named Entity Recognition (NER) recognizes named entities in the supplied text.

Evidence resolution is a deterministic mechanism that maps each entity to a given quote, document chunk and numeric span.

Canonicalization merges coreferent mentions in the text and expresses each cluster using a single canonical name. The graph is then deterministically re-derived to account for the entity clusters while remapping the evidence. Furthermore, if canonicalization occurs after relations are already present, it accounts for the changes to the entities.

Relation Extraction (RE) recognizes relations starting from entities extracted in the previous step and the supplied text.

Evidence resolution is the same mechanism as above also applied to extracted relations.

Evidence refinement is a step that takes as input a graph in which each entity or triple is associated with provenance and attempts to add missing triples and remove unsupported ones through critique. This step can be repeated multiple times.

Evidence resolution is the same mechanism as above applied to the whole graph.

3.3 Local Extraction with Evidence

The pipeline alternates model decisions with deterministic resolution (Figure 1). A language model identifies the entities/triples and returns a short verbatim quote for each; the resolver then maps those quotes to character offsets, trying exact, case-insensitive, and whitespace-tolerant matching.

Triple	Evidence quote	Span
`(Joshua Parlby, born_in, Longton)`	"born 1855 in Longton, Staffordshire"	30–65
`(Joshua Parlby, managed, Manchester City)`	"managed Manchester City in the 1890s"	103–139

Table 1: Two triples from a constructed HotpotQA graph. The quote is requested from the model with the proposition; the character span is assigned by deterministic matching against the source and is absent when no match is found.

An *evidence refiner* iteratively critiques entities and triples against the source. In each iteration it can remove unsupported items and add missing ones, stopping at a configured iteration limit.

3.4 Long Documents: Fold and Enrich

A prominent problem when extracting KGs from text using LLMs is posed by long documents. The long context may hinder the capabilities of an LLM to perform extraction. We propose two strategies to tackle this problem: *Fold* and *Enrich*.

Both long-document strategies split the text into chunks with recorded starting offsets and perform evidence-aware extraction within each chunk. The difference lies in how KGs are constructed and aggregated. *Fold* (Figure 2) accumulates graph triples and conditions each subsequent chunk extraction; *Enrich* (Figure 3) performs a whole-document extraction without refinement. Both strategies then perform document-level entity canonicalization followed by evidence resolution, aligning chunks with the original document spans.

Fold (sequential). The system starts with an empty graph and visits chunks in order. Triples from earlier chunks are supplied as anchors to subsequent extraction in later chunks. This is impor-

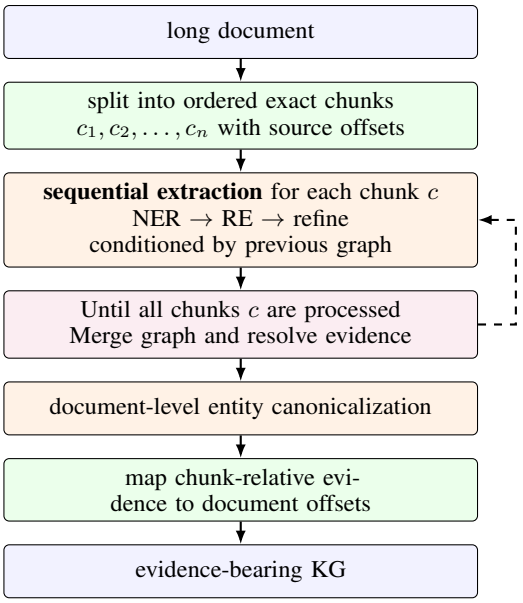


Figure 2: **Fold** strategy visits chunks sequentially. Later relation extraction can refer to entities accumulated from earlier chunks; canonicalization and final evidence resolution happen once after all chunks.

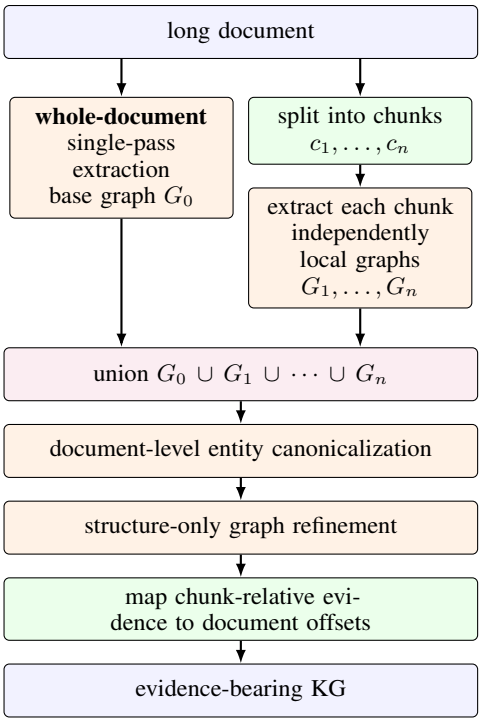


Figure 3: **Enrich** combines a whole-document base graph with independent chunk graphs before global canonicalization. It then adds a structure-only completion pass (i.e., without the document as context).

tant for RE because it allows a later proposition to point back to an earlier mention. Locally extracted entities/triples accumulate gradually and are followed by one final document-level canonicalization pass.

Enrich (global-first). The system first runs the complete pipeline over the entire document. It then extracts every chunk independently and merges local results with the base graph. After global canonicalization, a structure-only graph refiner proposes strongly implied missing edges among existing nodes. Enrich therefore uses more model calls and can add more coverage, but its structure-only completion is not source-grounded and must be evaluated for precision separately.

4 Experimental Setup

Every experiment uses `gpt-5.4-mini`[1] as the extraction and question-answering model.

We select three benchmarks with which to evaluate our system and answer our research questions:

- **Text2KGBench**: a text-to-knowledge-graph benchmark that also includes ontologies (RQ1).
- **MINE-1**: a recall-oriented benchmark that measures fact retrieval in long documents (RQ2).
- **HotpotQA**: a QA benchmark that supplies a document as context for the question (RQ3).

We reproduced all experiments as faithfully as possible, given the availability of the original data and upstream code.

4.1 Text2KGBench

We compare the same extraction pipeline firstly omitting the ontology and secondly providing such ontology on both Text2KGBench splits: Wikidata–TekGen ($n = 4062$, 19 ontologies), DBpedia–WebNLG ($n = 2014$, 10 ontologies). Metrics are the benchmark's macro-averaged triple precision, recall, and F1, ontology conformance, and subject, relation, and object hallucination rates. For definitions of these metrics, we refer the reader to the original paper (Mihindukulasooriya et al., 2023). Both conditions follow the benchmark's own prompting protocol: one in-context example per test sentence. We selected them using the benchmark's precomputed test–train similarity ranking. Because the input is sentence-level, we run the extractor without chunking, evidence

[1]Accessed via the OpenAI API (`GPT-5.4-mini`); see https://platform.openai.com/docs/models, July 2026.

refinement, or canonicalization.

4.2 MINE-1

MINE-1 contains 100 articles with 15 annotated facts each (Mo et al., 2025). We construct one open-schema graph per article with fold and enrich, using structure-aware chunks (i.e., taking into account paragraphs and blank lines). We use two evidence refinement rounds.

Scoring follows the benchmark's retrieve-then-judge procedure: each fact is embedded with a SentenceTransformers encoder (Reimers and Gurevych, 2019)[2]. The eight most similar nodes are retrieved, their two-hop neighborhoods are rendered as text, and a `gpt-5`[3] judge at high reasoning effort adjudicates entailment; this configuration reproduces the reference judge. The outcome is the percentage of source facts recoverable from the graph, averaged by article.

For external comparison, we include the KGGen, GraphRAG (Edge et al., 2024), and OpenIE (Angeli et al., 2015) results reported by Mo et al. (2025), as well as the Wikontic results reported by Chepurova et al. (2026).

4.3 HotpotQA

We use a subset ($n = 2500$) of the HotpotQA (Yang et al., 2018) distractor set in a closed-book per-item setting. We run two experiments: one in which the full passages are provided to establish a baseline and another in which a KG produced by LMKG is provided instead. For each question a graph is built from the passages supplied with that question, and the answerer receives graph facts instead of the passages. The graph is built on open schema by a single full-document extraction pass with two evidence refinement rounds and LLM canonicalization. The text configuration emits no sentence-level attribution, so its supporting-fact and joint scores are structurally zero and are not useful for comparison.

5 Results and discussion

5.1 Ontology conditioning: Text2KGBench (RQ1)

Ontology conditioning aligns the produced vocabulary with the target ontology schema. We can observe that this matters most in Text2KGBench, in which labels are constrained and not easily inferable (Table 2). LMKG without ontology conditioning performs poorly (F1 0.08–0.17) compared to the same system with ontology conditioning (F1 0.31–0.39). Part of this gap is definitional. Conformance and relation hallucination are defined against the supplied ontology, so an open-schema run that produces its own predicate labels is almost certain to score badly on both: the 0.11 and 0.37 conformance figures are largely attributed to open extraction not using the ontology's vocabulary. All four experiments issue about the same number of model calls per sentence (2.15–2.23), and cost per sentence differs by under 9% in either direction: TekGen ($0.0031 versus $0.0034); DBpedia ($0.0034 versus $0.0036).

5.2 Long documents and factual recall: MINE-1 (RQ2)

MINE-1 asks whether a graph contains enough information to recover known source facts. Our system consistently recovers facts from the text, achieving a high MINE-1 score with both strategies. Enrich recovers more source facts than fold: 90.80% against 84.87% (Table 3), the highest reported score on this benchmark. This gain comes with substantially larger graphs: enrich yields 80.1 edges per article on average against fold's 46.9, and 81.4 versus 67.2 entities. Because MINE-1 measures fact recovery and it does not penalize unsupported edges, this score should be read as coverage.

5.3 Question Answering: HotpotQA (RQ3)

The HotpotQA experiment shows that providing the full passages to the LLM produces significantly better results than providing LMKG-produced graphs as context (Table 4). Two explanations are possible: the graphs may omit information needed to answer, or the graph representation may be unsuitable for LLM consumption. Indirect evidence favours the latter as a contributing factor. On MINE-1, LMKG recovers 90.80% of source facts, indicating that its graphs preserve most salient content; and in the graph condition here the supporting-fact F1 is 71.67, so the facts needed to justify an answer may frequently be present in the graph even when the final answer is wrong. Moreover, we provided as input to the LLMs all the passages and the whole graph in each respective experiment. Accessing the whole knowledge graph rather than a

[2]We used `all-MiniLM-L6-v2`, the original embedding model used in KGGen.

[3]Accessed via the OpenAI API (`GPT-5`); see https://platform.openai.com/docs/models, July 2026

Dataset	System	Ont.	P ↑	R ↑	F1 ↑	Conf. ↑	Sub.H. ↓	Rel.H. ↓	Obj.H. ↓
Wikidata–TekGen	Vicuna-13B†	on	0.38	0.34	0.35	0.83	0.17	0.17	0.17
	Alpaca-LoRA-13B†	on	0.32	0.26	0.27	0.87	0.18	0.13	0.17
	LMKG	off	0.10	0.08	0.08	0.11	0.01	0.89	0.03
	LMKG	on	0.33	0.31	**0.31**	**0.97**	0.01	0.03	0.02
DBpedia–WebNLG	Vicuna-13B†	on	0.34	0.27	0.30	0.93	0.12	0.07	0.28
	Alpaca-LoRA-13B†	on	0.32	0.23	0.25	0.91	0.16	0.09	0.38
	LMKG	off	0.25	0.14	0.17	0.37	0.00	0.63	0.02
	LMKG	on	0.43	0.36	**0.39**	**1.00**	0.00	0.00	0.02

Table 2: Text2KGBench published baselines and full LMKG runs. Values are in the 0–1 scale and are macro averages across ontologies. Conf. is ontology conformance; Sub.H., Rel.H., and Obj.H. are subject, relation, and object hallucination rates. Bold marks the better of our two settings where the paired difference is significant. † Reported from Table 2 of Mihindukulasooriya et al. (2023)—*All* variant.

System	MINE-1 ↑
OpenIE†	29.29
GraphRAG†	47.94
KGGen†	65.41
WIKONTIC (gpt4o)†	84
WIKONTIC (gpt4.1-mini)†	86
Ours, fold	84.87
Ours, enrich	**90.80**

Table 3: MINE-1 fact recovery. Score is the mean per-article fact-recovery percentage. † Reported.

Context	Ans. EM ↑	Ans. F1 ↑	Sup. F1 ↑
Passages	**65.36**	**79.59**	n/a
Graph	55.40	68.48	71.67

Table 4: Comparison on 2,500 samples from the HotpotQA distractor subset.

selected subgraph (as happens in MINE-1) may partially explain this result. We do not test this directly and leave a controlled comparison to future work.

5.4 Evidence as audit interface

The provenance fields give the refiner a local proposition to inspect, allow deterministic code to recover document offsets, and allow downstream outputs to refer back to source units. This supports auditability by allowing a human reviewer to trace each entity or triple back to the original text.

6 Conclusion

We introduce LMKG, a modular and ontology-aware KG extraction pipeline that allows for flexible extraction by exposing modular components. We presented and evaluated three strategies for handling documents of different lengths: a simple strategy for sentences and two strategies for long documents. We tested our system in three different settings, two for evaluating KG quality and one for downstream usefulness. Our system reaches the highest reported score on the MINE-1 factuality recall benchmark, while demonstrating the importance of ontology awareness for LLMs whenever a target ontology is available. We observe a stark contrast between this strong fact recovery and the weaker downstream usability of the graphs produced by LMKG. This prompts us to question both the quality of the graphs and the LLM's ability to leverage them in the downstream task (QA), both of which will be addressed in future work.

Limitations

The comparison between *fold* and *enrich* on MINE-1 does not isolate the factors contributing to their performance. Because MINE-1 measures fact recovery (recall) and does not penalize unsupported edges, enrich's higher score reflects the additional coverage. A future ablation study would help isolate and quantify the sources of contribution and precision cost of each method.

Structure-only graph completion in enrich is especially vulnerable to plausible but unsupported edges. Canonicalization can merge distinct entities, and ontology constraints can discard valid facts that fall outside a schema. These errors matter in high-stakes domains; the resulting graphs should remain auditable and should not be treated as verified without human adjudication.

Although all selected benchmarks present the model with sentence-level or document-level extraction tasks, our system can also be used in a multi-document setting, which we plan to evaluate in future work.

LMKG operates on English text in the present study. Its evidence critic is a language-model self-check rather than an independent graph verifier. Although our evidence resolution mechanism partially mitigates this limitation, a stronger, language-independent external verification mechanism would be desirable.

All experiments use a single extraction and question-answering model. Our findings may therefore not transfer to other language models, and a systematic study of robustness across model families and sizes remains future work.

Acknowledgements

Thanks to DecisionBrain for providing computational resources to run the experiments.[4]

AI assistance. Generative AI tools were used to assist with language editing and improve the clarity of the manuscript. The authors reviewed and revised all generated suggestions and take full responsibility for the content of the paper.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735, Berlin, Heidelberg. Springer.

Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. 2024. SAC-KG: Exploiting large language models as skilled automatic constructors for domain knowledge graph. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4345–4360, Bangkok, Thailand. Association for Computational Linguistics.

Alla Chepurova, Aydar Bulatov, Mikhail Burtsev, and Yuri Kuratov. 2026. Wikontic: Constructing wikidata-aligned, ontology-aware knowledge graphs with large language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8304–8319, Rabat, Morocco. Association for Computational Linguistics.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Xiangkun Hu, Dongyu Ru, Lin Qiu, Qipeng Guo, Tianhang Zhang, Yang Xu, Yun Luo, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Knowledge-centric hallucination detection. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6953–6975, Miami, Florida, USA. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chengyue Jiang, Yong Jiang, Weiqi Wu, Yuting Zheng, Pengjun Xie, and Kewei Tu. 2023. COMBO: A complete benchmark for open KG canonicalization. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 340–357, Dubrovnik, Croatia. Association for Computational Linguistics.

Martin Josifoski, Nicola De Cao, Maxime Peyrard, Fabio Petroni, and Robert West. 2022. GenIE: Generative information extraction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4626–4643, Seattle, United States. Association for Computational Linguistics.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Nandana Mihindukulasooriya, Sanju Tiwari, Carlos F. Enguix, and Kusum Lata. 2023. Text2KGBench: A benchmark for ontology-driven knowledge graph generation from text. In *The Semantic Web – ISWC 2023*, volume 14266 of *Lecture Notes in Computer Science*, pages 247–265, Cham. Springer.

[4] https://decisionbrain.com

Belinda Mo, Kyssen Yu, Joshua Kazdan, Joan Cabezas, Proud Mpala, Lisa Yu, Chris Cundy, Charilaos Kanatsoulis, and Sanmi Koyejo. 2025. KGGen: Extracting knowledge graphs from plain text with language models. In *Advances in Neural Information Processing Systems*, volume 38, pages 33962–33985. Neural Information Processing Systems Foundation.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. Association for Computing Machinery.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Hongbin Ye, Honghao Gui, Xin Xu, Xi Chen, Huajun Chen, and Ningyu Zhang. 2023. Schema-adaptable knowledge graph construction. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6408–6431, Singapore. Association for Computational Linguistics.

Bowen Zhang and Harold Soh. 2024. Extract, define, canonicalize: An LLM-based framework for knowledge graph construction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9820–9836, Miami, Florida, USA. Association for Computational Linguistics.

A Model Prompts and Signatures

This appendix reports the instruction string and input–output signature of every model-facing component used by the system and the reported evaluations. DSPy (Khattab et al., 2024) serializes runtime values and structured types around these instructions. Components marked *CoT* use `ChainOfThought`; those marked *direct* use `Predict`. Text, ontology values, graph contents, and test-specific demonstrations are runtime inputs rather than fixed prompt text.

A.1 Modules

Entity extraction (CoT).

> *Extract the named entities mentioned in the text. If a list of allowed types is given, assign each entity exactly one of those types; if 'open', assign a suitable concise type. For each entity, set 'evidence' to a short verbatim quote copied from the text that mentions it (its supporting span).*

Inputs. `text` (*string*); `entity_types` (*string: comma-separated allowed types, or 'open'*).
Output. `entities` (*list of entities, each carrying a verbatim evidence quote*).

Relation extraction (CoT).

> *Extract every explicit subject-relation-object proposition in the text. Use the given entities as anchors, but preserve the actual entity or value asserted as the object. Each triple must express one fact. Its relation must name the semantic role of its object, never a copula; split descriptions or values that express multiple roles. Do not add outside facts. If allowed relations are given, use only those labels and skip propositions that do not fit; if 'open', use concise relation labels. Give every triple a short verbatim evidence quote supporting the whole proposition.*

Inputs. `text` (*string*); `entities` (*list of entity anchors containing names and types only*); `relation_types` (*string: allowed relations, or 'open'*).
Output. `triples` (*list of triples, each carrying a verbatim evidence quote*).

Entity canonicalization (CoT).

> *Group only entity mentions that unambiguously refer to the same entity. Keep uncertain mentions separate and choose an existing name for each group.*

Input. `entities` (*list of entities containing names and types only*).
Output. `clusters` (*list of entity clusters*).

Evidence refinement (CoT).

> *Verify each triple against the text, which is the source of truth. Remove unsupported triples and replace copula predicates or multi-role values with atomic propositions whose relations name*

their objects' semantic roles. If allowed relations are supplied, retain and add only those exact semantic labels; skip a proposition that does not fit rather than inventing a related label. Add explicit propositions that are missing; objects may be stated values rather than named entities. Give every added triple a verbatim evidence quote supporting the whole proposition.

Inputs. `text` (*string*); `triples` (*list of triples carrying evidence spans*); `relation_types` (*string: allowed relation labels, or 'open'*).
Outputs. `unsupported` (*triples to remove*); `missing` (*triples to add, each with an evidence quote*).

Structure-only graph completion (CoT).

You are given the entities of a knowledge graph and the relations already extracted between them. Propose additional relation triples that are missing but strongly implied by the existing structure—connecting entities that clearly should be linked given the relations already present. Use only the given entities as subjects and objects, and reuse existing relation labels where they fit. Do NOT restate triples already present. Return only the new triples.

Inputs. `entities` (*graph nodes containing names and types only*); `triples` (*existing relations without evidence*).
Output. `missing` (*new triples connecting existing nodes*).

A.2 Evaluation

HotpotQA answer generation (CoT).

Answer the question using ONLY the numbered facts provided.

The facts come from a knowledge graph extracted from the source document; they are all you have. If they do not contain the answer, say so rather than guessing from your own knowledge—an invented answer is worse than an admitted gap.

Answers may be an entity/span or exactly "yes" or "no". Return only that short answer. Never infer facts that are absent from the provided graph facts.

Inputs. `question` (*string*); `facts` (*numbered* `subject | relation | object`).
Outputs. `used_facts` (*list of supporting fact numbers, or an empty list*); `answer` (*short name, phrase, number, or date, without explanation*).

Zero-Shot Relation Classification with Large Language Models: Binary Prompting and Relevance Elicitation*

Sheila Alemany, Danielle Sullivan-Pao, Olga Simek
MIT Lincoln Laboratory
Lexington, MA, USA

Abstract

Relation classification (RC) transforms unstructured text into structured knowledge by identifying semantic relationships between entities. As natural language data volumes rapidly increase, manual knowledge extraction has become impractical. Existing zero-shot LLM-based approaches face limitations due to sparse RC task representation in instruction-tuning datasets and inherent biases toward certain relation types. To address these challenges, we propose Binary Relevance Elicitation (BRE), a true/false prompting framework that systematically queries an LLM for each candidate relation between two entities from a closed set of relation types. BRE incorporates relevance scoring to identify the most likely relation among those classified as "true," reducing false positive predictions. We evaluate BRE using multiple open-source LLMs and achieve state-of-the-art zero-shot RC accuracy across the TACRED family of datasets. Our results demonstrate performance improvements with newer model releases that are independent of parameter count, suggesting information representation advances within LLMs drive zero-shot RC capabilities.

1 Introduction

Social network analysis reveals how entities (people, organizations, locations) connect and how these connections influence social behavior and outcomes. Relation classification (RC) identifies semantic relationships between entities in sentences, converting unstructured text into structured data. These triplets enhance knowledge classification, improve information retrieval and question answering, and generate knowledge graphs (Zhao et al., 2024a; Nayak et al., 2021).

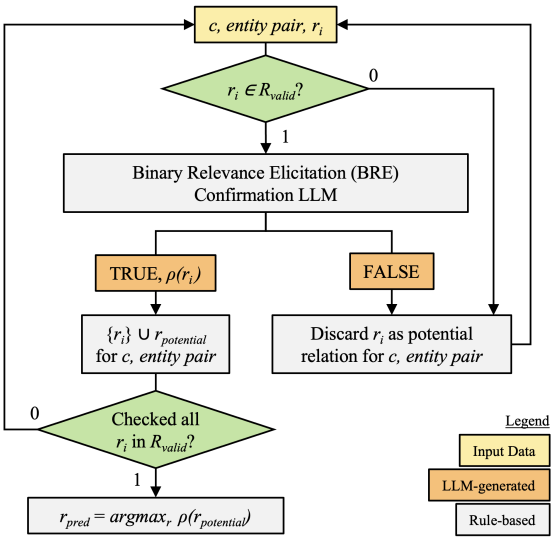


Figure 1: Zero-shot BRE prompting pipeline using the relevance elicitation measure (ρ) to select the relation for a given context (c) and entity pair. The predicted relation is the one with the highest relevance measure from the set of possible relations ($r_{pred} \in r_{potential} \subseteq R$).

RC has achieved state-of-the-art performance by leveraging pre-trained language models (PLMs) to extract relation triplets (Park and Kim, 2021; Han et al., 2022; Zhou and Chen, 2022; Huguet Cabot et al., 2021). However, a fundamental misalignment persists: top-performing PLMs rely on pre-training objectives like masked language modeling that fail to capture the structured semantic relationships between predefined entity pairs (Devlin, 2018; Liu, 2019). While some efforts have reframed RC as textual entailment or natural lan-

*DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Combatant Commands under Air Force Contract No. FA8702-15-D-0001 or FA8702-25-D-B002. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Combatant Commands.

© 2026 Massachusetts Institute of Technology. Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 343–358
October 28, 2026 ©2026 Association for Computational Linguistics

guage inference (Obamuyide and Vlachos, 2018; Chanthran et al., 2026), PLM-based approaches remain critically dependent on large volumes of high-quality labeled training data (Huguet Cabot et al., 2021; Elsahar et al., 2018; Zhao et al., 2024a). Language ambiguity makes annotating semantic relations inherently challenging, producing incomplete or inconsistent ground-truth data that varies by annotator perspective and dataset domain. The absence of standardized, closed-set training and testing datasets represents a fundamental obstacle to RC advancement, making zero-shot approaches not merely desirable but essential for domain-specific applications.

Large language models (LLMs) have shown promise with minimal training data (Hang et al., 2025), yet they falter in critical scenarios: capturing implied knowledge in long-tail cases where entity relations must be globally inferred (Zhao et al., 2024b), and handling complex multi-relation sentences without handcrafted templates for each relation type (Li et al., 2024; Geroimenko, 2025; Aguda et al., 2025). We introduce a zero-shot approach to RC centered on *relevance elicitation*, a quantitative ranking mechanism that identifies optimal relation types and directly addresses LLMs' systematic over-prediction bias. While our method uses binary true/false queries as a lightweight foundation which avoids manual template customization, the performance gains stem primarily from aggregating these queries into relevance scores rather than treating them as final decisions. This shift toward ranked, relevance-based selection is the key driver of our results, mirroring the binary relevance filtering approach of Zhuang et al. (2024) for document ranking. Our method achieves state-of-the-art zero-shot RC performance across the TACRED family of datasets (Zhang et al., 2017) using various open-source LLMs.

2 Related Work

2.1 From Supervised to Zero-Shot Closed RC

RC methods vary in supervision requirements and output structures. Traditional supervised RC approaches using deep neural networks require substantial annotated training data, limiting their applicability to new domains or relation types. Unsupervised relation extraction methods eliminate this annotation burden, primarily employing clustering-based techniques that extract entities via named-entity taggers and group entity pairs based on contextual similarity (Detroja et al., 2023). However, these methods produce open-ended clusters rather than assignments to predefined relation labels, making interpretation ambiguous and challenging to evaluate when sentences contain entity pairs with multiple relations. Zero-shot closed RC addresses these limitations by enabling users to specify a predefined set of relations of interest without requiring annotated training examples. This approach combines the flexibility of unsupervised methods (no annotation requirement) with the interpretability of supervised methods (explicit, user-defined label assignments), making it particularly suitable for domain-specific applications where relation types are known but labeled data is scarce.

2.2 Pre-Trained Language Models (PLMs)

PLMs have advanced RC by eliciting rich knowledge from large pre-trained models (Huguet Cabot et al., 2021; Tang et al., 2022; Sui et al., 2023). Models like BERT (Devlin, 2018), RoBERTa (Liu, 2019), and ELMo (Peters et al., 2018) produce contextualized representations where token meaning depends on surrounding context, enabling nuanced understanding of entity relationships within sentences. PLM-based RC approaches employ fine-tuning to adapt these models for specific RC tasks. However, standard pre-training objectives (e.g., masked language modeling, next-token prediction) do not naturally capture structured semantic relationships, creating reliance on task-specific training data (Hang et al., 2025). This creates two key limitations that zero-shot closed RC addresses: significant computational requirements for training and inference, and tendency to overfit on domain-specific datasets (Detroja et al., 2023).

2.3 Large Language Models (LLMs)

LLMs have transformed zero-shot RC through two primary approaches: direct RC and synthetic data generation for training downstream models. For direct classification, prompt engineering approaches typically require no additional fine-tuning, simplifying deployment and eliminating the need for annotated few-shot data (Chia et al., 2022; Li et al., 2024). Alternatively, LLMs can generate synthetic training data as supplementary samples for unseen relations (Xu et al., 2023). However, effective prompt engineering for direct classification requires considerable subject matter expertise specific to the RC-employed domain (Geroimenko, 2025), and out-of-the-box LLM performance is

limited by general language understanding capabilities and training biases (Aguda et al., 2025; Dai et al., 2024).

Recent efforts show LLM performance varies significantly with prompt templates (Liu et al., 2023; Zhang et al., 2023a; Xu et al., 2024). To address this, approaches have incorporated additional context (Wan et al., 2023), entity type information (Zhang et al., 2023a), multi-step prompting for chain-of-thought control (Layegh et al., 2024; Hannah et al., 2025), and logical searches for discrete prompts to avoid manual design (Xu et al., 2024). A common observation is that task simplification correlates with improved LLM performance (Zhou et al.; Khot et al.). Our approach uniquely simplifies RC to a binary decision and adapts the LLM-as-a-judge framework (Gu et al., 2024) to eliminate domain-specific prompt-tuning while improving performance.

3 Methodology

BRE, described in Algorithm 1, simplifies RC to a series of binary decisions while enabling principled relation selection through relevance scoring. For each entity pair (e_h, e_t) in context c, BRE iterates through type-compatible relations from a predefined set R, prompting the LLM to evaluate whether each relation r_i holds and rate its relevance ρ to the context on a 0-10 scale, given the relation definition d_i and the statement "e_h and e_t are somehow connected through the relationship of 'r_i'."

Pairing binary classification with a relevance score is critical: the binary decision identifies plausible relations, while ρ enables ranking among multiple true relations. BRE returns the highest-scoring relation among all 'true' classifications[1], or 'no relation' if none qualify. This mechanism directly addresses the multi-relation challenge in RC, providing a principled, interpretable basis for relation selection without domain-specific prompt engineering or few-shot examples. See Appendix A.1 for the prompt template.

3.1 Closed RC Definition

The RC task considers a given sentence c, a set of entities $E \subseteq c$, entity pairs $(e_h, e_t) \in E \times E$ where $e_h \neq e_t$, and a set of relation types R. The task identifies the semantic relations between entity pairs in c. The RC output is a set of sentence-level triplets $\langle e_h, r_{pred}, e_t \rangle$ where $r_{pred} \in R \cup \{\text{'no relation'}\}$. Zero-shot RC employs no fine-tuning on the target relations, and no examples are included in the prompt. The performance varies substantially across LLMs even under identical task framing. By holding the dataset and prompt fixed, this work isolates and evaluates these cross-model differences. Figure 1 shows the flow to identify the predicted relation for a valid entity pair.

Algorithm 1 Binary Relevance Elicitation (BRE)

Require: sentence c, entity pair (e_h, e_t), entity types (e_h^{type}, e_t^{type}), relation set R, valid type pairs T_r, definitions D, LLM θ
  $R_{valid} \leftarrow \{r \in R \mid (e_h^{type}, e_t^{type}) \in \mathcal{T}_r\}$
  $r_{pot} \leftarrow \emptyset$ ▷ Store LLM-selected relations
  for $r_i \in R_{valid}$ **do**
    $prompt \leftarrow$ construct_prompt$(c, e_h, r_i, e_t, D(r_i))$
    $(y_i, \rho_i) \leftarrow \theta(prompt)$
      ▷ $y_i \in \{\text{TRUE}, \text{FALSE}\}, \rho_i \in [0, 10]$
    if $y_i =$ TRUE **then**
      $r_{pot} \leftarrow r_{pot} \cup \{(r_i, \rho_i)\}$
    end if
  end for
  return $|r_{pot}| = 0$? 'no relation' : $argmax_{(r,\rho) \in r_{pot}} \rho$

3.2 Entity-Type Aware Querying

Entity type constraints critically reduce the search space of potential relation triplets. By defining valid entity type pairs (e_h^{type}, e_t^{type}) for each relation, we systematically filter candidate relations before LLM inference. Specifically,

$$R_{valid} = \{r \in R \mid (e_h^{type}, e_t^{type}) \in T_r\}$$

where T_r is the set of valid type pairs for relation r as defined by TACRED dataset annotation guidelines (Ellis et al., 2015). This leverages domain knowledge to eliminate semantically invalid combinations (e.g., testing "founded by" for two location entities) (Zhang et al., 2023a). This optional type-aware filtering improves computational efficiency and extraction precision by focusing on semantically plausible candidates, yielding more reliable and cost-effective RC systems.

3.3 Relation Definition Context

Including relation definitions in prompts serves as a disambiguation mechanism (Zhou et al., 2024). Natural language descriptions establish clear semantic boundaries for interpreting relation types, which is useful for ambiguous or overlapping relations (e.g., distinguishing "employed by" from "affiliated with"). Grounding reasoning in explicit

[1] If two relations are assigned the same relevance score, the selected relation is randomly chosen for this evaluation. Additional details are provided in Appendix A.5

Model	Huggingface Model ID	Parameter Size (B)	Release Date
Llama-3.2-3B	meta-llama/Llama-3.2-3B-Instruct	3	09/25/2024
Llama-3.1-8B	meta-llama/Llama-3.1-8B-Instruct	8	07/23/2024
Llama-3.1-70B	meta-llama/Llama-3.1-70B-Instruct	70	07/23/2024
Llama-3.3-70B	meta-llama/Llama-3.3-70B-Instruct	70	12/06/2024
Llama-4-Scout-17B-16E	meta-llama/Llama-4-Scout-17B-16E-Instruct	109	04/05/2025
Mistral-7B	mistralai/Mistral-7B-Instruct-v0.2	7	12/25/2023
Mixtral-8x7B	mistralai/Mixtral-8x7B-Instruct-v0.1	56	12/10/2023
Mixtral-8x22B	mistralai/Mixtral-8x22B-Instruct-v0.1	176	04/17/2024
Qwen2.5-VL-32B	Qwen/Qwen2.5-VL-32B-Instruct	32	02/19/2025
DeepSeek-R1-Distill-Qwen-32B	deepseek-ai/DeepSeek-R1-Distill-Qwen-32B	32	02/23/2025

Table 1: Evaluated LLM version details including their corresponding Huggingface model IDs, their parameter sizes, and release dates in the format MM/DD/YYYY

definitions reduces semantic confusion and ensures consistent interpretation across contexts, rather than relying on implicit, context-dependent associations. Specifically, we define $D = \{r_i : d_i\}$ such that d_i is the definition for r_i as described in the TACRED family dataset annotation guidelines (Ellis et al., 2015; Stoica et al., 2021).

4 Evaluation

4.1 Experimental Setup

LLM Models We evaluate open-source LLMs from multiple families (Llama, Mistral, Qwen, and DeepSeek) detailed in Table 1, spanning varying parameter counts to examine correlations between model size, organization, release date, and RC performance. The same prompt template was used across all models.

Computational Usage We implemented our approach using VLLM on NVIDIA H100 GPUs, scaling resources by model size (up to four H100s for the largest model). We performed five inference runs per model and report averaged results to account for stochastic variation.

NoTA Evaluation LLM-based RC systems exhibit a critical bias: they over-assign named relations rather than predicting "no relation" or the none-of-the-above (NoTA) class, even when no valid relationship exists (Wan et al., 2023; Ma et al.; Zhang et al., 2023b). This bias is amplified by the asymmetric nature of many relations, requiring evaluation of numerous ordered entity pairs with potentially no predefined relationship (Qin et al., 2021; Baek and Choi, 2022), resulting in severe class imbalance that increases false positives and reduces precision. Our relevance elicitation method mitigates this by ranking potential false positives through comparative relevance scoring across candidate relations. Since Gutierrez et al. (Gutierrez et al., 2022) attributed early LLM underperformance on RC largely to poor NoTA handling, we report both NoTA-included and NoTA-excluded metrics for comparison with prior work.

Parsing LLM Outputs All evaluated LLMs followed a specified format starting with "true" or false" followed by the numerical relevance score. Llama-4-Scout-17B-16E consistently placed this output at the end of its response and required a modified parsing script. Complete parsing scripts are provided in Appendix A.2. Additional analysis regarding invalid outputs and no-score outputs are included in Appendix A.5.

4.2 Baselines

Few-Shot Baselines ODDA (Zhong et al., 2025) employs data augmentation to iteratively generate few-shot examples based on the RE model's predictions for target relations, then produces diverse training data using attribute-based constraints. ConsistRE (Zheng et al., 2024) uses relation-specific keywords as generation constraints to encourage relation consistency. However, it requires access to an external corpus for keyword extraction, an assumption that may not be feasible in all scenarios. Lastly, both approaches use DeepSeek-v3 (671B parameters) along with its PLM backbone.

Zero-Shot Baselines QA4RE (Zhang et al., 2023a) reformulates RC as multiple-choice question-answering by incorporating valid entity pairs into pre-defined relation templates as answer options. SUMASK (Li et al., 2023) takes a different approach, combining LLMs with natural language inference and chain-of-thought reasoning to sequentially query whether each candidate relation exists between entity pairs through iterative decomposition. REA (Layegh et al., 2024)

Dataset	Model	macro-average			weighted-average			micro-F_1
		Prec.	Rec.	F_1	Prec.	Rec.	F_1	Accuracy
Re-TACRED	Llama-3.2-3B	$17.5_{\pm 0.29}$	$41.4_{\pm 0.91}$	$20.9_{\pm 0.4}$	$58.0_{\pm 0.37}$	$46.8_{\pm 0.74}$	$49.1_{\pm 0.59}$	$42.3_{\pm 0.56}$
	Llama-3.1-8B	$21.8_{\pm 0.3}$	$53.5_{\pm 1.02}$	$27.7_{\pm 0.24}$	$64.0_{\pm 0.22}$	$51.0_{\pm 0.88}$	$50.1_{\pm 1.0}$	$43.6_{\pm 1.02}$
	Llama-3.1-70B	$61.4_{\pm 0.49}$	$53.6_{\pm 1.07}$	$54.4_{\pm 0.76}$	$80.7_{\pm 0.2}$	$76.0_{\pm 0.17}$	$75.8_{\pm 0.11}$	$79.1_{\pm 0.25}$
	Llama-3.3-70B	$57.9_{\pm 1.92}$	$62.9_{\pm 3.21}$	$57.8_{\pm 1.53}$	$73.2_{\pm 8.32}$	$72.4_{\pm 2.93}$	$71.4_{\pm 4.47}$	$79.5_{\pm 0.4}$
	Llama-4-Scout-17B-16E	$57.8_{\pm 0.37}$	$73.3_{\pm 0.62}$	$62.1_{\pm 0.47}$	$86.9_{\pm 0.33}$	$83.9_{\pm 0.33}$	$84.6_{\pm 0.29}$	$83.5_{\pm 0.29}$
	Mistral-7B	$39.1_{\pm 0.89}$	$55.8_{\pm 2.84}$	$40.6_{\pm 2.31}$	$67.1_{\pm 4.91}$	$63.7_{\pm 3.35}$	$63.4_{\pm 3.44}$	$66.1_{\pm 1.55}$
	Mixtral-8x7B	$48.2_{\pm 0.96}$	$58.0_{\pm 1.02}$	$47.2_{\pm 0.47}$	$75.4_{\pm 0.41}$	$72.7_{\pm 0.57}$	$72.4_{\pm 0.39}$	$72.1_{\pm 0.61}$
	Mixtral-8x22B	$50.7_{\pm 1.28}$	$61.3_{\pm 1.16}$	$52.6_{\pm 1.25}$	$81.7_{\pm 0.35}$	$77.7_{\pm 0.34}$	$78.2_{\pm 0.33}$	$79.5_{\pm 0.5}$
	Qwen2.5-VL-32B	$66.2_{\pm 0.16}$	$64.4_{\pm 0.27}$	$61.9_{\pm 0.22}$	$86.5_{\pm 0.01}$	$82.5_{\pm 0.03}$	$83.2_{\pm 0.03}$	$85.4_{\pm 0.03}$
	DeepSeek-R1-Distill-Qwen-32B	$44.4_{\pm 0.29}$	$60.7_{\pm 2.36}$	$48.4_{\pm 0.78}$	$76.2_{\pm 0.38}$	$74.2_{\pm 0.36}$	$74.2_{\pm 0.28}$	$74.4_{\pm 0.45}$
TACREV	Llama-3.2-3B	$21.2_{\pm 0.24}$	$47.6_{\pm 0.82}$	$24.2_{\pm 0.39}$	$75.7_{\pm 0.23}$	$54.5_{\pm 0.41}$	$61.3_{\pm 0.19}$	$49.6_{\pm 0.36}$
	Llama-3.1-8B	$22.0_{\pm 0.52}$	$65.0_{\pm 1.64}$	$28.8_{\pm 0.67}$	$79.2_{\pm 0.32}$	$52.1_{\pm 1.05}$	$58.9_{\pm 1.02}$	$47.6_{\pm 0.99}$
	Llama-3.1-70B	$56.8_{\pm 0.55}$	$60.1_{\pm 2.03}$	$54.6_{\pm 1.16}$	$85.7_{\pm 0.33}$	$87.5_{\pm 0.06}$	$86.1_{\pm 0.18}$	$85.8_{\pm 0.13}$
	Llama-3.3-70B	$55.7_{\pm 0.2}$	$70.5_{\pm 0.01}$	$59.6_{\pm 0.31}$	$87.9_{\pm 0.37}$	$88.2_{\pm 0.26}$	$87.8_{\pm 0.3}$	$86.4_{\pm 0.22}$
	Llama-4-Scout-17B-16E	$52.2_{\pm 0.52}$	$82.2_{\pm 1.21}$	$60.6_{\pm 0.53}$	$90.9_{\pm 0.11}$	$88.3_{\pm 0.14}$	$88.9_{\pm 0.11}$	$86.7_{\pm 0.1}$
	Mistral-7B	$37.5_{\pm 0.13}$	$69.1_{\pm 0.73}$	$44.5_{\pm 0.15}$	$84.6_{\pm 0.13}$	$74.3_{\pm 0.21}$	$77.8_{\pm 0.15}$	$73.3_{\pm 0.22}$
	Mixtral-8x7B	$46.1_{\pm 0.79}$	$70.6_{\pm 1.4}$	$52.5_{\pm 0.48}$	$86.2_{\pm 0.24}$	$80.2_{\pm 0.4}$	$82.3_{\pm 0.25}$	$78.6_{\pm 0.39}$
	Mixtral-8x22B	$46.8_{\pm 0.43}$	$72.1_{\pm 0.79}$	$53.9_{\pm 0.44}$	$88.0_{\pm 0.11}$	$87.1_{\pm 0.14}$	$87.2_{\pm 0.11}$	$85.3_{\pm 0.11}$
	Qwen2.5-VL-32B	$62.6_{\pm 0.6}$	$76.7_{\pm 0.25}$	$65.0_{\pm 0.1}$	$91.8_{\pm 0.19}$	$90.6_{\pm 0.02}$	$90.5_{\pm 0.01}$	$90.0_{\pm 0.01}$
	DeepSeek-R1-Distill-Qwen-32B	$42.3_{\pm 0.6}$	$70.9_{\pm 0.6}$	$49.2_{\pm 0.47}$	$85.6_{\pm 0.17}$	$83.5_{\pm 0.45}$	$84.0_{\pm 0.23}$	$80.7_{\pm 0.47}$
TACRED	Llama-3.2-3B	$22.9_{\pm 0.39}$	$46.0_{\pm 1.06}$	$25.4_{\pm 0.34}$	$75.2_{\pm 0.24}$	$54.2_{\pm 0.36}$	$61.1_{\pm 0.28}$	$49.8_{\pm 0.34}$
	Llama-3.1-8B	$23.6_{\pm 0.26}$	$59.8_{\pm 0.96}$	$30.2_{\pm 0.4}$	$78.1_{\pm 0.05}$	$51.7_{\pm 0.98}$	$58.6_{\pm 0.83}$	$48.0_{\pm 0.87}$
	Llama-3.1-70B	$54.7_{\pm 1.36}$	$50.7_{\pm 1.34}$	$50.5_{\pm 1.11}$	$84.0_{\pm 0.19}$	$84.8_{\pm 0.19}$	$83.9_{\pm 0.18}$	$83.4_{\pm 0.3}$
	Llama-3.3-70B	$52.3_{\pm 0.38}$	$62.1_{\pm 1.55}$	$55.1_{\pm 1.0}$	$86.8_{\pm 0.29}$	$84.9_{\pm 0.04}$	$85.6_{\pm 0.15}$	$84.1_{\pm 0.07}$
	Llama-4-Scout-17B-16E	$52.0_{\pm 0.31}$	$70.0_{\pm 0.96}$	$57.6_{\pm 0.38}$	$88.5_{\pm 0.11}$	$85.3_{\pm 0.15}$	$86.5_{\pm 0.13}$	$84.6_{\pm 0.14}$
	Mistral-7B	$39.0_{\pm 0.82}$	$60.1_{\pm 0.48}$	$43.2_{\pm 0.38}$	$83.0_{\pm 0.07}$	$72.9_{\pm 0.32}$	$76.5_{\pm 0.23}$	$72.3_{\pm 0.32}$
	Mixtral-8x7B	$47.7_{\pm 0.39}$	$60.1_{\pm 0.63}$	$49.8_{\pm 0.42}$	$84.4_{\pm 0.09}$	$78.7_{\pm 0.31}$	$80.9_{\pm 0.2}$	$77.4_{\pm 0.25}$
	Mixtral-8x22B	$48.0_{\pm 0.2}$	$63.6_{\pm 0.18}$	$52.6_{\pm 0.16}$	$86.1_{\pm 0.44}$	$84.2_{\pm 0.15}$	$84.9_{\pm 0.26}$	$83.1_{\pm 0.16}$
	Qwen2.5-VL-32B	$60.6_{\pm 0.53}$	$63.4_{\pm 0.2}$	$58.4_{\pm 0.12}$	$89.0_{\pm 0.21}$	$86.8_{\pm 0.04}$	$87.3_{\pm 0.04}$	$86.9_{\pm 0.04}$
	DeepSeek-R1-Distill-Qwen-32B	$43.3_{\pm 0.84}$	$61.0_{\pm 1.02}$	$47.7_{\pm 0.71}$	$83.8_{\pm 0.26}$	$81.4_{\pm 0.32}$	$82.2_{\pm 0.21}$	$79.2_{\pm 0.37}$

Table 2: NoTA-included zero-shot RC performance on the TACRED family of datasets. The gray scores indicate highest performing score of the category. Performance and standard deviations are represented as percentages (%).

employs a more elaborate four-stage prompting strategy: identifying entity types, filtering relation candidates by entity type compatibility, generating hypothesis sentences with confidence scores for each candidate, and selecting the final relation based on accumulated evidence.

4.3 Datasets

TACRED (Zhang et al., 2017) is a widely-used dataset for sentence-level relation extraction with 42 relation types, containing annotated news articles and web documents with entities and predefined relations between persons and organizations for knowledge base construction. Updated versions include TACREV (Alt et al., 2020), which corrects systematic labeling errors (particularly false "no relation" labels), and Re-TACRED (Stoica et al., 2021), which employed improved crowd-sourced annotation and relabeled 23.9% of the original dataset for enhanced accuracy. Evaluating all three datasets (entire test sets) demonstrates our methodology's robustness to annotation inconsistencies.

5 Results

5.1 Relation Classification Performance

Qwen2.5-VL-32B consistently shows high RC performance likely due to its advanced attention mechanisms and multimodal pre-training which enhance structured relationship understanding (Wang et al., 2024). MoE architectures (e.g., Llama-4-Scout and Mixtral) with specialized tuning showed high accuracy with balanced classification performance (Shen et al., 2023). Both represent architectural innovations beyond standard decoder-only transformers, specifically beneficial for tasks requiring precise semantic relationship extraction.

Observation: DeepSeek-R1-Distill-Qwen-32B showed inferior expected performance suggesting distillation disproportionately affected semantic relationship identification.

5.2 Baseline Comparisons

BRE prompting demonstrates substantial improvements over existing zero-shot baselines across all

	Re-TACRED	TACREV	TACRED
Qwen2.5-VL-32B	74.7	82.4	73.9
Llama-4-Scout-17B-16E	78.8	86.3	78.8
Llama-3.3-70B	65.6	78.9	73.9
Mixtral-8x22B	69.0	79.4	70.7
Llama-3.1-70B	62.1	68.3	62.7
DeepSeek-R1-Distill-Qwen-32B	66.7	75.7	68.5
Mixtral-8x7B	65.9	71.3	68.0
REA + Mixtral-8x7B	-	-	58.0
REA + GPT-3.5-Turbo	-	-	63.5
QA4RE + FLAN-T5 XXL	67.3	54.3	54.5
QA4RE + GPT-3.5-Turbo	61.2	59.4	59.4
SUMASK + GPT-3.5-Turbo	-	-	57.7

Table 3: NoTA-excluded comparison to state-of-the-art zero-shot RC methods. The gray scores indicate the top two micro-F_1 scores and the underlined score is the highest.

	Re-TACRED	TACREV	TACRED
Qwen2.5-VL-32B	85.6	90.0	86.8
Llama-4-Scout-17B-16E	83.2	86.7	84.4
Llama-3.3-70B	79.5	86.4	83.9
Mixtral-8x22B	79.4	85.9	83.0
Llama-3.1-70B	79.1	85.8	83.4
DeepSeek-R1-Distill-Qwen-32B	74.5	80.8	79.3
Mixtral-8x7B	72.4	78.7	77.7
SUMASK + GPT-3.5-Turbo	73.8	75.1	79.6
ODDA/TYP Marker (8-shot)	76.8	-	75.8
ODDA/KnowPrompt (8-shot)	84.1	-	81.0
ConsistRE/TYP Marker (8-shot)	71.8	-	72.6
ConsistRE/KnowPrompt (8-shot)	75.8	-	80.1

Table 4: NoTA-included comparison to state-of-the-art zero- and few-shot RC methods. All approaches in this table are zero-shot unless otherwise specified. The gray scores indicate the top two micro-F_1 scores and the underlined score is the highest.

evaluated datasets[2]. Most notably, BRE outperforms REA (Layegh et al., 2024) even when both methods employ the same foundational Mixtral 8x7B model, achieving 17% relative increased accuracy on TACRED. The consistent gains across datasets highlights that our approach effectively addresses the over-prediction bias inherent in LLM-based RC.

Using Qwen2.5-VL-32B and Llama-4-Scout, BRE matches or surpasses the performance of state-of-the-art few-shot baselines which require labeled

[2]Reported baseline performance numbers are from original efforts. Missing reported values were not included in the original work.

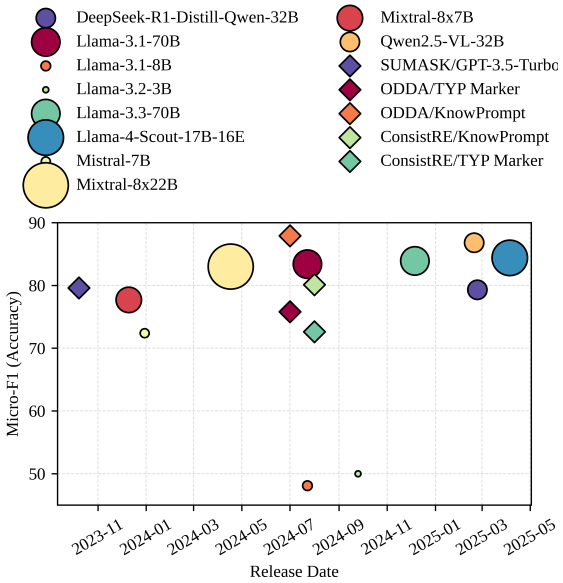


Figure 2: NoTA-included micro-F_1 comparison to state-of-the-art zero-shot RC methods on TACRED dataset

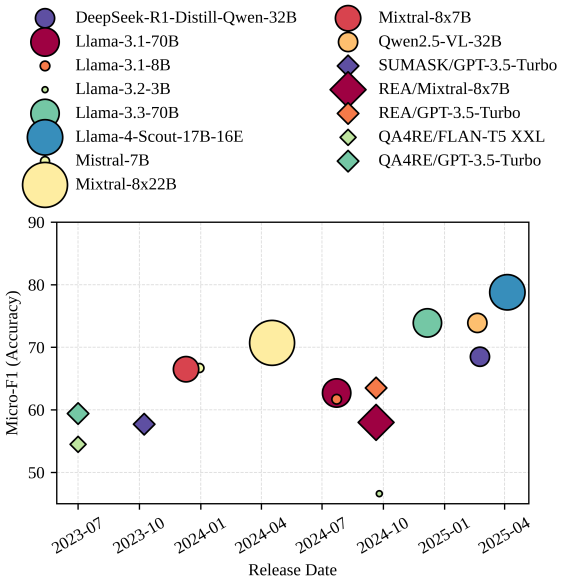


Figure 3: NoTA-excluded micro-F_1 comparison to state-of-the-art zero-shot RC methods on TACRED dataset

data or iterative data augmentation. However, analysis of NoTA-excluded metrics reveals an important distinction between our top-performing models: Llama-4-Scout outperforms Qwen2.5-VL-32B in NoTA-excluded scenarios.

Observation: Qwen2.5-VL-32B's high overall scores stem largely from strong NoTA classification, while Llama-4-Scout maintains more balanced performance.

In Figures 2 and 3, circles (○) represent BRE, and diamonds (◇) represent the baseline methods. The

size of the shape indicates the parameter size in the foundational model.[3] The baseline release dates included are those of the research publication. While the figures show an upward trend in RC task performance over time, our statistical analysis indicates that this trend is primarily attributable to parameter size rather than release date itself: a regression controlling for both factors found parameter size to be a significant predictor of performance ($p = 0.006$), while release date was not ($p = 0.667$), even after excluding performance inflation from NoTA relations in Figure 3.

Observation: Statistical analysis highlights performance gains over time are largely explained by increases in model parameter size rather than by time-dependent improvements in natural language representation alone.

5.3 Relation-Specific Results

Qwen2.5-VL-32B and Llama-4-Scout achieved the highest accuracy across all three datasets when paired with BRE prompting. However, the discrepancy between weighted-average and macro-average scores indicates that performance varies significantly across different relation types. Figure 4 presents per-relation F_1 scores on the TACRED dataset, revealing that all models exhibit consistently low performance on the same subset of relations regardless of overall accuracy differences. Similar heatmaps for the TACREV and Re-TACRED dataset are included in Appendix A.3.

Relations such as 'members of', 'members,' and 'shareholders' prove particularly difficult, likely due to semantic overlap between relation definitions. When multiple relations share similar meanings, the relevance elicitation process may assign comparable scores across overlapping categories for a given entity pair, making correct distinction difficult. Notably, BRE's relevance elicitation identifies these challenging cases by providing interpretable scores, enabling more transparent failure analysis compared to baseline approaches.

Observation: Relevance elicitation enables transparent identification of challenging relation types, particularly those with semantic overlap that confound classification.

[3]Figure 2 baseline parameter sizes are not scaled since ODDA and ConsistRE with the DeepSeek-v3 (671B parameters) backbones were too large to visually scale.



Figure 4: F_1 scores for the varying relation types in the TACRED dataset. The relations are sorted based on their F_1 and the models are sorted based on their macro-F_1 scores.

5.4 Ablation Study

Ablation results exploring the impact of: (1) only including the binary decision in the methodology, and (2) only using the relevance elicitation without the binary step are included in Appendix A.4.

Impact of Relation Definitions The impact of relation-specific challenges are further evidenced in our ablation study, where removing relation definitions resulted in an average relative accuracy degradation of 4% across the TACRED family of datasets (p-value $= 0.01 < 0.05$). Including relation definitions serves as a semantic anchor grounds the binary classification task and provides explicit criteria for what constitutes a valid relationship beyond mere label matching. It also contributes more context regarding what a potential relation may include in scenarios where the sentence or relation

Dataset	Model	macro-average			weighted-average			micro-F_1
		Prec.	Rec.	F_1	Prec.	Rec.	F_1	Accuracy
Re-TACRED	Qwen2.5-VL-32B	$65.1_{\pm 0.24}$	$54.9_{\pm 0.4}$	$54.9_{\pm 0.27}$	$84.9_{\pm 0.09}$	$77.6_{\pm 0.08}$	$78.4_{\pm 0.1}$	$82.7_{\pm 0.09}$
	Llama-4-Scout-17B-16E	$50.0_{\pm 0.79}$	$63.2_{\pm 0.3}$	$51.4_{\pm 0.45}$	$83.3_{\pm 0.39}$	$77.3_{\pm 0.36}$	$78.9_{\pm 0.36}$	$77.4_{\pm 0.31}$
	Llama-3.3-70B	$54.3_{\pm 0.86}$	$64.7_{\pm 1.88}$	$54.1_{\pm 0.92}$	$82.8_{\pm 0.42}$	$73.0_{\pm 0.34}$	$74.8_{\pm 0.12}$	$76.8_{\pm 0.33}$
	Mixtral-8x22B	$45.0_{\pm 0.11}$	$60.1_{\pm 0.89}$	$47.8_{\pm 0.4}$	$78.0_{\pm 0.2}$	$72.2_{\pm 0.07}$	$73.0_{\pm 0.05}$	$74.9_{\pm 0.06}$
	Llama-3.1-70B	$45.9_{\pm 0.85}$	$59.5_{\pm 2.32}$	$47.8_{\pm 0.95}$	$79.6_{\pm 0.34}$	$73.4_{\pm 0.3}$	$74.5_{\pm 0.28}$	$74.5_{\pm 0.54}$
TACREV	Qwen2.5-VL-32B	$63.0_{\pm 0.14}$	$70.9_{\pm 0.28}$	$61.6_{\pm 0.19}$	$91.0_{\pm 0.04}$	$89.7_{\pm 0.05}$	$89.5_{\pm 0.04}$	$89.1_{\pm 0.05}$
	Llama-4-Scout-17B-16E	$47.9_{\pm 0.91}$	$78.0_{\pm 0.81}$	$55.8_{\pm 0.83}$	$89.7_{\pm 0.05}$	$85.1_{\pm 0.1}$	$86.4_{\pm 0.08}$	$83.0_{\pm 0.04}$
	Llama-3.3-70B	$51.1_{\pm 0.55}$	$72.8_{\pm 0.25}$	$56.6_{\pm 0.36}$	$88.1_{\pm 0.25}$	$85.9_{\pm 0.07}$	$86.4_{\pm 0.14}$	$84.2_{\pm 0.1}$
	Mixtral-8x22B	$41.7_{\pm 0.0}$	$71.4_{\pm 0.0}$	$49.7_{\pm 0.0}$	$88.1_{\pm 0.0}$	$83.4_{\pm 0.0}$	$84.8_{\pm 0.0}$	$81.6_{\pm 0.0}$
	Llama-3.1-70B	$44.5_{\pm 0.58}$	$70.6_{\pm 3.75}$	$50.7_{\pm 0.74}$	$85.5_{\pm 0.82}$	$83.8_{\pm 0.81}$	$84.0_{\pm 0.46}$	$80.9_{\pm 0.79}$
TACRED	Qwen2.5-VL-32B	$60.9_{\pm 0.22}$	$57.7_{\pm 0.15}$	$55.1_{\pm 0.1}$	$88.3_{\pm 0.03}$	$86.2_{\pm 0.03}$	$86.2_{\pm 0.03}$	$86.2_{\pm 0.04}$
	Llama-4-Scout-17B-16E	$47.3_{\pm 0.59}$	$66.0_{\pm 0.62}$	$52.3_{\pm 0.41}$	$87.4_{\pm 0.19}$	$82.3_{\pm 0.35}$	$84.1_{\pm 0.3}$	$81.0_{\pm 0.36}$
	Llama-3.3-70B	$50.9_{\pm 0.81}$	$65.4_{\pm 1.6}$	$54.7_{\pm 0.5}$	$87.2_{\pm 0.66}$	$82.9_{\pm 0.73}$	$84.5_{\pm 0.2}$	$82.0_{\pm 0.51}$
	Mixtral-8x22B	$43.8_{\pm 0.0}$	$64.7_{\pm 0.0}$	$49.8_{\pm 0.0}$	$86.1_{\pm 0.0}$	$81.8_{\pm 0.0}$	$83.4_{\pm 0.0}$	$80.5_{\pm 0.0}$
	Llama-3.1-70B	$45.0_{\pm 1.2}$	$62.3_{\pm 1.69}$	$49.4_{\pm 0.35}$	$84.5_{\pm 0.43}$	$81.4_{\pm 0.76}$	$82.3_{\pm 0.37}$	$79.1_{\pm 0.72}$
	Average Δ	-5.29	-0.31	-4.33	-0.21	-2.95	-2.05	-3.29

Table 5: (Ablation 1) Excluding relation definitions during prompting

Dataset	Model	macro-average			weighted-average			micro-F_1
		Prec.	Rec.	F_1	Prec.	Rec.	F_1	Accuracy
Re-TACRED	Qwen2.5-VL-32B	$46.5_{\pm 6.46}$	$8.6_{\pm 0.86}$	$12.5_{\pm 1.46}$	$68.9_{\pm 0.74}$	$61.1_{\pm 0.18}$	$55.0_{\pm 0.28}$	$71.2_{\pm 0.15}$
	Llama-4-Scout-17B-16E	$32.8_{\pm 3.52}$	$8.7_{\pm 0.14}$	$11.6_{\pm 0.4}$	$62.5_{\pm 1.77}$	$60.3_{\pm 0.16}$	$54.2_{\pm 0.29}$	$69.6_{\pm 0.12}$
	Llama-3.3-70B	$39.2_{\pm 5.13}$	$8.8_{\pm 0.48}$	$12.0_{\pm 0.72}$	$64.0_{\pm 1.04}$	$59.7_{\pm 0.08}$	$53.1_{\pm 0.15}$	$69.9_{\pm 0.12}$
	Mixtral-8x22B	$19.6_{\pm 1.88}$	$6.4_{\pm 0.86}$	$8.0_{\pm 1.23}$	$53.0_{\pm 0.94}$	$58.8_{\pm 0.23}$	$51.7_{\pm 0.33}$	$68.8_{\pm 0.16}$
	Llama-3.1-70B	$34.6_{\pm 1.86}$	$6.7_{\pm 0.39}$	$8.9_{\pm 0.47}$	$61.4_{\pm 1.24}$	$59.6_{\pm 0.12}$	$52.4_{\pm 0.17}$	$70.0_{\pm 0.14}$
TACREV	Qwen2.5-VL-32B	$35.6_{\pm 3.99}$	$8.5_{\pm 1.0}$	$11.6_{\pm 1.35}$	$75.8_{\pm 0.52}$	$80.8_{\pm 0.14}$	$74.9_{\pm 0.29}$	$81.9_{\pm 0.13}$
	Llama-4-Scout-17B-16E	$25.6_{\pm 1.63}$	$8.1_{\pm 1.03}$	$10.2_{\pm 0.95}$	$73.2_{\pm 0.42}$	$79.6_{\pm 0.16}$	$74.0_{\pm 0.17}$	$80.4_{\pm 0.19}$
	Llama-3.3-70B	$29.6_{\pm 4.31}$	$7.8_{\pm 0.58}$	$10.2_{\pm 0.76}$	$75.6_{\pm 0.87}$	$80.2_{\pm 0.14}$	$74.6_{\pm 0.09}$	$81.2_{\pm 0.25}$
	Mixtral-8x22B	$15.5_{\pm 0.0}$	$6.1_{\pm 0.0}$	$7.2_{\pm 0.0}$	$70.5_{\pm 0.0}$	$79.2_{\pm 0.0}$	$73.2_{\pm 0.0}$	$80.3_{\pm 0.0}$
	Llama-3.1-70B	$27.8_{\pm 3.98}$	$6.9_{\pm 1.08}$	$9.1_{\pm 1.68}$	$73.8_{\pm 0.9}$	$80.0_{\pm 0.17}$	$73.9_{\pm 0.26}$	$81.2_{\pm 0.14}$
TACRED	Qwen2.5-VL-32B	$34.2_{\pm 2.49}$	$7.5_{\pm 1.2}$	$10.1_{\pm 1.28}$	$74.3_{\pm 0.43}$	$79.2_{\pm 0.23}$	$73.1_{\pm 0.29}$	$80.0_{\pm 0.21}$
	Llama-4-Scout-17B-16E	$23.4_{\pm 1.26}$	$6.6_{\pm 0.76}$	$8.5_{\pm 0.77}$	$71.4_{\pm 0.58}$	$78.2_{\pm 0.2}$	$72.2_{\pm 0.18}$	$78.7_{\pm 0.14}$
	Llama-3.3-70B	$30.8_{\pm 2.13}$	$6.5_{\pm 0.44}$	$8.9_{\pm 0.66}$	$74.3_{\pm 0.95}$	$78.7_{\pm 0.2}$	$72.8_{\pm 0.23}$	$79.4_{\pm 0.24}$
	Mixtral-8x22B	$16.3_{\pm 4.08}$	$5.8_{\pm 0.83}$	$6.8_{\pm 0.97}$	$69.6_{\pm 1.02}$	$77.9_{\pm 0.17}$	$71.8_{\pm 0.14}$	$78.5_{\pm 0.19}$
	Llama-3.1-70B	$23.8_{\pm 3.0}$	$6.4_{\pm 0.94}$	$8.0_{\pm 1.07}$	$72.0_{\pm 0.52}$	$78.6_{\pm 0.12}$	$72.2_{\pm 0.19}$	$79.4_{\pm 0.09}$
	Average Δ	-26.69	-58.5	-47.54	-20.49	-11.22	-17.52	-7.52

Table 6: (Ablation 2) Excluding entity-type aware filtering such that all relations were evaluated for each entity pair

context are ambiguous.

Impact of Entity-Type Awareness Without entity-type awareness, the model must evaluate all relations for each entity pair regardless of semantic plausibility. Naturally, precision suffers from false positives as the expanded search space introduces implausible relation candidates that dilute relevance elicitation. However, TACRED's single-relation constraint implies that BRE always selects one final relation, regardless of how many relations are equally relevant to a sentence. Thus, BRE randomly selects a relation among the highest scores, and with diluted relevance scores, this random selection significantly reduces recall.

6 Conclusion and Future Work

We introduce Binary Relevance Elicitation (BRE), a zero-shot RC approach that addresses systematic over-prediction bias in LLM-based RC through simplified binary prompting and relevance-based filtering. BRE decomposes multi-class classification into independent binary decisions with quantitative relevance scoring, achieving state-of-the-art zero-shot performance on the TACRED family of datasets and matching or exceeding few-shot baselines. Notably, regression analysis reveals that parameter size, rather than model release date, is the primary driver of zero-shot relation classification performance, with $\log_{10}$(parameters) accounting for the majority of variance in outcomes; this finding, while counter to initial visual trends

across models, suggests that performance scales log-linearly with model capacity rather than being primarily attributable to architectural advances associated with newer model releases.

Our ablation studies further isolate the source of BRE's effectiveness. Removing the binary True/-False component while retaining relevance elicitation ranking results in negligible performance change across all models and datasets, whereas removing relevance elicitation and relying solely on binary classification leads to substantial degradation, particularly in precision and overall accuracy, as the model must select the correct relation from a much larger candidate pool (Appendix A.4). These findings confirm that relevance elicitation, rather than the binary verification step, is the primary contributor to BRE's performance gains. Additionally, analysis of our parsing pipeline shows minimal invalid outputs and no-score rates across most models, confirming that observed performance trends reflect genuine model behavior rather than extraction artifacts (Appendix A.5). However, the comparatively higher tie rates observed across models indicate that relevance elicitation, while effective at narrowing the candidate set, does not always fully disambiguate among top-scoring relations, pointing to an opportunity for further refinement.

While BRE substantially improves zero-shot RC performance and provides a robust framework for NoTA handling, performance variation across relation types remains an open challenge. Future work should explore relation-specific solutions and hybrid approaches combining BRE's filtering strengths with retrieval-augmented generation for ambiguous sentence-level cases, as well as methods for resolving residual ties within the relevance elicitation ranking. Additionally, given the small sample size underlying our parameter-size analysis ($N = 10$), further validation with a larger and more diverse set of models, alongside additional covariates such as training data quality and instruction tuning, would help confirm and refine this relationship. Despite these limitations, our work demonstrates that binary decomposition and systematic relevance filtering represent a significant advancement toward more reliable and interpretable zero-shot RC.

Limitations

BRE achieves state-of-the-art RC performance when dataset schemas provide well-documented relation definitions and entity type specifications. Since relation definitions are incorporated directly into the prompt and valid relations are filtered based on entity types, incomplete or inconsistent metadata may reduce prompt quality and system performance, as demonstrated in our ablation studies.

Additionally, our sentence-level evaluation strategy introduces substantial computational costs. For each entity pair in a sentence, BRE queries each valid relation type individually, resulting in worst-case complexity of $\mathcal{O}(|C| \times |E|^2 \times |R|)$, where C represents the set of sentences, $E \times E$ is the set of entity pairs, and R is the set of candidate relations. While this granular approach enables interpretable relevance scoring, it may become prohibitive for large-scale deployment without optimization strategies such as batching or parallel processing.

Ethical Considerations

Our approach leverages LLM reasoning capabilities but inherits biases from their pre-training corpora, which predominantly reflect internet content from Western, English-speaking populations (Bender et al., 2021). These biases produce systematic performance disparities across languages, domains, and demographic groups (Navigli et al., 2023). Since RC commonly populates knowledge bases for downstream applications, unmitigated biases may propagate through entire pipelines.

References

Toyin Aguda, Erik Wilson, Allan Anzagira, Simerjot Kaur, and Charese Smiley. 2025. Conservative bias in large language models: Measuring relation predictions. *arXiv preprint arXiv:2506.08120*.

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. Tacred revisited: A thorough evaluation of the tacred relation extraction task. *arXiv preprint arXiv:2004.14855*.

Hyeong-Ryeol Baek and Yong-Suk Choi. 2022. Enhancing targeted minority class prediction in sentence-level relation extraction. *Sensors*, 22(13):4911.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Mohan Raj Chanthran, Soon Lay Ki, Ong Huey Fang, and Bhawani Selvaretnam. 2026. Document-level zero-shot relation extraction with entity side information. *arXiv preprint arXiv:2601.07271*.

Yew Ken Chia, Lidong Bing, Soujanya Poria, and Luo Si. 2022. Relationprompt: Leveraging prompts to generate synthetic data for zero-shot relation triplet extraction. *Findings of the Association for Computational Linguistics: ACL 2022*.

Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and unfairness in information retrieval systems: New challenges in the llm era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6437–6447.

Kartik Detroja, CK Bhensdadia, and Brijesh S Bhatt. 2023. A survey on relation extraction. *Intelligent Systems with Applications*, 19:200244.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Joe Ellis, Jeremy Getman, Heather Simpson, Kira Griffitt, Hoa Trang Dang, Ralph Grishman, Heng Ji, Catherine DePrince, Thomas Riese, and Neil Kuster. 2015. TAC KBP 2015 Slot Descriptions. Technical report, Linguistic Data Consortium. Version 1.0.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Vladimir Geroimenko. 2025. Key challenges in prompt engineering. In *The Essential Guide to Prompt Engineering: Key Principles, Techniques, Challenges, and Security Risks*, pages 85–102. Springer.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *The Innovation*.

Bernal Jimenez Gutierrez, Nikolas McNeal, Clayton Washington, You Chen, Lang Li, Huan Sun, and Yu Su. 2022. Thinking about gpt-3 in-context learning for biomedical ie? think again. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4497–4512.

Jiale Han, Shuai Zhao, Bo Cheng, Shengkun Ma, and Wei Lu. 2022. Generative prompt tuning for relation classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3170–3185.

Tingting Hang, Shuting Liu, Jun Feng, Hamza Djigal, and Jun Huang. 2025. Few-shot relation extraction based on prompt learning: A taxonomy, survey, challenges and future directions. *ACM Computing Surveys*, 58(2):1–38.

George Hannah, Jacopo de Berardinis, Terry R Payne, Valentina Tamma, Andrew Mitchell, Ellen Piercy, Ewan Johnson, Andrew Ng, Harry Rostron, and Boris Konev. 2025. Relrae: Llm-based relationship extraction, labelling, refinement, and evaluation. *arXiv preprint arXiv:2507.03829*.

Pere-Lluís Huguet Cabot, Roberto Navigli, et al. 2021. Rebel: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics. EMNLP 2021*, pages 2370–2381. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Amirhossein Layegh, Amir H Payberah, and Mihhail Matskin. 2024. Rea: Refine-estimate-answer prompting for zero-shot relation extraction. In *International Conference on Applications of Natural Language to Information Systems*, pages 301–316. Springer.

Guozheng Li, Peng Wang, and Wenjun Ke. 2023. Revisiting large language models as zero-shot relation extractors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6877–6892.

Guozheng Li, Peng Wang, Jiajun Liu, Yikai Guo, Ke Ji, Ziyu Shang, and Zijie Xu. 2024. Meta in-context learning makes large language models better zero and few-shot relation extractors. *arXiv preprint arXiv:2404.17807*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Yubo Ma, Yixin Cao, Yong Ching Hong, and Aixin Sun. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Roberto Navigli, Simone Conia, and Björn Ross. 2023. Biases in large language models: origins, inventory, and discussion. *ACM Journal of Data and Information Quality*, 15(2):1–21.

Tapas Nayak, Navonil Majumder, Pawan Goyal, and Soujanya Poria. 2021. Deep neural approaches to relation triplets extraction: A comprehensive survey. *Cognitive Computation*, 13(5):1215–1232.

Abiola Obamuyide and Andreas Vlachos. 2018. Zero-shot relation classification as textual entailment. In *Proceedings of the first workshop on fact extraction and VERification (FEVER)*, pages 72–78.

Seongsik Park and Harksoo Kim. 2021. Improving sentence-level relation extraction through curriculum learning. *arXiv preprint arXiv:2107.09332*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. arxiv 2018. *arXiv preprint arXiv:1802.05365*, 12.

Yongbin Qin, Weizhe Yang, Kai Wang, Ruizhang Huang, Feng Tian, Shaolin Ao, and Yanping Chen. 2021. Entity relation extraction based on entity indicators. *Symmetry*, 13(4):539.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. 2023. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint arXiv:2305.14705*.

George Stoica, Emmanouil Antonios Platanios, and Barnabás Póczos. 2021. Re-tacred: Addressing shortcomings of the tacred dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 13843–13850.

Dianbo Sui, Xiangrong Zeng, Yubo Chen, Kang Liu, and Jun Zhao. 2023. Joint entity and relation extraction with set prediction networks. *IEEE transactions on neural networks and learning systems*, 35(9):12784–12795.

Wei Tang, Benfeng Xu, Yuyue Zhao, Zhendong Mao, Yifeng Liu, Yong Liao, and Haiyong Xie. 2022. Unirel: Unified representation and interaction for joint relational triple extraction. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 7087–7099.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. Gpt-re: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 3534–3547.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *Preprint*, arXiv:2409.12191.

Benfeng Xu, Quan Wang, Yajuan Lyu, Dai Dai, Yongdong Zhang, and Zhendong Mao. 2023. S2ynre: Two-stage self-training with synthetic data for low-resource relation extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8186–8207.

Liang Xu, Xiaoxuan Bu, and Xuetao Tian. 2024. Dynamic Prompt-Driven Zero-Shot Relation Extraction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:2900–2912.

Kai Zhang, Bernal Jiménez Gutiérrez, and Yu Su. 2023a. Aligning instruction tasks unlocks large language models as zero-shot relation extractors. In *Findings of the association for computational linguistics: ACL 2023*, pages 794–812.

Kai Zhang, Bernal Jimenez Gutierrez, and Yu Su. 2023b. Aligning instruction tasks unlocks large language models as zero-shot relation extractors. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45, Copenhagen, Denmark. Association for Computational Linguistics.

Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. 2024a. A comprehensive survey on relation extraction: Recent advances and new frontiers. *ACM Computing Surveys*, 56(11):1–39.

Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. 2024b. A comprehensive survey on relation extraction: Recent advances and new frontiers. *ACM Computing Surveys*, 56(11):1–39.

Yifan Zheng, Wenjun Ke, Qi Liu, Yuting Yang, Ruizhuo Zhao, Dacheng Feng, Jianwei Zhang, and Zhi Fang. 2024. Making llms as fine-grained relation extraction data augmentor. In *IJCAI*, pages 6660–6668.

Yijie Zhong, Yunfan Gao, Xiaolian Zhang, and Haofen Wang. 2025. Odda: An ooda-driven diverse data augmentation framework for low-resource relation extraction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 267–285.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Sizhe Zhou, Yu Meng, Bowen Jin, and Jiawei Han. 2024. Grasping the Essentials: Tailoring Large Language Models for Zero-Shot Relation Extraction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13462–13486. Association for Computational Linguistics.

Wenxuan Zhou and Muhao Chen. 2022. An improved baseline for sentence-level relation extraction. In *Proceedings of the 2nd Conference of the Asia-Pacific*

Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 161–168.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2024. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. In *Proceedings of the 2024 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies (volume 2: short papers)*, pages 358–370.

A Appendix

A.1 BRE Prompt Template

Input Prompt: Using only the context text provided, label the provided statements 'True' or 'False' followed by a numerical rating representing how relevant the statement is to the main idea in the context text.

Answer 'True' if the statement is explicitly stated in the context and meets the definition of the relationship or if the numerical rating is higher than 5. The rating is an integer ranging from 0 to 10 where 0 implies no relevance and 10 implies the statement is highly relevant to the context text.

Fill in the answer below:
context: c
statement: e_h and e_t are somehow connected through the relationship of 'r_i'.
relationship definition: d_i

LLM Output Example: True, 10. The statement is supported in the provided context because...

Figure 5: Generation prompt template not including output formatting instructions for brevity

A.2 Parsing Scripts

All models were evaluated using standardized Python scripts that parse LLM outputs to extract binary predictions and relevance scores. While most models adhered to the specified output format placing answers at the beginning of responses, Llama-4-Scout exhibited poor formatting behavior, frequently positioning final answers at the end of its responses despite explicit instructions. To ensure fair comparison, our parsing scripts were adapted to handle this variation by searching the entire response for the required information. Figures 6 and 7 contain the script for all LLMs (excluding Llama-4-Scout) and then the one for Llama-4-Scout, respectively.

For relevance elicitation scoring included in Figure 8, the relevance elicitation measure is extracted from the same LLM paragraph that contains the "true" or "false" classification, which increases the likelihood of extracting the correct relevance elicitation score. When a numeric score could not be extracted from the model output, we conservatively retained the relation as a potential candidate with a default score of 10 (the maximum). This default assignment strategy may contribute to both false positives through over-inclusion of uncertain relations (reducing precision) and missed correct relations when multiple candidates receive the default score and random selection chooses incorrectly (reducing recall). Despite these parsing challenges, the evaluation methodology remained consistent across all models, ensuring that performance differences reflect genuine model capabilities rather than evaluation artifacts.

A.3 Additional Relation-Specific Results

Figures 9 and 10 reveal a pattern: all evaluated models exhibit consistently low performance on approximately the same subset of relations across both Re-TACRED and TACREV datasets. This shared weakness pattern persists regardless of overall model accuracy differences or architectural variations, indicating that certain relations pose fundamental challenges beyond what model selection or prompting strategies can address. The consistency of this pattern across two independently revised versions of TACRED (which differ in their annotation corrections and label distributions) further confirms that these difficulties are robust to dataset-specific artifacts. Thus, while BRE's relevance elicitation improves overall performance and NoTA handling, it cannot fully compensate for relations that lack clear distinguishing features or require deeper contextual reasoning beyond what current LLMs can reliably provide in zero-shot settings.

A.4 Additional Ablations

Based on the ablation results in Table 7, the performance differences between the full method and the score-only variant are minimal across all three datasets and all evaluated metrics. The average deltas are consistently near zero, ranging from -1.9 for macro-average recall to -0.0 for weighted-average F_1, with micro-F_1 accuracy showing a negligible change of only -0.04. This consistency holds across the models evaluated, from smaller models such as Llama-3.2-3B to larger ones such as Qwen2.5-VL-32B, and across the three relation extraction datasets (Re-TACRED,

```python
def parse_binary_output(llm_output):
    llm_output = llm_output.replace('\n', ' ')
    parts = re.split(r'[^\w\s]', llm_output)
    for part in parts: # Assuming output starts response with 'true' or 'false'
        if 'true' in part.lower() or 'false' in part.lower():
            return 'true' in part.lower() and 'not true' not in part.lower(),
            ↪  int(parse_relevance_elicitation_score(part))
    return False # No 'true' or 'false' in the output
```

Figure 6: Python script to parse outputs for all LLM models except Llama-4-Scout-17B-16E

```python
def parse_binary_output_scout(llm_output):
    parts = llm_output.split('\n\n')

    # If the output starts response as requested (with 'true' or 'false')
    if 'true' in parts[0].lower() or 'false' in parts[0].lower():
        return 'true' in parts[0].lower(), int(parse_relevance_elicitation_score(parts[0]))

    # Identify where the final answer is located
    phrases = ['correct answer', 'final answer']
    for part in parts:
        for phrase in phrases:
            if phrase.lower() in part.lower():
                sp = ' '.join(part.split('\n')[-2:])
                return 'true' in sp.lower(), int(parse_relevance_elicitation_score(sp.lower()))
    return False
```

Figure 7: Python script to parse outputs for Llama-4-Scout-17B-16E

```python
def parse_relevance_elicitation_score(llm_output):
    parts = re.findall(r'[0-9]+', llm_output)
    if parts:
        result = parts[0]
    else:
        result = 10 # No confidence score found or we miss it; keep the result
    return result
```

Figure 8: Python script to parse outputs relevance elicitation scores for all LLM models

TACREV, and TACRED). Since the score-only configuration removes the binary True/False classification component while retaining the relevance elicitation ranking, the fact that performance remains largely stable indicates that the binary classification step contributes little additional discriminative value beyond what is already captured by the relevance elicitation scores. This provides strong evidence that the relevance elicitation mechanism itself, rather than the auxiliary binary verification, is the primary driver of the observed performance gains.

In contrast, the binary-only ablation shown in Table 8, which removes the relevance elicitation ranking and relies solely on True/False verification, exhibits substantial performance degradation across most metrics, with average deltas of -13.98 in weighted-average F_1 and -15.03 in micro-F_1 accuracy. Notably, this configuration shows a marked increase in macro-average recall ($+15.3$) alongside significant decreases in precision (-5.66 macro, -7.46 weighted), a pattern consistent with the model effectively guessing among a much larger pool of candidate relations rather than benefiting from the reduced candidate space that relevance elicitation provides. Because the binary-only setup lacks the relevance elicitation step to first narrow down plausible candidates, the model is more likely to indiscriminately affirm many relations as 'true', inflating recall while severely compromising precision and overall accuracy. This stands in sharp contrast to the score-only ablation, where performance remained stable, and together these results demonstrate that relevance elicitation, rather than the binary classification step, is the primary contributor to the method's performance, as it is responsible for effectively constraining the candidate set to a manageable and relevant subset prior to final classification.

Dataset	Model	macro-average			weighted-average			micro-F_1
		Prec.	Rec.	F_1	Prec.	Rec.	F_1	Accuracy
Re-TACRED	Llama-3.2-3B	$17.55_{\pm 0.37}$	$41.46_{\pm 1.33}$	$20.79_{\pm 0.51}$	$58.4_{\pm 0.43}$	$45.75_{\pm 0.86}$	$48.77_{\pm 0.74}$	$41.94_{\pm 0.6}$
	Llama-3.1-8B	$21.99_{\pm 0.24}$	$55.4_{\pm 0.94}$	$28.24_{\pm 0.29}$	$64.38_{\pm 0.31}$	$50.92_{\pm 0.79}$	$50.41_{\pm 0.95}$	$43.82_{\pm 0.94}$
	Llama-3.1-70B	$61.43_{\pm 0.5}$	$53.46_{\pm 1.3}$	$54.18_{\pm 1.0}$	$80.66_{\pm 0.22}$	$75.87_{\pm 0.31}$	$75.66_{\pm 0.23}$	$79.02_{\pm 0.33}$
	Llama-3.3-70B	$58.44_{\pm 2.02}$	$62.75_{\pm 3.27}$	$57.79_{\pm 1.6}$	$73.23_{\pm 8.43}$	$72.4_{\pm 2.85}$	$71.38_{\pm 4.47}$	$79.47_{\pm 0.34}$
	Llama-4-Scout-17B-16E	$55.94_{\pm 0.53}$	$72.77_{\pm 0.31}$	$60.93_{\pm 0.27}$	$86.62_{\pm 0.33}$	$83.55_{\pm 0.29}$	$84.23_{\pm 0.3}$	$83.38_{\pm 0.32}$
	Mistral-7B	$38.75_{\pm 1.82}$	$57.77_{\pm 1.51}$	$41.28_{\pm 2.82}$	$67.29_{\pm 4.9}$	$63.45_{\pm 2.83}$	$63.62_{\pm 3.31}$	$66.09_{\pm 1.79}$
	Mixtral-8x7B	$47.2_{\pm 1.55}$	$58.43_{\pm 1.16}$	$46.92_{\pm 0.49}$	$75.34_{\pm 0.38}$	$72.46_{\pm 0.49}$	$72.45_{\pm 0.32}$	$71.86_{\pm 0.53}$
	Mixtral-8x22B	$50.24_{\pm 0.49}$	$60.83_{\pm 0.84}$	$51.96_{\pm 0.39}$	$81.98_{\pm 0.25}$	$77.4_{\pm 0.26}$	$78.03_{\pm 0.17}$	$79.3_{\pm 0.02}$
	Qwen2.5-VL-32B	$68.13_{\pm 1.22}$	$66.11_{\pm 0.31}$	$63.25_{\pm 0.36}$	$86.86_{\pm 0.47}$	$82.72_{\pm 0.07}$	$83.44_{\pm 0.08}$	$85.74_{\pm 0.08}$
	DeepSeek-R1-Distill-Qwen-32B	$38.75_{\pm 0.182}$	$57.77_{\pm 1.51}$	$48.33_{\pm 0.55}$	$67.29_{\pm 4.9}$	$63.45_{\pm 2.83}$	$74.15_{\pm 0.25}$	$74.32_{\pm 0.48}$
TACREV	Llama-3.2-3B	$21.04_{\pm 0.28}$	$45.11_{\pm 1.56}$	$24.26_{\pm 0.46}$	$75.74_{\pm 0.23}$	$55.01_{\pm 0.33}$	$61.47_{\pm 0.17}$	$49.92_{\pm 0.25}$
	Llama-3.1-8B	$21.96_{\pm 0.52}$	$63.56_{\pm 0.87}$	$28.82_{\pm 0.62}$	$79.27_{\pm 0.32}$	$52.78_{\pm 1.13}$	$59.06_{\pm 1.01}$	$48.1_{\pm 1.01}$
	Llama-3.1-70B	$57.0_{\pm 0.51}$	$58.61_{\pm 1.91}$	$54.0_{\pm 0.93}$	$85.62_{\pm 0.32}$	$87.4_{\pm 0.16}$	$86.04_{\pm 0.18}$	$85.66_{\pm 0.16}$
	Llama-3.3-70B	$55.8_{\pm 0.39}$	$66.66_{\pm 0.68}$	$57.79_{\pm 0.21}$	$87.72_{\pm 0.36}$	$87.92_{\pm 0.28}$	$87.52_{\pm 0.3}$	$86.15_{\pm 0.23}$
	Llama-4-Scout-17B-16E	$47.84_{\pm 0.89}$	$72.33_{\pm 0.91}$	$53.93_{\pm 0.7}$	$90.37_{\pm 0.1}$	$87.83_{\pm 0.09}$	$88.32_{\pm 0.08}$	$86.26_{\pm 0.11}$
	Mistral-7B	$39.38_{\pm 0.38}$	$69.72_{\pm 1.0}$	$46.77_{\pm 0.48}$	$84.84_{\pm 0.17}$	$75.28_{\pm 0.33}$	$78.38_{\pm 0.21}$	$73.7_{\pm 0.26}$
	Mixtral-8x7B	$46.72_{\pm 1.35}$	$68.4_{\pm 0.62}$	$52.35_{\pm 1.06}$	$86.26_{\pm 0.23}$	$80.48_{\pm 0.6}$	$82.57_{\pm 0.43}$	$78.64_{\pm 0.54}$
	Mixtral-8x22B	$46.37_{\pm 0.4}$	$68.2_{\pm 1.28}$	$52.66_{\pm 0.64}$	$87.94_{\pm 0.05}$	$87.29_{\pm 0.16}$	$87.22_{\pm 0.09}$	$85.31_{\pm 0.15}$
	Qwen2.5-VL-32B	$60.47_{\pm 0.42}$	$70.87_{\pm 0.24}$	$62.22_{\pm 0.26}$	$91.29_{\pm 0.05}$	$90.43_{\pm 0.01}$	$90.47_{\pm 0.03}$	$89.84_{\pm 0.02}$
	DeepSeek-R1-Distill-Qwen-32B	$42.01_{\pm 0.51}$	$69.07_{\pm 0.3}$	$48.63_{\pm 0.38}$	$85.57_{\pm 0.2}$	$83.56_{\pm 0.38}$	$83.97_{\pm 0.18}$	$80.67_{\pm 0.44}$
TACRED	Llama-3.2-3B	$22.82_{\pm 0.31}$	$44.84_{\pm 0.88}$	$25.59_{\pm 0.2}$	$75.23_{\pm 0.29}$	$54.55_{\pm 0.33}$	$61.27_{\pm 0.29}$	$50.04_{\pm 0.33}$
	Llama-3.1-8B	$23.54_{\pm 0.24}$	$58.42_{\pm 1.05}$	$30.1_{\pm 0.22}$	$78.12_{\pm 0.06}$	$52.2_{\pm 0.9}$	$58.69_{\pm 0.8}$	$48.27_{\pm 0.82}$
	Llama-3.1-70B	$55.0_{\pm 0.98}$	$50.24_{\pm 1.38}$	$50.09_{\pm 0.54}$	$83.93_{\pm 0.17}$	$84.71_{\pm 0.19}$	$83.84_{\pm 0.14}$	$83.29_{\pm 0.29}$
	Llama-3.3-70B	$53.66_{\pm 0.29}$	$60.72_{\pm 1.45}$	$54.68_{\pm 1.01}$	$86.75_{\pm 0.32}$	$84.62_{\pm 0.07}$	$85.42_{\pm 0.18}$	$83.74_{\pm 0.1}$
	Llama-4-Scout-17B-16E	$48.95_{\pm 0.39}$	$62.52_{\pm 1.38}$	$52.65_{\pm 0.56}$	$88.09_{\pm 0.07}$	$84.96_{\pm 0.14}$	$86.07_{\pm 0.11}$	$84.12_{\pm 0.15}$
	Mistral-7B	$40.28_{\pm 0.26}$	$59.79_{\pm 1.19}$	$44.64_{\pm 0.22}$	$83.29_{\pm 0.12}$	$73.44_{\pm 0.24}$	$77.04_{\pm 0.2}$	$72.45_{\pm 0.27}$
	Mixtral-8x7B	$47.36_{\pm 0.39}$	$56.86_{\pm 0.93}$	$48.4_{\pm 0.53}$	$84.38_{\pm 0.17}$	$78.72_{\pm 0.44}$	$80.99_{\pm 0.32}$	$77.29_{\pm 0.34}$
	Mixtral-8x22B	$47.51_{\pm 0.31}$	$59.53_{\pm 0.08}$	$51.01_{\pm 0.01}$	$85.94_{\pm 0.33}$	$84.22_{\pm 0.02}$	$84.86_{\pm 0.17}$	$82.92_{\pm 0.01}$
	Qwen2.5-VL-32B	$59.25_{\pm 0.38}$	$57.62_{\pm 0.33}$	$55.77_{\pm 0.12}$	$88.42_{\pm 0.07}$	$86.59_{\pm 0.04}$	$87.09_{\pm 0.03}$	$86.58_{\pm 0.03}$
	DeepSeek-R1-Distill-Qwen-32B	$43.01_{\pm 0.65}$	$59.61_{\pm 0.87}$	$47.05_{\pm 0.44}$	$83.75_{\pm 0.28}$	$81.37_{\pm 0.3}$	$82.16_{\pm 0.21}$	$79.12_{\pm 0.35}$
	Average Δ	-0.46	-1.9	-0.73	-0.32	-0.36	-0.0	-0.04

Table 7: Score-only ablation considers the relevance elicitation ranking and not the binary True/False response

A.5 Model Output Statistics

Our parsing scripts introduced minimal error into the evaluation pipeline, as reflected by the consistently low rates of invalid output and no-score across nearly all models and datasets in Table 9. Invalid output, which captures instances where the parser could not extract a valid True/False response, and no-score, which captures instances where the system defaulted to the relevance elicitation score due to a parsing failure, both remain below 1.5% for the majority of models, with several models such as Llama-3.3-70B and Qwen2.5-VL-32B showing rates of exactly 0.0%. The one notable exception is Llama-3.2-3B, which exhibits elevated rates of approximately 7-8% for both metrics across all three datasets, suggesting this smaller model has more difficulty consistently adhering to the expected output format; however, even this represents a relatively small fraction of overall outputs. This overall low incidence of parsing failures confirms that the performance trends observed in our ablations are attributable to the underlying methods rather than artifacts of our extraction process. In contrast, the tie rates are considerably higher, ranging from roughly 8% to over 50% depending on the model and dataset. This indicates that while relevance elicitation is effective at narrowing the candidate relation set, it does not always resolve ambiguity down to a single top-scoring answer, leaving a non-trivial proportion of instances where multiple candidates receive equivalent scores. Taken together, these results suggest that relevance elicitation substantially improves the tractability of the classification task, but leaves room for improvement in fully disambiguating among the reduced candidate set.

Dataset	Model	macro-average			weighted-average			micro-F_1
		Prec.	Rec.	F_1	Prec.	Rec.	F_1	Accuracy
Re-TACRED	Llama-3.2-3B	$15.63_{\pm 0.29}$	$56.65_{\pm 1.12}$	$19.66_{\pm 0.49}$	$54.44_{\pm 0.75}$	$28.04_{\pm 1.32}$	$22.15_{\pm 2.12}$	$19.49_{\pm 1.09}$
	Llama-3.1-8B	$17.06_{\pm 0.16}$	$78.5_{\pm 0.77}$	$24.11_{\pm 0.23}$	$53.0_{\pm 0.4}$	$49.28_{\pm 2.15}$	$38.92_{\pm 2.38}$	$29.11_{\pm 1.38}$
	Llama-3.1-70B	$40.67_{\pm 0.39}$	$84.25_{\pm 0.49}$	$51.34_{\pm 0.41}$	$68.27_{\pm 0.23}$	$90.32_{\pm 0.12}$	$76.05_{\pm 0.15}$	$72.71_{\pm 0.05}$
	Llama-3.3-70B	$50.13_{\pm 1.54}$	$75.92_{\pm 4.75}$	$57.4_{\pm 2.02}$	$61.11_{\pm 11.57}$	$78.79_{\pm 3.94}$	$66.8_{\pm 7.23}$	$72.25_{\pm 2.81}$
	Llama-4-Scout-17B-16E	$53.05_{\pm 0.54}$	$85.16_{\pm 0.11}$	$61.76_{\pm 0.37}$	$80.43_{\pm 0.35}$	$59.87_{\pm 1.55}$	$64.25_{\pm 1.46}$	$60.59_{\pm 1.31}$
	Mistral-7B	$34.91_{\pm 0.73}$	$75.57_{\pm 3.11}$	$42.7_{\pm 0.7}$	$54.68_{\pm 6.89}$	$80.4_{\pm 5.29}$	$63.98_{\pm 6.16}$	$64.51_{\pm 1.56}$
	Mixtral-8x7B	$39.7_{\pm 0.1}$	$84.71_{\pm 0.19}$	$48.96_{\pm 0.11}$	$64.05_{\pm 0.32}$	$84.18_{\pm 3.01}$	$71.22_{\pm 1.17}$	$66.4_{\pm 1.43}$
	Mixtral-8x22B	$39.07_{\pm 0.36}$	$83.18_{\pm 0.75}$	$49.48_{\pm 0.45}$	$69.41_{\pm 0.18}$	$88.53_{\pm 0.36}$	$75.61_{\pm 0.03}$	$72.8_{\pm 0.22}$
	Qwen2.5-VL-32B	$63.89_{\pm 0.12}$	$72.5_{\pm 0.51}$	$62.9_{\pm 0.55}$	$83.98_{\pm 2.47}$	$49.51_{\pm 11.46}$	$56.93_{\pm 11.06}$	$56.18_{\pm 10.34}$
	DeepSeek-R1-Distill-Qwen-32B	$39.47_{\pm 0.35}$	$78.54_{\pm 0.95}$	$48.6_{\pm 0.5}$	$66.3_{\pm 0.6}$	$52.93_{\pm 4.12}$	$54.79_{\pm 3.52}$	$51.3_{\pm 3.22}$
TACREV	Llama-3.2-3B	$19.31_{\pm 0.38}$	$57.45_{\pm 1.17}$	$22.02_{\pm 0.31}$	$73.62_{\pm 1.58}$	$17.49_{\pm 2.09}$	$16.97_{\pm 3.38}$	$16.18_{\pm 1.74}$
	Llama-3.1-8B	$19.47_{\pm 0.32}$	$80.96_{\pm 0.87}$	$25.77_{\pm 0.46}$	$70.65_{\pm 0.38}$	$32.22_{\pm 4.02}$	$33.2_{\pm 5.15}$	$24.92_{\pm 2.69}$
	Llama-3.1-70B	$40.38_{\pm 0.31}$	$88.36_{\pm 0.87}$	$51.38_{\pm 0.36}$	$76.85_{\pm 0.11}$	$96.98_{\pm 0.17}$	$84.98_{\pm 0.07}$	$79.63_{\pm 0.05}$
	Llama-3.3-70B	$49.36_{\pm 0.31}$	$86.53_{\pm 0.06}$	$59.24_{\pm 0.34}$	$79.21_{\pm 0.22}$	$94.94_{\pm 1.27}$	$85.88_{\pm 0.42}$	$82.52_{\pm 0.45}$
	Llama-4-Scout-17B-16E	$48.88_{\pm 0.69}$	$87.32_{\pm 0.59}$	$58.15_{\pm 0.56}$	$89.07_{\pm 0.63}$	$53.14_{\pm 2.78}$	$62.3_{\pm 2.58}$	$61.07_{\pm 2.14}$
	Mistral-7B	$36.09_{\pm 0.13}$	$82.59_{\pm 0.3}$	$46.07_{\pm 0.13}$	$75.08_{\pm 0.24}$	$89.53_{\pm 1.62}$	$80.65_{\pm 0.63}$	$73.48_{\pm 0.68}$
	Mixtral-8x7B	$41.03_{\pm 0.23}$	$87.94_{\pm 0.2}$	$52.17_{\pm 0.18}$	$77.73_{\pm 0.76}$	$85.65_{\pm 3.17}$	$79.98_{\pm 1.29}$	$71.73_{\pm 1.35}$
	Mixtral-8x22B	$39.28_{\pm 0.11}$	$88.31_{\pm 0.31}$	$50.78_{\pm 0.11}$	$77.25_{\pm 0.08}$	$95.68_{\pm 0.27}$	$84.67_{\pm 0.1}$	$80.41_{\pm 0.1}$
	Qwen2.5-VL-32B	$59.76_{\pm 0.21}$	$80.59_{\pm 0.39}$	$63.98_{\pm 0.28}$	$90.68_{\pm 1.55}$	$43.26_{\pm 10.7}$	$53.41_{\pm 10.85}$	$54.92_{\pm 9.25}$
	DeepSeek-R1-Distill-Qwen-32B	$38.67_{\pm 0.44}$	$83.98_{\pm 0.75}$	$48.3_{\pm 0.46}$	$76.51_{\pm 0.52}$	$48.83_{\pm 3.92}$	$55.25_{\pm 3.44}$	$52.33_{\pm 2.86}$
TACRED	Llama-3.2-3B	$20.76_{\pm 0.29}$	$55.66_{\pm 1.68}$	$23.43_{\pm 0.36}$	$73.44_{\pm 0.28}$	$18.93_{\pm 2.44}$	$19.21_{\pm 4.2}$	$18.2_{\pm 2.07}$
	Llama-3.1-8B	$20.55_{\pm 0.07}$	$74.94_{\pm 1.09}$	$26.96_{\pm 0.19}$	$71.65_{\pm 0.45}$	$34.98_{\pm 2.75}$	$36.99_{\pm 3.36}$	$27.5_{\pm 1.77}$
	Llama-3.1-70B	$40.81_{\pm 0.42}$	$76.33_{\pm 0.82}$	$50.13_{\pm 0.32}$	$75.71_{\pm 0.15}$	$94.96_{\pm 0.28}$	$83.68_{\pm 0.04}$	$78.57_{\pm 0.05}$
	Llama-3.3-70B	$48.72_{\pm 0.11}$	$74.31_{\pm 0.07}$	$56.01_{\pm 0.14}$	$79.16_{\pm 0.31}$	$89.72_{\pm 0.59}$	$83.73_{\pm 0.44}$	$80.42_{\pm 0.41}$
	Llama-4-Scout-17B-16E	$49.21_{\pm 0.34}$	$74.8_{\pm 0.7}$	$55.84_{\pm 0.37}$	$87.76_{\pm 0.42}$	$49.09_{\pm 2.03}$	$59.16_{\pm 1.96}$	$58.32_{\pm 1.62}$
	Mistral-7B	$37.06_{\pm 0.09}$	$74.31_{\pm 0.22}$	$45.14_{\pm 0.11}$	$74.15_{\pm 0.27}$	$88.72_{\pm 2.56}$	$79.93_{\pm 1.0}$	$72.93_{\pm 1.06}$
	Mixtral-8x7B	$42.25_{\pm 0.06}$	$78.54_{\pm 0.39}$	$50.82_{\pm 0.13}$	$75.84_{\pm 0.33}$	$86.55_{\pm 2.69}$	$79.78_{\pm 1.32}$	$71.96_{\pm 1.3}$
	Mixtral-8x22B	$40.0_{\pm 0.13}$	$77.67_{\pm 0.06}$	$49.89_{\pm 0.2}$	$76.83_{\pm 0.5}$	$91.26_{\pm 1.87}$	$82.78_{\pm 0.54}$	$78.43_{\pm 0.59}$
	Qwen2.5-VL-32B	$57.83_{\pm 0.09}$	$66.26_{\pm 0.31}$	$58.05_{\pm 0.27}$	$86.84_{\pm 2.05}$	$54.6_{\pm 9.05}$	$64.84_{\pm 7.41}$	$64.36_{\pm 6.68}$
	DeepSeek-R1-Distill-Qwen-32B	$39.54_{\pm 0.33}$	$73.66_{\pm 0.96}$	$47.58_{\pm 0.51}$	$76.59_{\pm 0.73}$	$50.28_{\pm 3.26}$	$57.27_{\pm 2.83}$	$54.08_{\pm 2.35}$
	Average Δ	-5.66	$+15.3$	-0.95	-7.46	-9.11	-13.98	-15.03

Table 8: Binary-only ablation considers the True/False binary response and not the relevance elicitation ranking

Dataset	Model	Invalid Output	No-Score	Tie Rates
Re-TACRED	Llama-3.2-3B	$6.875_{\pm 0.092}$	$6.7704_{\pm 0.0754}$	$33.537_{\pm 0.4}$
	Llama-3.1-8B	$0.9029_{\pm 0.0897}$	$0.8203_{\pm 0.0635}$	$53.6742_{\pm 1.12}$
	Llama-3.1-70B	$1.1672_{\pm 0.0161}$	$1.1638_{\pm 0.0175}$	$50.8824_{\pm 0.2}$
	Llama-3.3-70B	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$36.7596_{\pm 2.83}$
	Llama-4-Scout	$0.5486_{\pm 0.1732}$	$0.1814_{\pm 0.1323}$	$20.8321_{\pm 0.49}$
	Mistral-7B	$0.1472_{\pm 0.0116}$	$0.0638_{\pm 0.0048}$	$48.4066_{\pm 4.47}$
	Mixtral-8x7B	$0.0047_{\pm 0.0031}$	$0.0151_{\pm 0.0019}$	$50.3324_{\pm 2.19}$
	Mixtral-8x22B	$0.5088_{\pm 0.012}$	$0.7329_{\pm 0.0053}$	$46.5395_{\pm 0.6}$
	Qwen2.5-VL-32B	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$14.1511_{\pm 4.73}$
	DeepSeek-R1-Distill-Qwen-32B	$0.0033_{\pm 0.0014}$	$0.048_{\pm 0.0102}$	$24.0543_{\pm 2.23}$
TACREV	Llama-3.2-3B	$7.8906_{\pm 0.1481}$	$7.1393_{\pm 0.0721}$	$25.1931_{\pm 0.73}$
	Llama-3.1-8B	$1.1367_{\pm 0.1694}$	$0.9979_{\pm 0.1001}$	$39.2765_{\pm 1.15}$
	Llama-3.1-70B	$1.0822_{\pm 0.0582}$	$1.0801_{\pm 0.0567}$	$40.7312_{\pm 0.3}$
	Llama-3.3-70B	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$34.1769_{\pm 0.67}$
	Llama-4-Scout	$1.665_{\pm 0.6556}$	$1.2021_{\pm 0.681}$	$12.0653_{\pm 0.75}$
	Mistral-7B	$0.1417_{\pm 0.0051}$	$0.0748_{\pm 0.0089}$	$39.1463_{\pm 1.19}$
	Mixtral-8x7B	$0.0044_{\pm 0.0028}$	$0.0118_{\pm 0.0048}$	$37.6169_{\pm 2.12}$
	Mixtral-8x22B	$0.5256_{\pm 0.016}$	$0.7697_{\pm 0.0339}$	$37.4573_{\pm 0.3}$
	Qwen2.5-VL-32B	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$7.9154_{\pm 2.08}$
	DeepSeek-R1-Distill-Qwen-32B	$0.0047_{\pm 0.002}$	$0.507_{\pm 0.0042}$	$17.6066_{\pm 1.43}$
TACRED	Llama-3.2-3B	$8.0146_{\pm 0.1861}$	$7.3173_{\pm 0.07}$	$25.9359_{\pm 0.82}$
	Llama-3.1-8B	$1.118_{\pm 0.0914}$	$0.977_{\pm 0.05}$	$40.8356_{\pm 1.11}$
	Llama-3.1-70B	$1.1151_{\pm 0.0371}$	$1.1104_{\pm 0.04}$	$40.8124_{\pm 0.41}$
	Llama-3.3-70B	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$30.8047_{\pm 0.09}$
	Llama-4-Scout	$1.4025_{\pm 0.5118}$	$0.9141_{\pm 0.52}$	$11.6436_{\pm 0.72}$
	Mistral-7B	$0.1531_{\pm 0.0085}$	$0.0806_{\pm 0.0097}$	$39.4906_{\pm 1.67}$
	Mixtral-8x7B	$0.0061_{\pm 0.0034}$	$0.0128_{\pm 0.00414}$	$39.3088_{\pm 0.98}$
	Mixtral-8x22B	$0.5162_{\pm 0.0131}$	$0.7156_{\pm 0.000725}$	$35.1699_{\pm 1.5}$
	Qwen2.5-VL-32B	$0.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$11.6616_{\pm 2.64}$
	DeepSeek-R1-Distill-Qwen-32B	$0.0052_{\pm 0.0017}$	$0.0574_{\pm 0.0089}$	$18.8329_{\pm 1.39}$

Table 9: Reported average percentage % $\pm$ standard deviation

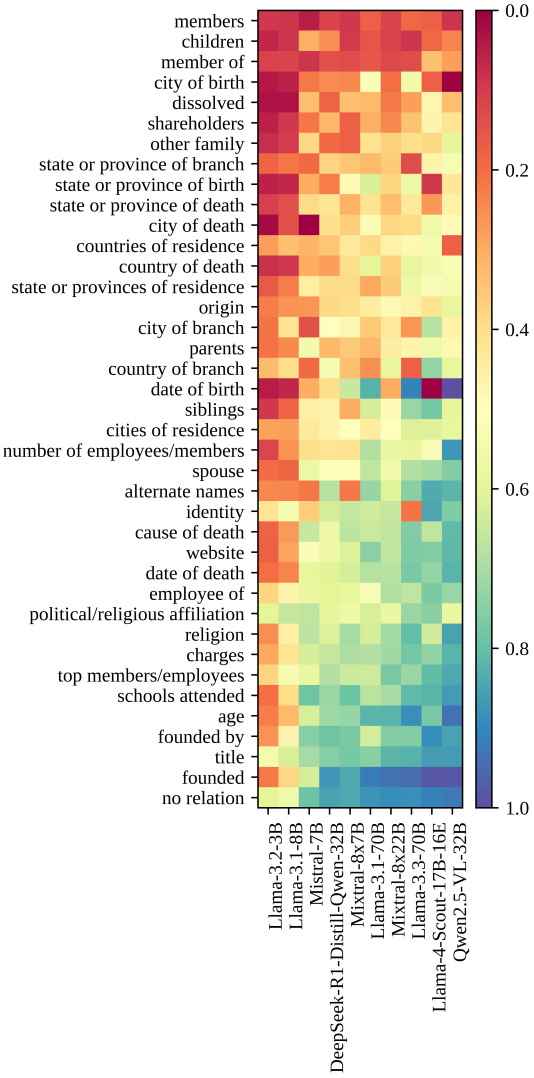


Figure 9: F_1 scores for the varying relation types in the Re-TACRED dataset. The relations are sorted based on their F_1 and the models are sorted based on their macro-F_1 scores.

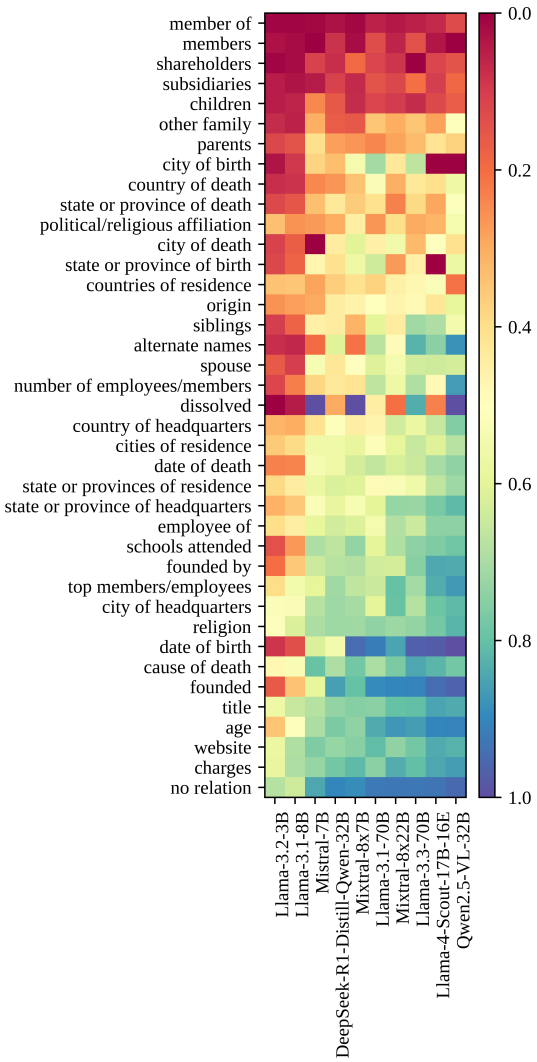


Figure 10: F_1 scores for the varying relation types in the TACREV dataset. The relations are sorted based on their F_1 and the models are sorted based on their macro-F_1 scores.

Distill-RE: Few-Shot Relation Extraction via Teacher LLM Reasoning Distillation*

Danielle Sullivan-Pao, Sheila Alemany, Olga Simek
MIT Lincoln Laboratory
Lexington, MA, USA

Abstract

Despite their generalization capabilities, large language models (LLMs) have counterintuitively underperformed smaller pre-trained language models (PLMs) on few-shot relation extraction (RE), the task of identifying semantic relationships between entities in text. This gap is primarily due to a systematic over-prediction bias in LLMs. We propose Distill-RE, a two-stage knowledge distillation framework in which a teacher LLM generates chain-of-thought reasoning over existing training examples to supervise two student models: a classification model that predicts relations and a confirmation model that filters false positives. Across student scales from 3B to 70B, Distill-RE establishes a new state-of-the-art among RE models, with its top-performing configuration outperforming the strongest baseline by over 31.5 micro-F_1 points averaged across TACRED, TACREV, and Re-TACRED at 8-, 16-, and 32-shot settings. Without synthesizing new relation instances, we outperform most data augmentation baselines and comes within 1.8 micro-F_1 points on average of the strongest.

1 Introduction

Relation extraction (RE) identifies semantic relationships between entities in text, transforming unstructured language into structured knowledge for downstream tasks such as knowledge graph construction, question answering, and semantic search (Zhao et al., 2024; Nayak et al., 2021). Recent advances leveraging pre-trained language models (PLMs) and large language models (LLMs) have achieved strong RE performance, but typically require substantial labeled data (Han et al., 2018; Cabot and Navigli, 2021; Zhao et al., 2024). Annotating RE datasets is particularly challenging due to semantic ambiguity and nuanced relational distinctions that can lead annotators to interpret the same relationship differently, resulting in label noise and low inter-annotator agreement (Hang et al., 2025; Stoica et al., 2021; Alt et al., 2020).

Few-shot RE (FSRE) methods aim to match the performance of fully-supervised models while training on only a small number of labeled examples per relation, directly reducing the annotation burden (Gao et al., 2019). LLM-based few-shot approaches have paradoxically underperformed compared to smaller PLM-based methods (Wan et al., 2023; Xu et al., 2023b; Ma et al., 2023b) due to two factors: (1) limited RE task representation in pre-training data (Wang et al., 2022; Zhang et al., 2023), and (2) systematic over-prediction bias, where models incorrectly assign positive relation labels to instances that should be classified as "no relation" (Wan et al., 2023; Ma et al., 2023b). Since "no relation" comprises roughly 80% of instances in standard benchmarks (Baek and Choi, 2022), this bias has an outsized impact on performance.

We propose Distill-RE, a two-stage knowledge distillation RE framework using CoT-augmented supervision, motivated by two insights. First, augmenting labels with teacher-generated chain-of-thought (CoT) explanations is more sample-efficient than learning from labels alone, making reasoning patterns explicit and enabling better generalization from fewer examples. Second, over-prediction bias can be mitigated through task decomposition: rather than training a single model

*DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Combatant Commands under Air Force Contract No. FA8702-15-D-0001 or FA8702-25-D-B002. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Combatant Commands.

© 2026 Massachusetts Institute of Technology. Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 359–378
October 28, 2026 ©2026 Association for Computational Linguistics

to balance recall and precision, we use specialized classification and confirmation models that optimize for complementary objectives under CoT-augmented supervision. Our main contributions are:

- Distill-RE sets a new state-of-the-art among RE models, outperforming the strongest baseline by over 31.5 micro-F_1 points on average across datasets and shot settings, without relying on LLM-generated relation instances.
- The confirmation stage reduces over-prediction bias, improving over the classification-only stage by 16.3 macro-F_1 and 36.9 micro-F_1 points on average across 135 few-shot experimental configurations (3 backbone architectures $\times$ 5 seeds $\times$ 3 k-shot values $\times$ 3 datasets).
- CoT-augmented supervision consistently improves performance across model sizes and datasets, with mean gains of 9.1 macro-F_1 and 7.6 micro-F_1 across the same configurations.

2 Background

2.1 Sentence-Level Closed Relation Extraction

Given a sentence s with entity set $\mathcal{E}$ and predefined relation types $\mathcal{R}$, RE produces triplets $\langle e_h, r, e_t \rangle$ where $e_h, e_t \in \mathcal{E}$ are the head and tail entities, $e_h \neq e_t$, and $r \in \mathcal{R}$ is the predicted relation type. Sentence-level RE restricts both entities and evidence to a single sentence, unlike document-level RE where evidence may span multiple sentences. Many RE approaches also leverage entity type information, either as hard constraints (Han et al., 2022a; Yang et al., 2024) or learned features (Cabot and Navigli, 2021), to narrow the relation search space and guide disambiguation. We adopt the former, incorporating entity types directly into our prompt structure and listing only the relations compatible with each entity pair's types, narrowing the candidate relation set.

2.2 Few-Shot Relation Extraction

Few-shot relation extraction aims to achieve strong RE performance with minimal labeled data, typically evaluated under an N-way k-shot setting where N is the number of relation types and k is the number of labeled examples per relation. The primary challenges include: insufficient data for learning discriminative inter-class features, overfitting when fine-tuning with small k, and class imbalance arising from the long-tail distribution of relation types in natural text, where a small number of relations dominate and the majority are rare (Qin et al., 2021; Baek and Choi, 2022).

2.3 The Over-Prediction Problem

A central challenge for LLM-based RE is task-specific resistance to predicting "no relation," as these models exhibit a bias toward assigning a named relation to entity pairs even when none exists (Wan et al., 2023; Ma et al., 2023b). This failure mode is structurally amplified by the RE task formulation. Because many relation types are asymmetric, models must evaluate numerous ordered entity pairs, the vast majority of which do not express any predefined relation (Qin et al., 2021; Baek and Choi, 2022). The resulting extreme class imbalance increases false positives and degrades precision, making reliable "no relation" prediction a central challenge that motivates the two-stage design of our method (Section 4).

3 Related Work

3.1 PLM-Based Few-Shot RE

Prompt-based methods mitigate the mismatch between token-level PLM pre-training objectives and structured RE by reformulating the task as masked language modeling (Chen et al., 2022b; Han et al., 2022b; Zhang et al., 2024) or conditional text generation (Han et al., 2022a; Cabot and Navigli, 2021). Retrieval augmentation has been used to further improve PLM-based label prediction via similarity-weighted neighbor distances (Chen et al., 2022a). Despite these advances, PLM-based approaches fail to leverage the reasoning capacity and sample efficiency that emerge at scale in LLMs (Wei et al., 2022a), reducing their ability to capture fine-grained relation structure with limited data.

3.2 LLM-Based FSRE via In-Context Learning

When applied as frozen backbones under in-context learning (ICL), LLMs are susceptible to prompt-induced biases including majority label and recency effects (Zhao et al., 2021). GPT-RE (Wan et al., 2023) and RAG4RE (Efeoglu and Paschke, 2024) attempt to mitigate prompt sensitivity through improved demonstration retrieval, but both rely on access to the full training set at inference and depart from the strict few-shot setting (Ma et al., 2023a). More fundamentally, because model parameters

remain frozen, systematic ICL biases, including positive relation over-prediction, are largely unaddressed.

3.3 LLM-Augmented Data Generation

A complementary research direction uses LLMs purely as data generators to augment the few-shot training set of smaller, fine-tuned PLMs. Unleash (Xu et al., 2023b) and Diverse Samples (Li et al., 2025) generate schema-constrained synthetic examples to augment PLM classifiers' training sets. ConsistRE (Zheng et al., 2024) incorporates relation-relevant keywords to guide generation and preserve relation consistency, but assumes access to a sufficiently informative external corpus for keyword generation, a requirement that may not hold in practice. ODDA (Zhong et al., 2025) selects training instances with moderate RE model confidence as demonstrations for LLM-based generation, then filters the generated dataset via a global diversity objective. These methods treat the LLM as an offline data source and never directly leverage LLM reasoning for relation classification. Crucially, data generation pipelines are orthogonal to RE model design: they ask how training data can be improved for a fixed model, whereas our method asks how the model can be improved given fixed training data.

3.4 Hybrid LLM/PLM Methods

Beyond offline data generation, several methods incorporate LLM outputs directly into PLM-based training or inference. S^2ynRE (Xu et al., 2023a) fine-tunes an LLM on few-shot gold data to synthesize in-domain training instances, combining synthetic and gold supervision in a two-stage training loop. DSARE (Liu et al., 2024) generates LLM-augmented data to train a PLM-based kNN retrieval model, and uses an LLM to adjudicate ties in the kNN inference stage. TKRE (Guo et al., 2025) generates relational logic explanations conditioned on gold labels to guide PLM pre-training. In all cases, the LLM serves as a data synthesis component and is never fine-tuned to reason over relational semantics. Notably, DSARE leverages the LLM in a zero-shot setting to break ties between down-selected relations.

3.5 LLM Fine-Tuning for RE

In contrast to ICL, fine-tuning updates LLM parameters to internalize task structure, reducing sensitivity to prompt artifacts and alleviating reliance on complex retrieval pipelines (Liu et al., 2022). CoT training (Wei et al., 2022b) extends zero-shot CoT prompting to parameter updates, using teacher-generated explanations to make implicit relational patterns explicit, enabling data-efficient fine-tuning via distillation (Hsieh et al., 2023; Ho et al., 2023). LoRA (Hu et al., 2021) addresses the computational cost of LLM adaptation via low-rank weight updates, achieving strong performance while mitigating overfitting in low-data regimes. LoRA-based fine-tuning for RE has been explored in full-data settings (Efeoglu and Paschke, 2024) but never in the few-shot regime. To our knowledge, no prior work combines parameter-efficient adaptation with reasoning-augmented supervision for FSRE, a gap our method directly addresses.

4 Methodology

Our approach distills reasoning from a large teacher LLM into student LLMs via a two-stage pipeline: Stage 1, a classification stage that predicts relation r_{pred} for entity pair (e_h, e_t), and Stage 2, a confirmation Stage that validates r_{pred} by predicting $y_{conf} \in \{True, False\}$, filtering false positives and mitigating over-prediction. For each stage, we construct CoT-augmented training data using a frozen teacher LLM, with hard negative sampling in Stage 2. Student LLMs are then fine-tuned on teacher-generated CoT completions and gold labels (relation verbalizations for Stage 1, $True/False$ for Stage 2) via causal language modeling.

4.1 Prompt Construction

All prompts used in our pipeline are provided in full in Appendix A.1. Teacher prompts were engineered using Mixtral-8x22B (Mixtral-8x22B-Instruct-v0.1) and student prompts using Mistral-7B (Mistral-7B-Instruct-v0.2) with no downstream quantitative optimization.

Type-Aware Relation Filtering For Stage 1, we apply type-aware filtering to reduce the candidate relation set. For each entity pair (e_h, e_t) with types (t_h, t_t), only entity type-compatible relations are included in the prompt, producing a filtered set $\mathcal{R}_{valid} \subset \mathcal{R}$ (e.g., "city_of_death" is excluded when both entities are of type "person"). Compatibility is determined using the TACRED family dataset annotation guidelines (Ellis et al., 2015; Stoica et al., 2021), and prior work confirms this filtering improves classification accuracy (Zhang et al., 2023; Yang et al., 2024).

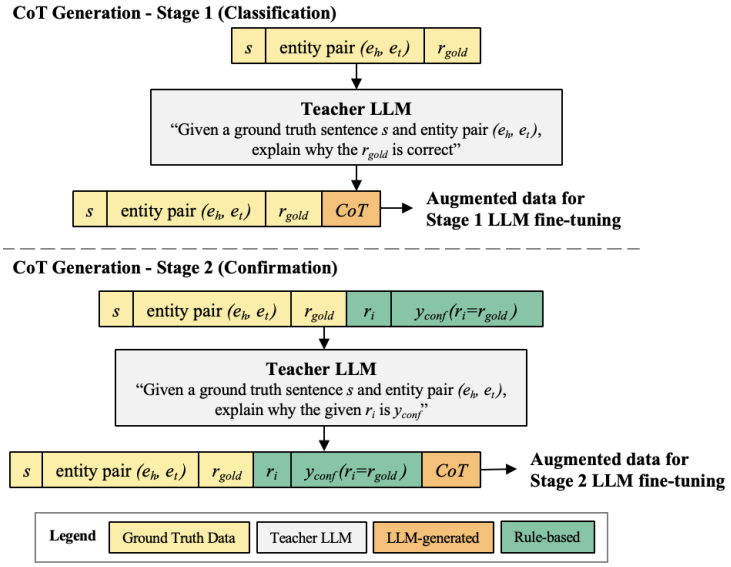


Figure 1: CoT generation pipeline. A zero-shot teacher LLM augments every training sample with a CoT explanation. For Stage 1, the teacher explains why a sample matches its ground-truth relation; for Stage 2, the teacher explains why a candidate r_i, the Stage 1 prediction or a sampled candidate (Section 4.2), is $True$ or $False$. Explanations are paired with labels to fine-tune the student on prediction and reasoning jointly.

Teacher Prompt Templates We use Mixtral-8x22B (AI, 2024) as the frozen teacher model for all CoT training data augmentation, an open-weight model that achieved state-of-the-art reasoning performance at the time of our experiments. The classification teacher prompt provides entities (e_h, e_t), entity types (t_h, t_t), gold relation r_{gold}, its definition d_{gold}, and instructions to output a supporting explanation of the r_{gold} label. The confirmation teacher prompt provides (e_h, e_t), Stage 1 prediction r_{pred}, its definition d_{pred}, the ground truth label $y_{conf} \in \{True, False\}$, and instructions to output a supporting explanation of y_{conf}.

Student Prompt Templates The classification student prompt provides input sentence s, entity pair (e_h, e_t) with types (t_h, t_t), candidate relations $\mathcal{R}_{valid}$ with definitions, and instructions to predict r_{pred} followed by a supporting explanation. The confirmation student prompt provides input sentence s, entity pair (e_h, e_t), Stage 1 prediction r_{pred}, its definition d_{pred}, and instructions to predict y_{conf} followed by a supporting explanation.

4.2 Training Data Construction

Stage 1: Teacher-Augmented Classification Data Each Stage 1 training sample is augmented with a teacher-generated CoT explanation for why r_{gold} holds for (e_h, e_t) in sentence s.

Stage 2: Confirmation Data from Stage 1 Predictions and Sampling To address over-prediction, we introduce a confirmation stage. After training the Stage 1 LLM, we construct Stage 2 data from its predictions on the training set. For samples where $r_{pred} \neq$ "no relation," correct predictions receive $y_{conf} = True$ and incorrect ones receive $y_{conf} = False$. For samples where $r_{gold} =$ "no relation," we generate false examples by pairing (e_h, e_t) with relations randomly selected from $\mathcal{R}_{valid}$.

To increase difficulty, we augment with hard negatives. For each named-relation (i.e., $r_{pred} \neq$ "no relation") prediction, we sample 2 of the 3 most semantically similar type-compatible relations (selected via BERT-base-uncased (Devlin et al., 2019) embeddings of relation verbalizations) to create additional False training examples. The number of hard negatives was selected via hyperparameter sweep (see Subsection 5.4). Each Stage 2 training sample is constructed around a candidate r_i, drawn either from Stage 1 predictions (r_{pred}) or from the sampled negatives. At inference, Stage 2 is only executed when $r_{pred} \neq$ "no relation"; otherwise the pipeline directly outputs "no relation."

4.3 Student LLM Fine-tuning

We fine-tune separate student LLMs for classification and confirmation on teacher-generated prompt completions and relation labels (r_{gold} for Stage 1,

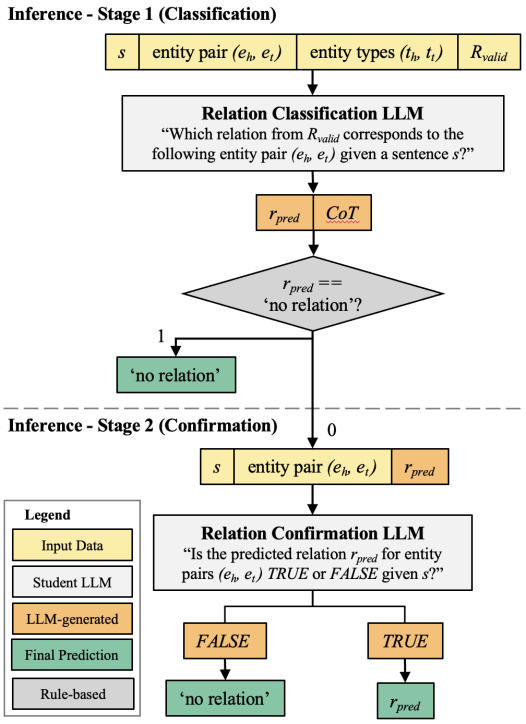


Figure 2: Inference pipeline. Stage 1, the LLM predicts relation r_{pred} with CoT reasoning given input sentence s, entity pair (e_h, e_t) with types (t_h, t_t), and candidate relations $\mathcal{R}_{valid}$. If r_{pred} ="no relation," the pipeline directly outputs "no relation." Stage 2, the LLM verifies r_{pred}. Samples predicted $True$ retain r_{pred}; those predicted $False$ are relabeled as "no relation."

y_{conf} for Stage 2) using standard causal language modeling (CLM), minimizing:

$$\mathcal{L}_{CLM} = -\sum_{t=1}^{T} \log P_\theta(y_t \mid y_{<t}, x) \qquad (1)$$

where x is the templated prompt, $y = (y_1, \ldots, y_T)$ is the target sequence (gold label followed by CoT reasoning), and θ are the student model parameters. The classification student generates r_{pred} followed by a supporting explanation; the confirmation student generates $y_{conf} \in \{True, False\}$ followed by reasoning. Unlike prior work applying CoT to RE via in-context learning (Wan et al., 2023) or PLM fine-tuning (Guo et al., 2025), we are, to our knowledge, the first to fine-tune student LLMs on teacher-generated CoT completions for RE, training student LLMs on explicit RE reasoning patterns rather than class label associations alone.

4.4 Answer Parsing

We extract predictions using regular expressions targeting the relation verbalization or $True/False$ decision at the start of each response. For Stage 1, we compute BERT embeddings of the parsed string and select the most similar relation in $\mathcal{R}$:

$$r_{pred} = \arg\max_{r \in \mathcal{R}} \cos(\mathbf{e}_{parsed}, \mathbf{e}_r) \qquad (2)$$

where $\mathbf{e}_{parsed}$ is the BERT embedding of the parsed output string and $\mathbf{e}_r$ is the embedding of relation r's verbalization. For Stage 2, we directly parse the $True/False$ token to obtain y_{conf}.

5 Experimental Setup

5.1 LLM Selection

We selected three student models, Llama-3B (Meta-Llama-3.2-3B-Instruct), Mistral-7B (Mistral-7B-Instruct-v0.2), and Llama-70B (Meta-Llama-3-70B-Instruct), to span a range of contemporary instruction-tuned architectures and parameter scales. This selection covers small (3B), medium (7B), and large (70B) capacity models, enabling evaluation of Distill-RE under varying computational constraints. We pair these students with Mixtral-8x22B (Mixtral-8x22B-Instruct-v0.1) as an offline teacher model for chain-of-thought generation, explicitly choosing a teacher architecture distinct from the student families to reduce potential overlap in model inductive biases between teacher and student.

5.2 Datasets

We evaluate on three relation extraction benchmarks derived from the original TACRED (Zhang et al., 2017) (41 relation types, newswire and forum text), TACREV (Alt et al., 2020) (corrected TACRED label errors via expert re-annotation of the 5,000 most challenging dev and test examples), and Re-TACRED (Stoica et al., 2021) (refined relation definitions with 23.9% of examples relabeled for improved consistency). As shown in Table 1, all three datasets exhibit substantial "no relation" class imbalance, which highlights one of the core challenges we address in this work. For more details on the datasets, see Appendix A.2.

Dataset	Train	Dev	Test	no rel%
TACRED	68,124	22,631	15,509	78.6
TACREV	68,124	22,631	15,509	79.9
Re-TACRED	58,465	19,584	13,418	57.9

Table 1: Dataset statistics. no rel% refers to the proportion of "no relation" instances in the test set.

5.3 Few-shot Sampling

All experiments use a low-resource k-shot setting, where the training and validation sets are each constructed by randomly sampling k examples per relation across all relation types ($N = |\mathcal{R}|$). Final performance is reported on the full test set. We use the sampling code from Han et al. (2022a) and report results over 5 random seeds, following Gao et al. (2021).

5.4 Implementation Details

Compute All experiments were implemented in PyTorch (Paszke et al., 2019) using the HuggingFace Transformers library (Wolf et al., 2019). Student models were trained with supervised fine-tuning under a causal language modeling objective using a single NVIDIA H100 NVL GPU. Parameter-efficient fine-tuning was performed using LoRA adapters via the PEFT framework (Mangrulkar et al., 2022). All models were fine-tuned with LoRA adapters in bf16. Llama-3B and Mistral-7B used full bf16 precision, while Llama-70B used QLoRA with the base model loaded in 4-bit quantization to accommodate GPU memory constraints. LoRA was applied to all attention and feed-forward projection layers as well as the language modeling head. Teacher models were hosted with vLLM (Kwon et al., 2023) for efficient GPU inference. Mixtral-8x22B was used exclusively for offline CoT generation as a data augmentation step.

Hyperparameter Sweep We conducted a grid search over LoRA rank $\{16, 32, 64, 128\}$, scaling factor $\{0.25r, 0.5r, 0.75r\}$, learning rate $\{5 \times 10^{-4}, 5 \times 10^{-5}, 5 \times 10^{-6}\}$, and weight decay $\{0, 10^{-5}, 10^{-4}, 10^{-3}\}$. For Stage 2, we tested sampling 0, 1, 2 and 3 hard negatives per Stage 1 named-relation prediction, selecting 2 based on validation performance. To limit leakage, prompt development and hyperparameter tuning were completed on $k = 8$ and seed 21 Re-TACRED split. Resulting settings were applied across all datasets and experiments. We restricted experiments to configurations in which Stage 1 and Stage 2 used identical hyperparameter settings. Final hyperparameter selections are shown in Table 2.

Sequential Training & Checkpoint Selection All final experimental models were trained for 15 epochs. Training proceeded sequentially: the best validation checkpoint from Stage 1 was selected, and its outputs were used as inputs to Stage 2. The validation checkpoint with the highest validation micro-F_1 was used for each stage's final test evaluation.

Model	LR	WD	Rank	α	LoRA Params (M)
Llama-3B	5×10^{-5}	0.05	32	0.75r	52
Mistral-7B	1×10^{-4}	0	128	0.5r	340
Llama-70B	5×10^{-5}	0.05	64	0.5r	837

Table 2: Hyperparameter sweep results showing the optimal configuration per student model, selected by validation micro-F_1.

6 Results and Discussion

6.1 Relation Classification Performance

Table 3 shows that Distill-RE with the Llama-70B backbone achieves the strongest overall performance across all datasets and supervision levels, substantially outperforming all PLM-based and hybrid baselines. Aggregating across datasets, Distill-RE Llama-70B achieves mean micro-F_1 scores of 82.3 ± 1.2, 84.4 ± 2.3, and 84.9 ± 2.0 at the 8-, 16-, and 32-shot settings respectively, substantially outperforming the strongest baseline at each setting, with absolute improvements of $+32.4$, $+31.3$, and $+31.0$ points.

Notably, Distill-RE achieves these results without any synthetically generated relation instances. The blue rows in Table 3 report LLM-based data augmentation methods results, which improve existing RE baseline results by augmenting training data with synthetic sentence-relation pairs; a complementary setting not directly comparable to ours. The strongest such method, ODDA, outperforms Distill-RE by 0.4 and 3.2 micro-F_1 points at $k = 8$ and $k = 16$ respectively, averaged across TACRED and Re-TACRED[1], but does so using $8\times k$ additional synthetic samples per run. Whether incorporating synthetic data during Distill-RE training would further improve performance remains an open question, as neither code nor data has been released by these methods. **Within the purely few-shot setting, where only k real labeled examples are available, Distill-RE establishes a new state-of-the-art.** Additional results and discussion can be found in Appendix A.3.

6.2 Effect of Stage 2 Confirmation Model

To isolate the contribution of the confirmation stage, we compare Stage 1 (classification only)

[1]TACREV and $k = 32$ results not reported by the authors.

RE Method	Baseline RE Method	Backbone§	TACRED 8	TACRED 16	TACRED 32	TACREV 8	TACREV 16	TACREV 32	Re-TACRED 8	Re-TACRED 16	Re-TACRED 32
LLM-Based RE Methods											
Distill-RE		Llama-3B	75.6 ± 5.7	79.8 ± 3.4	77.2 ± 3.5	78.6 ± 7.1	79.7 ± 3.2	79.7 ± 5.8	77.9 ± 1.2	79.3 ± 1.0	80.6 ± 3.6
Distill-RE		Mistral-7B	78.8 ± 3.5	74.0 ± 4.1	76.6 ± 4.7	77.3 ± 2.1	79.6 ± 5.8	76.2 ± 3.6	79.2 ± 1.6	79.3 ± 1.9	79.6 ± 3.6
Distill-RE		Llama-70B	**81.0 ± 1.9**	**82.8 ± 2.4**	**83.1 ± 4.1**	**82.7 ± 1.2**	**87.0 ± 0.6**	**87.1 ± 0.6**	**83.3 ± 0.8**	**83.5 ± 0.9**	**84.4 ± 0.9**
Baselines: PLM-Based RE Models											
TYP Marker(Zhou and Chen, 2022)†		RoBERTa	*28.9*	*32.0*	*32.4*	*27.6*	*31.2*	*32.0*	*44.8*	*54.1*	*60.0*
PTR(Han et al., 2022b)†		RoBERTa	*28.1*	*30.7*	*32.1*	*28.7*	*31.4*	*32.4*	*51.5*	*56.3*	*62.1*
KnowPrompt(Chen et al., 2022b)†		RoBERTa	*32.0*	*35.4*	*36.5*	*32.1*	*33.1*	*43.7*	*55.3*	*63.3*	*65.0*
GenPT(Han et al., 2022a)		BART	*32.6 ± 0.5*	*35.6 ± 1.8*	*37.1 ± 1.0*	*31.1 ± 0.8*	*34.3 ± 1.9*	*36.6 ± 1.1*	*52.9 ± 1.5*	*58.1 ± 1.8*	*63.9 ± 1.5*
GenPT(Han et al., 2022a)		RoBERTa	*35.7 ± 1.1*	*36.6 ± 1.5*	*37.4 ± 1.8*	*34.4 ± 0.8*	*34.6 ± 1.6*	*36.2 ± 1.8*	*57.1 ± 2.3*	*60.4 ± 1.3*	*65.2 ± 2.0*
GenPT(Han et al., 2022a)		T5	*31.8 ± 0.8*	*33.3 ± 1.3*	*36.0 ± 1.1*	*31.3 ± 1.7*	*32.6 ± 1.5*	*34.6 ± 1.2*	*54.7 ± 2.3*	*58.7 ± 2.4*	*62.5 ± 1.0*
RetrievalRE(Chen et al., 2022a)∗		RoBERTa	*32.0*	*35.2*	*37.1*	*28.9*	*32.3*	*37.3*	*33.7*	*55.8*	*61.8*
PTRE(Zhang et al., 2024)		RoBERTa	*32.7*	*36.5*	*39.2*	*33.9*	*36.3*	*38.1*	*58.8*	*64.4*	*66.3*
Baselines: Hybrid LLM-PLM Models											
S^2ynRE(Xu et al., 2023a) (CP)		RoBERTa	*36.3*	*37.2*	*38.7*	*35.2*	*36.1*	*36.9*	*59.4*	*63.9*	*67.5*
DSARE(Liu et al., 2024)		RoBERTa	***43.8***	***45.4***	***45.9***	***44.7***	***46.6***	***46.9***	*60.0*	*66.8*	*67.1*
TKRE(Guo et al., 2025)	TYP Marker	RoBERTa	*36.7*	*38.3*	*39.1*	*37.2*	*38.2*	*39.0*	*58.7*	*63.8*	*65.5*
TKRE(Guo et al., 2025)	GenPT	RoBERTa	*39.6*	*42.1*	*43.2*	*40.2*	*41.3*	*42.9*	***61.3***	***67.6***	***68.9***
LLM Data Generation Methods											
Unleash(Xu et al., 2023b)	KnowPrompt	RoBERTa / GPT-3.5	*37.9*	*37.4*	-	***42.6***	***41.0***	-	*62.7*	*66.2*	-
ConsistRE(Zheng et al., 2024)‡	TYP Marker	RoBERTa / DeepSeek-v3	*71.8 ± 1.7*	*76.1 ± 1.5*	-	-	-	-	*72.6 ± 3.4*	*81.7 ± 2.5*	-
ConsistRE(Zheng et al., 2024)‡	KnowPrompt	RoBERTa / DeepSeek-v3	*75.8 ± 1.5*	*80.9 ± 0.8*	-	-	-	-	*80.1 ± 1.0*	*84.9 ± 0.6*	-
Diverse(Li et al., 2025) (mix-OBO)	RetrievalRE	RoBERTa Large / LLaMA2-7B-Chat	*33.8*	*35.8*	***37.5***	*33.8*	*34.1*	***36.9***	*58.6*	*62.6*	***65.2***
Diverse(Li et al., 2025) (mix-AAO)	RetrievalRE	RoBERTa Large / LLaMA2-7B-Chat	*34.9*	*36.6*	*37.2*	*35.1*	*35.6*	*36.8*	*55.4*	*61.1*	*64.8*
ODDA(Zhong et al., 2025)	TYP Marker	RoBERTa / DeepSeek-v3	*75.8 ± 1.4*	*80.5 ± 0.7*	-	-	-	-	*76.8 ± 0.9*	*84.9 ± 0.7*	-
ODDA(Zhong et al., 2025)	KnowPrompt	RoBERTa / DeepSeek-v3	***81.0 ± 1.0***	***84.8 ± 0.4***	-	-	-	-	***84.1 ± 0.9***	***87.9 ± 0.3***	-

Table 3: Distill-RE few-shot micro-F_1 results (mean ± std., higher is better) over five random splits on TACRED, TACREV, and Re-TACRED. Baseline results are reported from their original papers when available; otherwise as reported in Han et al. (2022a) (†) or Zhong et al. (2025) (‡). **Red** and **Blue** denote the best and second-best results across the Baseline and Distill-RE sections; **bold** marks the top result per section. Blue rows denote LLM Data Generation (DG) methods, which augment training data with LLM-synthesized samples and use existing methods for RE; these address a complementary setting and are thus not directly comparable. For LLM DG rows, Backbone is formatted as Baseline RE Backbone/DG LLM. Unreported results are marked "–".

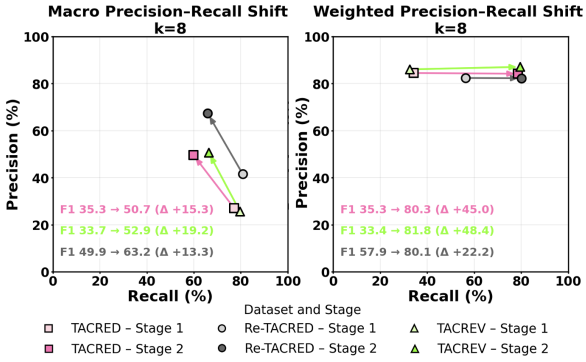


Figure 3: Precision-Recall shift from Stage 1 to Stage 2 of Distill-RE at $k = 8$, averaged across student LLM architectures. Left: macro-averaged; right: weighted-averaged. Arrows indicate the direction and magnitude of the shift, with ΔF_1 annotated per dataset.

Avg	k	Dataset	Stage 1 P	Stage 1 R	Stage 1 F_1	Stage 2 P	Stage 2 R	Stage 2 F_1
Macro	8	TACRED	27.1	77.1	35.3	49.6	59.8	50.7
		TACREV	25.7	79.6	33.7	50.7	66.3	52.9
		Re-TACRED	41.5	80.8	49.9	67.3	65.9	63.2
Weighted	8	TACRED	84.5	34.2	35.3	84.2	78.6	80.3
		TACREV	86.0	32.5	33.4	87.1	79.5	81.8
		Re-TACRED	82.4	56.3	57.9	82.3	80.1	80.1
Micro	8	TACRED		34.2			78.6	
		TACREV		32.5			79.5	
		Re-TACRED		56.3			80.1	

Table 4: Stage 1 and Stage 2 performance metrics at $k = 8$ averaged across seeds and backbone architectures.

results against their corresponding full two-stage pipeline results. As shown in Figure 3 and Table 4, Stage 1 achieves strong macro-average recall of 77.1%, 79.6%, and 80.8% on TACRED, TACREV, and Re-TACRED respectively at $k = 8$, averaged across three backbone architectures, indicating that the retrieval stage successfully identifies true named-relation instances. Yet Stage 1 weighted recall, dominated by the majority "no relation" class, is substantially lower at 34.2%, 32.5%, and 56.3%, revealing that Stage 1 over-predicts named-relations at the expense of "no relation" classification.

Stage 2 directly addresses the over-prediction problem by reassigning falsely predicted named-relations to the "no relation" class. The confirmation model recovers "no relation" classification while preserving named-relation discrimination, evidenced by weighted recall increases of +44.4, +47.0, and +23.8 percentage points across TACRED, TACREV, and Re-TACRED, respectively, as weighted precision remains stable. At the macro level, the precision-recall trade-off yields net macro-F_1 gains of +15.4, +19.2, and +13.3 points (Table 4). Additional error analysis containing the performance ceiling introduced by each stage is detailed in Appendix A.5.

The impact on micro-F_1 is also substantial, improving from $(34.2, 32.5, 56.3)$ to $(78.6, 79.5, 80.1)$ on TACRED, TACREV and Re-TACRED, respectively. The comparatively smaller gain on Re-TACRED may reflect its cleaner label space, which partially mitigates Stage 1 over-prediction, in combination with its smaller "no relation" percentage, reducing the potential room for improvement by Stage 2. **Taken together, these results validate the two-stage design: named-relation retrieval followed by targeted True/False confirmation effectively balances recall across relation classes with accurate "no relation" rejection.** Consistent trends at $k = 16$ and $k = 32$ are reported in Appendix A.4.

6.3 Effect of CoT Supervision

To isolate the contribution of incorporating teacher CoT into supervised training, we compare results from matched split-and-seed runs trained with versus without CoT supervision (i.e., gold-label fine-tuning alone). **Our results show that CoT-augmented supervision improves Distill-RE performance across backbone models, shot counts, and datasets, yielding mean gains of $+7.6$ micro-F_1 and $+9.1$ macro-F_1 points across 135 runs.** Only 11.1% of runs show degradation, with mean losses of -2.7 micro-F_1 and -1.8 macro-F_1 points; substantially smaller than typical gains.

Dataset label quality unsurprisingly moderates the effectiveness of CoT-based supervision, as inaccurate reasoning patterns are explicitly distilled into student models. Re-TACRED average improvements are largest and monotonically increase with k, suggesting CoT reasoning chains provide reliable signal under relatively clean supervision. With TACRED and TACREV, gains are relatively more modest and exhibit higher variance, consistent with label noise disrupting CoT-based learning. Model capacity further influences robustness to label noise, as evidenced by Llama-70B maintaining the strongest performance gains with the lowest variance across all settings even in the presence of mislabeled data, while the smaller models (Llama-3B, Mistral-7B) show non-monotonic behavior on noisy datasets. An ablation comparing CoT supervision with No-CoT supervision is detailed in Appendix A.7.

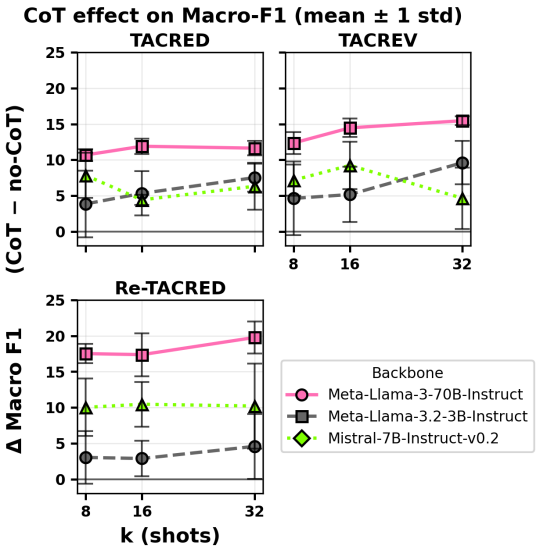


Figure 4: Macro-F_1 comparison (CoT − No CoT) between Distill-RE runs with identical k-shot values, backbone model, and random seed across each tested dataset.

	Stage 1		Stage 2	
Model	Train	Infer	Train	Infer
Llama-3B	0.1	3.0	0.1	3.6
Mistral-7B	0.3	4.3	0.1	5.0
Llama-70B	1.5	55.3	1.1	40.1

Table 5: Runtime on Re-TACRED (seconds per sample)

6.4 Runtime Analysis

Table 5 reports training and inference latency in seconds per sample. Training latency is computed by measuring total fine-tuning time and normalizing by epochs and the number of training samples, using the $k = 8$ setting on Re-TACRED with a maximum sequence length of 1000 tokens (prompt + output). Because TACRED, TACREV, and Re-TACRED share the same sentences and the metric is normalized per sample, latency is expected to be similar across other k values and datasets. Inference latency is averaged over 500 samples using autoregressive generation (`max_new_tokens=500`), which requires multiple forward passes and therefore results in higher latency than training. Runtime increases with model size, with Llama-70B substantially slower than the 3B and 7B models. The increases in model size and runtime reflect the performance-latency trade-off, which practitioners should consider when selecting a configuration for their use case.

7 Conclusion and Future Work

We introduced Distill-RE, a two-stage distillation framework that transfers chain-of-thought reasoning from a teacher LLM to student models for few-shot relation extraction. By decomposing prediction into classification and confirmation, the framework directly addresses over-prediction bias, a central failure mode in RE, and achieves state-of-the-art few-shot performance across TACRED, TACREV, and Re-TACRED among methods that do not leverage LLM-generated relation samples. Our results highlight two key insights for NLP practitioners:

1. Decoupling recall and precision objectives through a two-stage design leads to substantial and consistent gains, particularly on datasets with severe class imbalance.
2. Reasoning-augmented supervision, rather than labels alone, provides a sample-efficient training signal, enabling strong generalization even in extremely low-resource settings.

For future work, several directions could further strengthen this approach. Integrating Distill-RE with LLM-based data augmentation may provide complementary gains by combining improved supervision with increased data diversity. The quality of teacher-generated reasoning is also a critical factor: exploring stronger teachers, model ensembles, or filtering mechanisms could yield more reliable supervision signals. Finally, extending Distill-RE beyond closed, sentence-level RE to more realistic settings, such as open relation extraction or document-level reasoning, would test the generality of the framework and its ability to scale to more complex information extraction tasks.

Limitations

One of the primary drivers of Distill-RE's few-shot success is also the source of its main limitation, the computationally intensive backbone. LLMs require substantial GPU resources for both training and inference, with costs considerably greater than smaller PLMs, a burden compounded by our finding that performance scales with model size. Distill-RE also depends on a capable teacher model for generating informative relation label explanations. We do not characterize the effect of teacher model quality on overall Distill-RE performance, leaving the teacher model's effect an open question.

Additionally, our algorithm depends on a well-documented dataset schema with clearly defined relations and entity types. We use relation definitions directly in our classification and confirmation prompts, and filter candidate relations based on entity types, meaning incomplete or inconsistent metadata may compromise prompt quality and limit applicability.

Finally, model performance is sensitive to training data quality, as evidenced by our strongest results on Re-TACRED (the cleanest TACRED variant) and weakest on the original, noisiest TACRED. While the few-shot paradigm reduces dependence on large labeled datasets, accurate relation labels remain a key component of our approach. Additionally, our evaluation is limited to TACRED family datasets, which broadly share the same underlying sample sentences and relation schema. Generalization to other domains, relation ontologies and writing styles remains unexamined.

Ethical Considerations

Compared to prior relation extraction approaches, our methodology places particular emphasis on the reasoning capabilities of large language models. However, the pre-training corpora of LLMs are derived predominantly from internet-crawled, human-generated content, and consequently encode the biases of the individuals and societies that produced them. Furthermore, access to the internet is not equally distributed across demographic groups, thus these corpora disproportionately reflect the perspectives of wealthier, Western, and English-speaking populations; referred to by Bender et al. (Bender et al., 2021) as a "hegemonic viewpoint." The biases associated with this viewpoint can produce systematic performance disparities across languages, domains, and perceived gender, sexual identity, race, or ethnicity of the entities whose relations are being modeled (Navigli et al., 2023). This concern is compounded by the fact that relation extraction is commonly used to populate knowledge bases that ground downstream applications, meaning that unmitigated biases will propagate through the entire pipeline. One downstream application worth highlighting specifically is surveillance and monitoring. Relation extraction systems can be used to extract and aggregate structured information about individuals from unstructured text without the knowledge or consent of those being monitored (Brayne, 2017), raising serious concerns regarding the potential for harm caused by a biased upstream RE system.

References

Mistral AI. 2024. Cheaper, better, faster, stronger. https://mistral.ai/news/mixtral-8x22b. Accessed: 2026-02-23.

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020. Tacred revisited: A thorough evaluation of the tacred relation extraction task. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 1558–1569.

Hyeong-Ryeol Baek and Yong-Suk Choi. 2022. Enhancing targeted minority class prediction in sentence-level relation extraction. *Sensors*, 22(13):4911.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Sarah Brayne. 2017. Big data surveillance: The case of policing. *American sociological review*, 82(5):977–1008.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. Rebel: Relation extraction by end-to-end language generation. In *Findings of the association for computational linguistics: EMNLP 2021*, pages 2370–2381.

Xiang Chen, Lei Li, Ningyu Zhang, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022a. Relation extraction as open-book examination: Retrieval-enhanced prompt tuning. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 2443–2448.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022b. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web conference 2022*, pages 2778–2788.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Sefika Efeoglu and Adrian Paschke. 2024. Relation extraction with fine-tuned large language models in retrieval augmented generation frameworks. *arXiv preprint arXiv:2406.14745*.

Joe Ellis, Jeremy Getman, Heather Simpson, Kira Griffitt, Hoa Trang Dang, Ralph Grishman, Heng Ji, Catherine DePrince, Thomas Riese, and Neil Kuster. 2015. TAC KBP 2015 Slot Descriptions. Technical report, Linguistic Data Consortium. Unpublished.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830.

Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2019. Fewrel 2.0: Towards more challenging few-shot relation classification. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 6250–6255.

Quanjiang Guo, Jinchuan Zhang, Sijie Wang, Ling Tian, Zhao Kang, Bin Yan, and Weidong Xiao. 2025. Bridging generative and discriminative learning: Few-shot relation extraction via two-stage knowledge-guided pre-training. *arXiv preprint arXiv:2505.12236*.

Jiale Han, Shuai Zhao, Bo Cheng, Shengkun Ma, and Wei Lu. 2022a. Generative prompt tuning for relation classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3170–3185.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022b. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 4803–4809.

Tingting Hang, Wei Wu, Jun Feng, Hamza Djigal, and Jun Huang. 2025. A survey of few-shot relation extraction combining meta-learning with prompt learning. *Neurocomputing*, 647:130534.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2023. Large language models are reasoning teachers. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 14852–14882.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Zexuan Li, Hongliang Dai, and Piji Li. 2025. Generating diverse training samples for relation extraction with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 713–726.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Ye Liu, Kai Zhang, Aoran Gan, Linan Yue, Feng Hu, Qi Liu, and Enhong Chen. 2024. Empowering few-shot relation extraction with the integration of traditional re methods and large language models. In *International Conference on Database Systems for Advanced Applications*, pages 349–359. Springer.

Xilai Ma, Jing Li, and Min Zhang. 2023a. Chain of thought with explicit evidence reasoning for few-shot relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2334–2352.

Yubo Ma, Yixin Cao, Yong Hong, and Aixin Sun. 2023b. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In *Findings of the association for computational linguistics: EMNLP 2023*, pages 10572–10601.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, Benjamin Bossan, and Marian Tietz. 2022. PEFT: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Roberto Navigli, Simone Conia, and Björn Ross. 2023. Biases in large language models: origins, inventory, and discussion. *ACM Journal of Data and Information Quality*, 15(2):1–21.

Tapas Nayak, Navonil Majumder, Pawan Goyal, and Soujanya Poria. 2021. Deep neural approaches to relation triplets extraction: a comprehensive survey. *Cognitive computation*, 13(5):1215–1232.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Yongbin Qin, Weizhe Yang, Kai Wang, Ruizhang Huang, Feng Tian, Shaolin Ao, and Yanping Chen. 2021. Entity relation extraction based on entity indicators. *Symmetry*, 13(4):539.

George Stoica, Emmanouil Antonios Platanios, and Barnabás Póczos. 2021. Re-tacred: Addressing shortcomings of the tacred dataset. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13843–13850.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. Gpt-re: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 3534–3547.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pages 5085–5109.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Benfeng Xu, Quan Wang, Yajuan Lyu, Dai Dai, Yongdong Zhang, and Zhendong Mao. 2023a. S2ynre: Two-stage self-training with synthetic data for low-resource relation extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8186–8207.

Xin Xu, Yuqi Zhu, Xiaohan Wang, and Ningyu Zhang. 2023b. How to unleash the power of large language models for few-shot relation extraction? In *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pages 190–200.

Hangxiao Yang, Changpu Chen, Shaokai Zhang, Baiyang Chen, Chang Liu, and Qilin Li. 2024. Etcgn: entity type-constrained graph networks for document-level relation extraction. *International Journal of Machine Learning and Cybernetics*, 15(12):5949–5962.

Kai Zhang, Bernal Jiménez Gutiérrez, and Yu Su. 2023. Aligning instruction tasks unlocks large language models as zero-shot relation extractors. In *Findings of the association for computational linguistics: ACL 2023*, pages 794–812.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 35–45.

Zirui Zhang, Yiyu Yang, and Benhui Chen. 2024. Prompt tuning for few-shot relation extraction via modeling global and local graphs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13233–13242.

Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. 2024. A comprehensive survey on relation extraction: Recent advances and new frontiers. *ACM Computing Surveys*, 56(11):1–39.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. Pmlr.

Yifan Zheng, Wenjun Ke, Qi Liu, Yuting Yang, Ruizhuo Zhao, Dacheng Feng, Jianwei Zhang, and Zhi Fang. 2024. Making llms as fine-grained relation extraction data augmentor. In *IJCAI*, pages 6660–6668.

Yijie Zhong, Yunfan Gao, Xiaolian Zhang, and Haofen Wang. 2025. Odda: An ooda-driven diverse data augmentation framework for low-resource relation extraction. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 267–285.

Wenxuan Zhou and Muhao Chen. 2022. An improved baseline for sentence-level relation extraction. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 161–168.

A Appendix

A.1 Prompt Templates

This section presents all prompts used in the Distill-RE pipeline. Prompts labeled "Teacher" were used to generate teacher model CoT augmentations, while those labeled "Student" were used to prompt student models during fine-tuning and inference. Variable prompt components are denoted by curly brackets { } containing descriptive variable names indicating which text will be inserted into each slot. The rules to fill these variable slots and the role the prompts fill in the Distill-RE pipeline are detailed in Section 4. Variables used across the prompts are defined as follows:

- $sentence$: the sample sentence from which the relation triple is extracted
- $head\_entity$, $tail\_entity$: the head (subject) and tail (object) entities of interest contained in $sentence$
- $head\_entity\_type$, $tail\_entity\_type$: the head and tail entity types (e.g., person, city)
- $true\_relation$: the ground-truth relation label verbalization as defined in (Han et al., 2022a)
- $definition$: a manually written relation definition derived primarily from (Ellis et al., 2015; Stoica et al., 2021)
- $hypothesis$: the relation definition rewritten as a True/False statement, with $head\_entity$ and $tail\_entity$ inserted to ground it in the sample sentence
- $confirmation\_boolean$: a Boolean indicating whether a sample's corresponding $hypothesis$ is correct
- $valid\ relation\ and\ definitions$: enumerated list of all valid relation classes ($\mathcal{R}_{valid}$) and their corresponding definitions based on $head\_entity\_type$ and $tail\_entity\_type$
- $relation\ verbalization\ list$: enumerated list of all valid relation classes ($\mathcal{R}_{valid}$) based on $head\_entity\_type$ and $tail\_entity\_type$
- $cot\_classification$: LLM-generated CoT explanation for relation classification
- $cot\_confirmation$: LLM-generated CoT explanation for True/False relation confirmation labels

A.2 Dataset Details

TACRED The TAC Relation Extraction Dataset (Zhang et al., 2017) is a widely used large-scale dataset for sentence-level relation extraction. It contains sentences from both newswire and discussion forums, and includes 41 relation types. The dataset relations were crowd sourced using Mechanical Turk.

TACREV TACREV (Alt et al., 2020) is a revised version of TACRED that addresses substantial label errors in the original crowdsourced annotations. The authors identified the 5,000 most challenging examples from the development and test sets based on disagreement across 49 relation extraction models, then reannotated them using trained linguists

```
Using the relationship definition provided, explain step-by-step why the relationship between the
head entity ‘{head_entity}’ and the tail entity ‘{tail_entity}’ is classified as ‘{true_relation}’.
The definition states that ‘{true_relation}’ refers to: {definition}.

Please base your explanation solely on the evidence provided in the sentence: "{sentence}"

List each piece of evidence from the sentence that supports this classification and explain how it
relates to the relationship definition. Make sure your explanation is clear, logical, and avoids
any assumptions not supported by the sentence.
```

Figure 5: Teacher Stage 1 Classification Prompt. Zero-shot prompt used by the teacher model to generate CoT explanations for fine-tuning the Stage 1 classification student model.

```
sentence: ‘{sentence}’
statement: ‘{hypothesis}’

Using only the text of the sentence provided above as evidence, provide a step by step explanation
of why the given statement is {confirmation_boolean}. We know it is {confirmation_boolean} with
absolute certainty.
```

Figure 6: Teacher Stage 2 Confirmation Prompt. Zero-shot prompt used by the teacher model to generate CoT explanations for fine-tuning the Stage 2 confirmation student model.

```
sentence: ‘{sentence}’
statement: ‘{hypothesis}’

You are given a sentence and a statement. Your task is to decide whether the statement can be
reasonably inferred from the sentence.

Answer True if the sentence clearly supports or entails the statement. Answer False if the sentence
does not support the statement, contradicts it, or leaves it uncertain. Begin your answer with True
or False, followed by a step by step explanation based only on the sentence.

{confirmation_boolean}. {cot_confirmation}
```

Figure 7: Student Stage 2 Confirmation Prompt. Student prompt to generate Stage 2 relation confirmation. During inference, the prompt ends after “explanation based only on the sentence.”

```
{valid relation and definitions}
Instructions: Based strictly on the context within the sentence and the definitions of each
relationship type, determine if there is a direct and explicit relationship between the "head
entity" {head_entity} and the "tail entity" {tail_entity}. Choose a relation type only if clear
evidence in the sentence aligns with one of the specified definitions. If no evidence directly
supports a defined relationship, select "no relation."
Sentence: {sentence}
Head entity: {head_entity}
Head entity type: {head_entity_type}
Tail entity: {tail_entity}
Tail entity type: {tail_entity_type}
Relation types: {relation verbalization list}
Question: From the list of possible relation types, which is the relation type between the "head
entity" {head_entity} and the "tail entity" {tail_entity}? Ensure that the choice matches the
definitions and has explicit evidence from the sentence.
Format your prediction as follows: [REASONING] chain of thought reasoning here [RELATION] predicted
relation type here.
### relation type prediction: [REASONING] {cot_classification} [RELATION] {true_relation}
```

Figure 8: Student Stage 1 Classification Prompt. Student prompt to generate Stage 1 relation classification. During inference, the prompt ends after ‘### relation type prediction: ’

"members": Organizations or Geopolitical entities that are members of the head entity organization.
"siblings": The brothers and sisters of the head entity person. Full siblings, step-siblings and half-siblings are acceptable answers. Brothers- or sisters-in-law are not acceptable responses (they are for other family).
"spouse": The spouse(s) of the head entity person. Former spouses are acceptable answers.
"country of branch": The country of an organization's branch or office. The tail entity must be the name of a country.
"country of death": The country in which the head entity person died. The tail entity must be the name of a country.
"parents": The parents of the head entity person. In addition to biological parents, stepparents and adoptive parents are also acceptable answers.
"state or provinces of residence": Geopolitical entities at the state or province level in which the head entity person has lived, must be filled with state or province names. Residence must be lexically supported in source documents.
"top members or employees": The persons in high-level, leading positions at the head entity organization. Top Member/Employee positions should imply a level of decision-making authority over the entire head entity organization.
"dissolved": The date on which the head entity organization was dissolved. When companies merge to form a new entity, the original companies should be considered as dissolved.
"number of employees or members": The total number of people who are employed by or have membership in an organization.
"state or province of death": The geopolitical entity at state or province level in which the head entity person died. The tail entity must be a state or province name.
"origin": The nationality and/or ethnicity of the head entity person.
"children": The children of the head entity person, including adopted and step-children.
"political or religious affiliation": Ideological groups with which the organization is associated.
"city of birth": The geopolitical entity at the municipality level (city, town, or village) in which the head entity person was born. Mentions of 'native' or related synonyms indicate birth location. The tail entity must be the name of a city, town, or village.
"title": Official or unofficial name(s) of the employment or membership positions that have been held by the head entity person.
"shareholders": The head entity organization, person, or geopolitical entity has documented evidence of stock holdings or stock transactions involving the tail entity.
"employee of": The organizations or geopolitical entities (governments) by which the head entity person has been employed.
"member of": Organizations or geopolitical entities of which the head entity organization is a member itself.
"founded by": The person, organization, or geopolitical entity that founded the head entity organization.
"countries of residence": Countries in which the head entity person has lived.
"other family": Entities share an explicit familial relationship, limited to brother-in-law, sister-in-law, cousin, uncle, aunt, grandchild, grandparent.
"religion": The religion to which the head entity person has belonged. Former religions are acceptable answers.
"identity": Both entities refer to the same person, including cases where a pronoun is used.
"date of birth": The date on which the head entity person was born.
"city of branch": The geopolitical entity at the level of city, town, village in which a branch or office is located.
"alternate names": Any name used to refer to the head entity organization that is distinct from the "official" name. Alternate names may include former names, aliases, alternate spellings, acronyms, abbreviations, translations or transliterations of names, and any official designators such as stock ticker code or airline call sign.
"website": An official top level URL for the organization's website.
"cause of death": The explicit cause of death for the head entity person.
"state or province of branch": The geopolitical entity at state or province level where a branch of an organization is located. The tail entity must be a state or province name.
"schools attended": Any school (college, high school, university, etc.) that the head entity person has attended.
"country of birth": The country in which the head entity person was born. Mentions of 'native' or related synonyms indicate birth location. The tail entity must be the name of a country.
"date of death": The date of the head entity person's death.
"city of death": The geopolitical entity at the level of city, town, village in which the head entity person died. The tail entity must be a city, town, or village name.
"founded": The date on which the head entity organization was founded.
"cities of residence": Geopolitical entities at the level of city, town, or village in which the head entity person has lived. The tail entity must be the name of a city, town, or village.
"age": A reported age of the head entity person at a moment in time.
"charges": The charges or crimes (alleged or convicted) of the head entity person.
"state or province of birth": The geopolitical entity at state or province level in which the head entity person was born. Mentions of 'native' or related synonyms indicate birth location. The tail entity must be the name of a state or province.
"no relation": No clear evidence of any of the other listed relationships between the entities.

Figure 9: List of All Relation Class Definitions. Definitions used in classification CoT generation and Stage 1 classification model prompting.

"members": The geopolitical entity or organization tail_entity is a member of the organization head_entity.
"siblings": The individuals head_entity and tail_entity are siblings (this includes full, step-, and half-siblings, but not in-laws).
"spouse": head_entity is the spouse of tail_entity. Former spouses are valid.
"country of branch": The organization head_entity has a branch or office in the country tail_entity. tail_entity must be the name of a country.
"country of death": The person head_entity died in the country tail_entity. tail_entity must be the name of a country.
"parents": The individual tail_entity is a parent of head_entity (this includes biological, step-, or adoptive parents).
"state or provinces of residence": The person head_entity has lived at the state or province tail_entity. tail_entity is at the state or province level.
"top members or employees": The persons tail_entity is in a high-level, leading position at the organization head_entity. tail_entity has a level of decision-making authority over the organization head_entity.
"dissolved": The organization head_entity was dissolved on the date tail_entity. When companies merge to form a new entity, the original companies should be considered as dissolved.
"number of employees or members": head_entity organization has tail_entity number of employees or members.
"state or province of death": head_entity died in the state or province tail_entity. tail_entity is at the state or province level.
"origin": head_entity has nationality or ethnicity of tail_entity.
"children": tail_entity is a child (biological, adopted, or step-child) of head_entity.
"political or religious affiliation": head_entity is affiliated with the political or religious group tail_entity.
"city of birth": head_entity was born in the city of tail_entity. tail_entity is at the city level.
"title": head_entity has held the title or position of tail_entity.
"shareholders": head_entity has documented evidence of stock holdings or stock transactions involving tail_entity.
"employee of": head_entity was employed by the organization or geopolitical entity tail_entity.
"member of": Organization head_entity is a member of the organization or geopolitical entity tail_entity.
"founded by": head_entity was founded by tail_entity.
"countries of residence": head_entity has lived in the country tail_entity. tail_entity must be the name of a country.
"other family": head_entity is the brother-in-law, sister-in-law, cousin, uncle, aunt, grandparent, grandson, or granddaughter of tail_entity.
"religion": head_entity belonged to the religion tail_entity.
"identity": head_entity and tail_entity refer to the same person.
"date of birth": head_entity was born on the date tail_entity.
"city of branch": head_entity has a branch or office located in the city tail_entity. tail_entity is at the city level.
"alternate names": tail_entity is a name used to refer to the head_entity that is distinct from the "official" name. This may include former names, aliases, alternate spellings, acronyms, abbreviations, translations or transliterations of names, and any official designators such as stock ticker code or airline call sign.
"website": tail_entity is the official website of the organization head_entity.
"cause of death": head_entity died of the cause of death tail_entity.
"state or province of branch": head_entity has a branch or office located in the state or province tail_entity. tail_entity is at the state or province level.
"schools attended": head_entity attended the school tail_entity.
"country of birth": head_entity was born in the country tail_entity. tail_entity must be the name of a country.
"date of death": head_entity died on the date tail_entity.
"city of death": head_entity died in the city of tail_entity. tail_entity is at the city level.
"founded": The organization head_entity was founded on the date tail_entity.
"cities of residence": head_entity lived in the city tail_entity.
"age": head_entity is the age of tail_entity.
"charges": head_entity was charged with or convicted of the crime tail_entity.
"state or province of birth": head_entity was born in the state or province tail_entity. tail_entity is at the state or province level.

Figure 10: List of All Relation Class True/False Statements. Statements used in confirmation CoT generation and Stage 2 confirmation model prompting. When used in prompts, $head\_entity$ and $tail\_entity$ are replaced with their sample values.

following a rigorous protocol. This process relabeled approximately 50% of the challenging examples while maintaining TACRED's original relation schema and structure.

Re-TACRED Re-TACRED (Stoica et al., 2021) is a reannotated version of the TACRED dataset, created for improved relation label accuracy. The authors refined ambiguous relation definitions to reduce annotator confusion and improve label consistency. The relation labeling process featured an improved crowd-sourced annotation strategy and relabeled 23.9% of the original TACRED dataset.

A.3 Relation Classification Performance

Figure 11 shows macro and weighted performance metrics across all experiments, aggregated by backbone, k, and dataset configuration. Overlapping Stage 1 and Stage 2 bars highlight the change in performance between stages. The bar for the stage with the smaller value is plotted on top of the bar with the larger value for visibility.

In the first row, there are very minor performance gaps between Stage 1 and Stage 2 weighted-precision results, reflecting the accuracy of the Stage 2 confirmation model, with only a small set of named-relation samples being incorrectly reassigned to "no relation." Notably, the two-stage design maintains strong weighted-precision, while substantially improving weighted-recall, showing the model accurately reassigns the over-predicted "no relation" samples. This combination of a maintenance in strong precision and improved recall results in a large increase in weighted-F_1 across all experimental configurations.

The second row shows macro-averaged performance metrics. Stage 2 reduces false positives present across classes on average, consistently improving macro-precision. However, macro-recall drops across all configurations as the confirmation stage reassigns some correctly predicted named-relations from Stage 1 to the "no relation" class. Despite this, the improvement in precision outweighs the reduction in recall as macro-F_1 improved across all experimental settings.

We illustrate the Stage 1 to Stage 2 macro and weighted metric trade-offs in Figure 13, aggregated over student-LLM architectures to highlight the consistency of the trend across k values and datasets (Section 6.2). Figure 12 shows consistent improvements in micro-F_1 from Stage 1 to Stage 2 across all experimental settings. This gain is driven by the accurate reassignment of over-predicted samples to the correct "no relation" class via the Stage 2 confirmation model, as supported by the maintenance of weighted precision between stages, while correctly predicted named-relation samples remain largely preserved, reflected by the minor decrease in macro-recall between stages. The magnitude of the improvement between stages is largely driven by the prevalence of "no relation" samples in each dataset and the accuracy of the Stage 2 confirmation model.

A.4 Effect of Stage 2 Confirmation Model Across Few-Shot Settings

			Stage 1			Stage 2		
Avg	k	Dataset	P	R	F_1	P	R	F_1
Macro	8	TACRED	27.1	77.1	35.3	49.6	59.8	50.7
		TACREV	25.7	79.6	33.7	50.7	66.3	52.9
		Re-TACRED	41.5	80.8	49.9	67.3	65.9	63.2
Weighted	8	TACRED	84.5	34.2	35.3	84.2	78.6	80.3
		TACREV	86.0	32.5	33.4	87.1	79.5	81.8
		Re-TACRED	82.4	56.3	57.9	82.3	80.1	80.1
Micro	8	TACRED		34.2			78.6	
		TACREV		32.5			79.5	
		Re-TACRED		56.3			80.1	
Macro	16	TACRED	26.9	78.7	35.6	50.3	63.0	52.2
		TACREV	26.4	81.7	34.9	51.5	69.6	55.3
		Re-TACRED	42.9	83.1	51.7	66.6	69.7	65.0
Weighted	16	TACRED	84.6	33.6	34.4	84.9	79.1	80.9
		TACREV	86.6	34.6	36.7	87.8	82.1	83.9
		Re-TACRED	83.5	59.9	62.4	83.0	80.7	80.9
Micro	16	TACRED		33.6			79.1	
		TACREV		34.6			82.1	
		Re-TACRED		59.9			80.7	
Macro	32	TACRED	27.3	79.8	36.4	49.7	65.1	53.1
		TACREV	27.5	83.2	36.7	51.0	72.2	55.8
		Re-TACRED	44.3	85.6	54.1	67.1	73.2	67.0
Weighted	32	TACRED	85.1	36.3	38.4	85.4	79.0	80.9
		TACREV	87.2	38.5	41.5	88.4	81.0	83.3
		Re-TACRED	84.3	63.2	65.4	84.1	81.5	81.8
Micro	32	TACRED		36.3			79.0	
		TACREV		38.5			81.0	
		Re-TACRED		63.2			81.5	

Table 6: Stage 1 and Stage 2 performance metrics at $k \in \{8, 16, 32\}$ averaged across seeds and backbone architectures.

As shown in Figure 13 and Table 6, Stage 1 achieves strong macro-averaged recall across each dataset and k setting. This indicates that the retrieval stage successfully identifies named-relation instances across data regimes. However, Stage 1

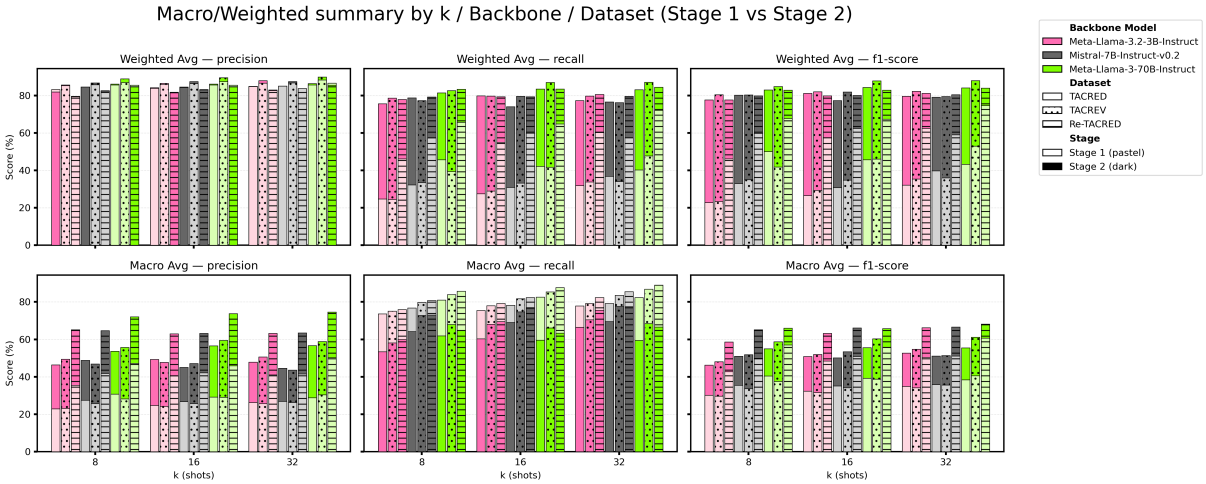


Figure 11: Macro and weighted precision, recall and F_1 across backbone models, k settings, and datasets. Stage 1 and Stage 2 results are shown as overlapping bars to highlight the performance delta between stages. Datasets are distinguished by bar pattern, backbone models by color and stage by color saturation (Stage 1, pastel colors and Stage 2, saturated colors). Results reflect performance averaged over 5 random seeds. Bars smaller in value are plotted on top of bars larger in value.

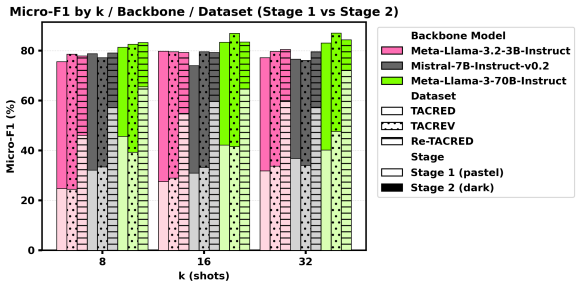


Figure 12: Micro-F_1 across backbone models, k settings, and datasets

weighted-recall, dominated by the majority "no relation" class, substantially underperforms macro-averaged recall across all k settings and datasets. This reveals that Stage 1 over-predicts named relations at the expense of "no relation" class across each data setting. Additional data availability results in Stage 1 improvement gains, with mean macro-recall improvements from 79.2 at $k = 8$ to 81.2 at $k = 16$ and 82.9 at $k = 32$. Micro-recall improves similarly from 41.0 at $k = 8$ to 42.7 at $k = 16$ and 46.0 at $k = 32$, showing the additional data helps improve named-relation recall while also helping mitigate the over-prediction problem but does not fully resolve it.

Stage 2 effectively addresses the over-prediction problem across the data regimes tested, evidenced by average weighted recall increases of $+38.4$, $+37.9$, and $+34.5$ for $k = 8, 16, 32$ respectively. At the macro level, the precision-recall trade-off yields net macro-F_1 gains of $+16.0$, $+16.8$, and $+16.2$ at $k = 8, 16, 32$. Additional data improves macro-F_1 primarily in Stage 1, with Stage 2 performance more similar across k values.

The impact on micro-F_1 is most striking: improving from 41.0 to 79.4 at $k = 8$, 42.7 to 80.6 at $k = 16$, and 46.0 to 80.5 at $k = 32$ when averaged across datasets and backbones. Taken together, these results validate the two-stage design: named-relation retrieval followed by targeted $True/False$ confirmation effectively balances recall across relation classes with accurate "no relation" rejection.

A.5 Error Analysis

We characterize the performance ceiling introduced by each stage of our pipeline shown in Tables 7, 8, 9. Stage 1 is designed to achieve high named-relation recall; "no relation" samples that are predicted as a named-relation can then be corrected by the Stage 2 confirmation model. Stage 2, in turn, is designed to correct Stage 1's named-relation over-prediction, reassigning incorrectly predicted named-relation samples to the "no relation" class. Consequently, there are two types of Stage 1 errors that are not recoverable by Stage 2: 1) named-relations incorrectly predicted as "no relation," and 2) confusion between named-relation classes (e.g., predicting "founded_by" when the gold label is "employed_by"). Stage 2 can only reassign a prediction to "no relation," it cannot reassign it to a different named-relation class. For each dataset,

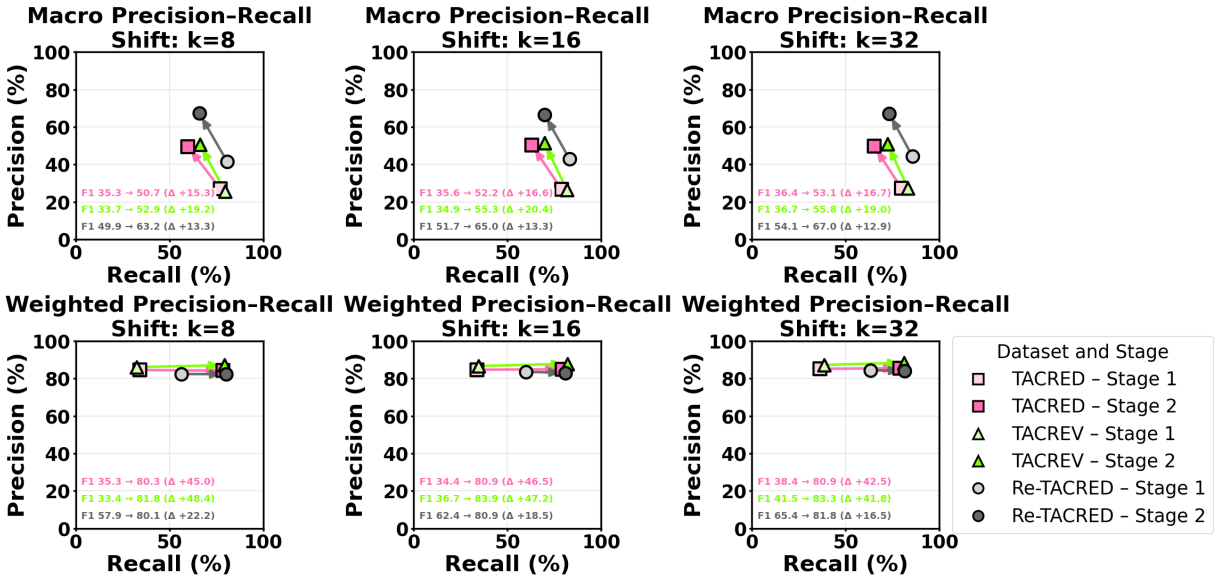


Figure 13: Precision-Recall shift from Stage 1 (Classification) to Stage 2 (Confirmation) of Distill-RE at k=8, 16, 32, averaged across student LLM architectures. Top row: macro-averaged; Bottom row: weighted-averaged. Arrows indicate the direction and magnitude of the shift, with Δ F_1 annotated per dataset.

we report:

- Stage 1 Micro-F_1: Micro-F_1 after Stage 1 alone
- Ceiling Micro-F_1: The upper-bound Micro-F_1 achievable if Stage 2 corrected every recoverable error perfectly
- Actual Micro-F_1 S1+2: The Micro-F_1 achieved by the full two-stage pipeline
- Max Micro-F_1 Gain: The maximum possible improvement from Stage 1 to Stage 2 (Ceiling Micro-F_1 − Stage 1 Micro-F_1)
- % Ceiling Captured: The share of that maximum Micro-F_1 gain actually realized by Stage 2, i.e. (Actual S1+2 − Stage 1 Micro-F_1) / max gain

We complete this analysis for each few-shot setting, $k = 8, 16, 32$ training samples per relation class. Results are averaged across random seed and backbone model configurations. In each experimental setting, Ceiling Micro-F_1 is much higher than Actual S1+2 (e.g., at $k = 8$, TACREV with CoT reaches a 97.1% ceiling but only 79.5% actual accuracy, capturing 72.8% of the possible gain). This gap holds across all datasets, CoT settings, and values of k, indicating substantial room to improve the confirmation model independent of how much training data Stage 1 has.

Comparing across k, Actual S1+2 Micro-F_1 is largely stable or improves as k grows, particularly with CoT: Re-TACRED with CoT rises from 80.1 to 81.5 as k increases from 8 to 32, and TACREV with CoT peaks at 82.1 at $k = 16$. Without CoT, Actual S1+2 is flat to mildly declining across all three datasets (e.g., TACRED: 73.6 → 74.1 → 72.5; Re-TACRED: 70.1 → 69.8 → 69.2).

A.6 Stage 1 Named-Relation Performance

This table reports Stage 1 performance on named-relation classification. Stage 1 is especially important because Stage 2 can only retain the relation predicted by Stage 1 or reject it as "no relation"; it cannot recover a named relation that Stage 1 initially misses. Accordingly, named relations must be identified correctly during the first stage of the pipeline. With CoT prompting, the miss rate, the percentage of gold named-relation instances incorrectly classified as "no relation", remains very low across all datasets and values of k, ranging from 0.3% for TACREV at $k = 32$ to 1.7% for Re-TACRED at $k = 8$. Confusion among named-relation labels also generally decreases as k increases, suggesting that additional demonstrations help the model distinguish between semantically similar relations. Averaged across datasets, backbone models, seeds, and values of k, CoT increases the correct classification rate from approximately 68.3% to 87.3%. Most of this 19.0 percentage-point improvement results from the near elimination of named-relation misses, whose average rate decreases from approximately 17.8% without CoT to 1.1% with CoT. The average confusion rate also decreases, from approximately 14.0% to 11.6%, indicating that CoT pri-

Dataset	CoT Training	Stage 1 Accuracy	Ceiling Accuracy	Actual S1+2	Max Gain	% Ceiling Captured
TACRED	False	41.7	93.4	73.6	51.7	61.8
TACRED	True	34.2	96.5	78.6	62.3	71.2
TACREV	False	39.4	94.0	75.2	54.6	65.6
TACREV	True	32.5	97.1	79.5	64.6	72.8
Re-TACRED	False	49.4	83.9	70.1	34.5	60.1
Re-TACRED	True	56.3	94.0	80.1	37.7	63.2

Table 7: Stage 2 Ceiling Analysis, $k = 8$ (perfect-confirmation upper bound vs. actual)

Dataset	CoT Training	Stage 1 Accuracy	Ceiling Accuracy	Actual S1+2	Max Gain	% Ceiling Captured
TACRED	False	41.5	94.1	74.1	52.7	61.9
TACRED	True	33.6	97.0	79.1	63.4	71.7
TACREV	False	44.2	94.5	75.0	50.4	61.2
TACREV	True	34.6	97.6	82.1	63.0	75.3
Re-TACRED	False	48.5	83.9	69.8	35.4	60.0
Re-TACRED	True	59.9	95.0	80.7	35.2	59.3

Table 8: Stage 2 Ceiling Analysis, $k = 16$

Dataset	CoT Training	Stage 1 Accuracy	Ceiling Accuracy	Actual S1+2	Max Gain	% Ceiling Captured
TACRED	False	45.3	94.2	72.5	48.9	55.8
TACRED	True	36.3	97.3	79.0	61.0	70.0
TACREV	False	42.2	94.7	74.6	52.5	61.9
TACREV	True	38.5	98.1	81.0	59.6	71.3
Re-TACRED	False	52.4	83.1	69.2	30.7	54.6
Re-TACRED	True	63.2	95.9	81.5	32.7	55.9

Table 9: Stage 2 Ceiling Analysis, $k = 32$

CoT Training	k	Dataset	Miss	Confusion	Correct
False	8	TACRED	13.8	17.1	69.1
False	8	TACREV	13.1	16.8	70.1
False	8	Re-TACRED	25.1	13.2	61.7
False	16	TACRED	13.7	13.7	72.6
False	16	TACREV	13.5	13.8	72.7
False	16	Re-TACRED	25.1	13.1	61.8
False	32	TACRED	13.9	13.2	72.9
False	32	TACREV	13.0	13.4	73.6
False	32	Re-TACRED	28.8	11.3	59.9
True	8	TACRED	1.4	14.9	83.7
True	8	TACREV	0.7	13.9	85.4
True	8	Re-TACRED	1.7	12.5	85.8
True	16	TACRED	1.0	13.0	86.0
True	16	TACREV	0.5	11.4	88.2
True	16	Re-TACRED	1.5	10.2	88.2
True	32	TACRED	1.1	11.5	87.4
True	32	TACREV	0.3	8.9	90.8
True	32	Re-TACRED	1.4	8.2	90.3

Table 10: Stage 1 named-relation performance across CoT settings, k, and dataset. The Miss, Confusion, and Correct values are all reported as percentages (%).

marily improves performance by preventing named relations from being incorrectly rejected as "no relation," while also modestly improving discrimination among named-relation classes.

A.7 CoT Ablation Results

We ran an ablation comparing CoT supervision (relation verbalization + CoT) to No-CoT supervision (relation verbalization only), for both Stage 1 (classification only) and Stage 1+2 (full pipeline), across $k = 8, 16, 32$. Comparisons use matched pairs with identical backbone, dataset, k, and seed, differing only in CoT use during training. The table reports the CoT effect (Δ = CoT − No-CoT) in points averaged across random seed and backbone model configurations.

At Stage 1, CoT-augmented supervision improves macro-recall across every dataset and k setting, with average gains of $+11.9$, $+12.0$, and $+13.0$ points at $k = 8, 16, 32$, respectively. This improvement is driven by better retrieval of named relations. For TACRED and TACREV, however, this comes at the cost of lower "no relation" recall relative to verbalization-only supervision, with declines ranging from -8.9 to -15.3 points, indicating increased over-prediction of named relations.

Re-TACRED behaves differently. There, CoT-augmented training yields "no relation" precision

k	Dataset	Class	Stage 1 ΔP	Stage 1 ΔR	Stage 1 ΔF_1	Stage 1+2 ΔP	Stage 1+2 ΔR	Stage 1+2 ΔF_1
8	TACRED	Macro	$+0.4$	$+9.9$	$+2.6$	$+9.5$	$+0.3$	$+7.5$
8	TACRED	No Relation	$+2.9$	-13.5	-8.5	$+1.3$	$+4.9$	$+3.9$
8	TACRED	Micro			-7.5			$+5.0$
8	TACREV	Macro	$+1.4$	$+12.1$	$+3.0$	$+9.6$	$+4.6$	$+8.0$
8	TACREV	No Relation	$+3.5$	-12.5	-7.2	$+2.4$	$+2.8$	$+3.0$
8	TACREV	Micro			-6.9			$+4.3$
8	Re-TACRED	Macro	$+5.5$	$+13.8$	$+8.5$	$+13.8$	$+4.0$	$+10.2$
8	Re-TACRED	No Relation	$+16.9$	-5.4	$+5.1$	$+8.4$	$+5.4$	$+7.5$
8	Re-TACRED	Micro			$+7.0$			$+10.0$
16	TACRED	Macro	-0.2	$+10.2$	$+1.6$	$+9.0$	$+3.2$	$+7.2$
16	TACRED	No Relation	$+3.5$	-13.7	-8.4	$+1.7$	$+4.7$	$+3.9$
16	TACRED	Micro			-7.9			$+5.0$
16	TACREV	Macro	$+1.0$	$+12.9$	$+2.5$	$+10.0$	$+5.8$	$+9.6$
16	TACREV	No Relation	$+3.8$	-15.0	-10.9	$+2.6$	$+6.3$	$+5.2$
16	TACREV	Micro			-8.8			$+7.1$
16	Re-TACRED	Macro	$+6.2$	$+13.1$	$+9.0$	$+13.6$	$+4.6$	$+10.2$
16	Re-TACRED	No Relation	$+18.8$	$+0.4$	$+12.3$	$+10.0$	$+4.9$	$+8.2$
16	Re-TACRED	Micro			$+11.3$			$+10.9$
32	TACRED	Macro	-0.4	$+10.3$	$+1.6$	$+10.1$	$+2.7$	$+8.5$
32	TACRED	No Relation	$+3.5$	-15.3	-11.0	$+1.8$	$+6.3$	$+5.1$
32	TACRED	Micro			-9.0			$+6.4$
32	TACREV	Macro	$+1.8$	$+13.3$	$+4.0$	$+9.4$	$+7.2$	$+9.9$
32	TACREV	No Relation	$+3.9$	-8.9	-4.1	$+3.2$	$+4.8$	$+4.7$
32	TACREV	Micro			-3.6			$+6.4$
32	Re-TACRED	Macro	$+5.6$	$+15.3$	$+9.7$	$+12.7$	$+6.5$	$+11.5$
32	Re-TACRED	No Relation	$+20.7$	-3.5	$+7.0$	$+11.5$	$+4.6$	$+9.0$
32	Re-TACRED	Micro			$+10.8$			$+12.3$

Table 11: CoT ablation: effect (Δ = CoT $-$ No-CoT) on precision, recall, and F_1 at Stage 1 and Stage 1+2

gains an order of magnitude larger than their verbalization-only counterparts ($+16.9$ to $+20.7$ points across $k = 8, 16, 32$, versus $+2.9$ to $+3.9$ points on TACRED/TACREV). Moreover, rather than the recall degradation seen on TACRED and TACREV, Re-TACRED's "no relation" recall at $k = 16$ actually turns positive ($+0.4$).

Stage 2 corrects the Stage 1 tradeoff: with "no relation" recall improving in all nine settings, while the macro-level gains are preserved. Consequently, full-pipeline micro-F_1 improves by $+5.0$, $+4.3$, $+10.0$ at $k = 8$, $+5.0$, $+7.1$, $+10.9$ at $k = 16$, and $+6.4$, $+6.4$, $+12.3$ at $k = 32$ for TACRED, TACREV, and Re-TACRED, respectively. CoT supervision therefore consistently strengthens Stage 1's retrieval of named relations, while Stage 2 filters the resulting false positives, demonstrating that CoT supervision and the two-stage design provide complementary rather than redundant improvements.

From Imitation to Reasoning: Sample-Efficient LLM Agents for Knowledge Graph Completion

Chenjun Zheng[1], Zhigang Wang[1], Wenjia Zhang[1], Zhe Wang[1*], Xiaolin Yang[1*]

[1]Institute of Basic Medical Sciences & School of Basic Medicine,
Chinese Academy of Medical Sciences & Peking Union Medical College
s2024005050@student.pumc.edu.cn,wangzhe@ibms.pumc.edu.cn,
yangxl@pumc.edu.cn

Abstract

While large language models (LLMs) have been introduced to knowledge graph completion (KGC) to improve multi-hop reasoning, most existing approaches still rely on embedding models for candidate pruning, which limits their structured reasoning capability. In contrast, fully path-based reasoning better reflects the nature of graph inference but suffers from extremely low sample efficiency. We propose AdaG-Reason, a path-based LLM-KGC framework. We formulate reasoning data synthesis as a non-stationary multi-armed bandit problem and introduce Active Reasoning Distillation to select informative trajectories by jointly considering distribution alignment, relation difficulty, and uncertainty. We further extend this bandit-based sampling principle to the reinforcement learning phase, adaptively allocating rollout queries by per-relation difficulty. For training, we propose Implicit Reasoning Distillation optimized via Step-wise Group Relative Policy Optimization (GRPO), enabling direct and stable learning of pure graph navigation policies without value function estimation. Experiments on YAGO3-10 and NELL-995 show that AdaG-Reason achieves competitive MRR and state-of-the-art Hits@1 on YAGO3-10, demonstrating the effectiveness of active data selection and pure policy learning for path-based LLM-KGC.

1 Introduction

Knowledge Graph Completion (KGC) aims to infer missing facts in a knowledge graph and is a fundamental problem in representation learning and symbolic reasoning. While embedding-based methods dominate current benchmarks, they perform inference through static similarity matching in latent spaces, lacking explicit reasoning processes and struggling to model multi-hop dependencies or provide interpretable decisions (Bordes et al., 2013; Sun et al., 2019; Dettmers et al., 2018).

To address these limitations, path-based reasoning methods reformulate KGC as a sequential decision-making problem on graphs (Xiong et al., 2018), where an agent navigates from a source entity by iteratively selecting relations and entities. Representative work such as MINERVA formulates multi-hop reasoning as a POMDP and enables reinforcement learningbased graph navigation (Das et al., 2018). However, traditional path-based methods remain constrained by exponentially growing search spaces and limited generalization: policies learned over symbolic or embedding-level actions often overfit to local structural patterns and fail to generalize across semantically equivalent but topologically distinct paths (Ma et al., 2025; Lin et al., 2018).

Recent advances in large language models (LLMs) offer a natural extension to path-based reasoning (Wei et al., 2023). By operating in a semantic space, LLMs can abstract relational meanings and functional roles of paths, enabling stronger semantic generalization than conventional policy networks (Ahn et al., 2022). In parallel, direct fine-tuning approaches such as KG-FIT (Jiang et al., 2024) and SAT (Liu et al., 2025) encode KG triples as textual sequences, leveraging the generative capacity of LLMs without explicit graph navigation. LLM-based graph reasoning frameworks leverage step-wise interaction (Yao et al., 2023) to perform multi-hop inference, demonstrating improved flexibility in complex decision making. However, this semantic expressiveness comes at a substantial cost. In graph reasoning, decision spaces grow combinatorially, and multiple reasoning paths may lead to the same correct answer (Das et al., 2018; Lei et al., 2020; Lin et al., 2018). As a result, naively training LLM-based agents to imitate full trajectories is inefficient, redundant, and often misaligned with the final reasoning objective.

These observations motivate a key question: how can LLMs acquire path-based reasoning without imitating full trajectories? We argue that effective training should focus on high-information decision

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 379–392
October 28, 2026 ©2026 Association for Computational Linguistics

signals rather than exhaustive path imitation (Russo and Roy, 2014; Auer et al., 2002). Our contributions are:

- **Active Reasoning Distillation (ARD):** a non-stationary bandit formulation for trajectory synthesis that jointly models distribution alignment, relation difficulty, and uncertainty, adaptively allocating a fixed budget toward learnable relations.
- **Implicit Reasoning Distillation (IRD):** we identify a semantic–topological mismatch in teacher-generated reasoning. Among 5,000 analyzed trajectories, only 2.2% contain explicit topological violations, whereas 89.6% are classified as implicit semantic misalignment: their graph actions remain valid, yet the resulting reasoning trajectory leads to an incorrect prediction. IRD addresses this mismatch through action-centric masking, training the policy on grounded actions without imitating the teacher's full CoT.
- **Step-wise GRPO with MAB-guided rollout:** a critic-free RL stage that unifies bandit-based sampling across both training phases, achieving state-of-the-art Hits@1 on YAGO3-10 (0.572) while training on trajectories covering only 6.6% of entities.

2 Related Work

Knowledge Graph Completion. Early research was dominated by **embedding-based methods**, which map entities and relations into vector spaces. Representative models like TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), and RotatE (Sun et al., 2019), along with structure-aware extensions (e.g., Schlichtkrull et al. (2017) and Vashishth et al. (2020)), achieve strong performance on benchmarks like YAGO3-10 (Suchanek et al.). To address the lack of explicit reasoning, **path-based approaches** formulate KGC as a multi-step decision process (e.g., MultiHop (Lin et al., 2018), M-Walk (Shen et al., 2018), and RuleGuider (Lei et al., 2020)). Notably, MINERVA (Das et al., 2018) models reasoning as a POMDP via reinforcement learning, establishing an end-to-end paradigm. Despite improving interpretability, these methods grapple with exponentially large search spaces, unstable training dynamics (Ma et al., 2025), and limited semantic generalization due to their reliance on symbolic representations (Lin et al., 2018).

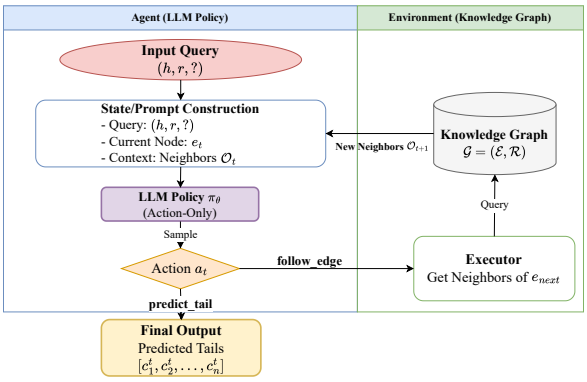


Figure 1: Illustration of the ReAct-style sequential decision process on the knowledge graph.

PLMs and LLM Agents for KGC. Pretrained Language Models (PLMs) such as KG-BERT (Yao et al., 2019), SaTR (Wang et al., 2021), and SimKGC (Wang et al., 2022) enhance semantic modeling by encoding triples as text. However, they remain fundamentally similarity-based, serving as semantic encoders rather than performing explicit sequential reasoning. Recently, Large Language Models (LLMs) have evolved into agents capable of interleaving reasoning and action (Zhao et al., 2025; Tan et al., 2025). Frameworks like ReAct (Yao et al., 2023), combined with policy optimization algorithms such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024), have demonstrated effectiveness in complex reasoning tasks (Shinn et al., 2023). In parallel, direct LLM fine-tuning approaches such as KG-FIT (Jiang et al., 2024) and SAT (Liu et al., 2025) encode KG triples as textual sequences for generative training, achieving competitive results by leveraging the semantic capacity of pretrained LMs without explicit graph traversal. Nevertheless, adapting these agent frameworks to large-scale KGC remains challenging due to rigid structural constraints, vast discrete action spaces, and low sample efficiency.

3 Methodology

3.1 Preliminaries

A knowledge graph $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ consists of entities $\mathcal{E}$, relations $\mathcal{R}$, and triples $\mathcal{T} = \{(h, r, t)\}$. KGC answers queries $(h, r, ?)$ by ranking candidate tail entities.

We formulate multi-hop reasoning as a POMDP (Figure 1). Given a query $q = (h, r, ?)$, at step t

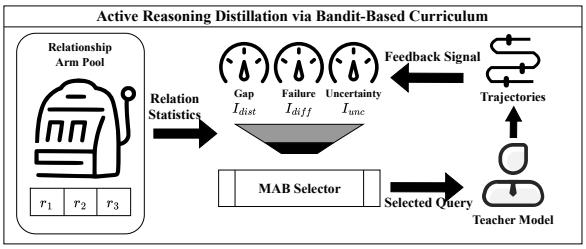


Figure 2: Overview of the **Active Reasoning Distillation** framework. The MAB Selector dynamically prioritizes relations through a hierarchical acquisition function. Teacher-generated trajectories are validated and fed back to update the bandit statistics in a closed loop.

the agent positioned at entity e_t observes a local subgraph O_t and conditions on $x_t = (q, O_t)$. The LLM parameterizes a stochastic policy over the vocabulary $\mathcal{V}$ (full prompt template in Appendix E):

$$P_\theta(w \mid x_t) = \mathrm{Softmax}(\mathrm{Logits}_\theta(x_t)). \quad (1)$$

The action space has two types. A *path-expansion* action $a_t^{\mathrm{f}} = (r', e')$ follows an outgoing edge $(e_t, r', e') \in \mathcal{G}$. A *termination* action $a_t^{\mathrm{p}} = L_t$ outputs a ranked candidate list $L_t = [c_1, \ldots, c_k]$ ($k \le K_{\max}$) in descending confidence, with probability:

$$P(a_t^{\mathrm{p}} \mid x_t) \propto \prod_{i=1}^{k} P_\theta(c_i \mid x_t, c_{<i}). \quad (2)$$

A trajectory $\tau = (o_0, a_0, \ldots, a_T)$ ends with a terminal `predict_tail` action producing L_T. The objective maximizes the probability of ranking the ground-truth entity e^* in L_T:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\big[\mathbb{I}(e^* \in L_T)\big]. \quad (3)$$

Directly optimizing this objective is prohibitive: the graph branches exponentially, and rewards are sparse (binary success at trajectory termination). We address this through two stages—first **bootstrapping** the policy from selectively distilled teacher trajectories (Section 3.2), then **reinforcement fine-tuning** (Section 3.3).

3.2 Distillation-Based Policy Bootstrapping

3.2.1 Active Reasoning Distillation (ARD)

We cast trajectory synthesis under a fixed budget as a non-stationary multi-armed bandit (Figure 2), where the reward distribution shifts as per-relation success rates evolve with accumulated trajectories. Each relation $r \in \mathcal{R}$ is an arm; pulling arm r randomly selects a head entity h from the training triples of r to form query $(h, r, ?)$, then queries a teacher LLM for a trajectory. The goal is to allocate budget toward relations yielding *learnable* trajectories rather than uniform exploration.

Composite Acquisition Function. The priority score for relation r at iteration t combines three terms:

$$S_t(r) = \mathcal{I}_{\text{dist}}(r) + \lambda_1 \mathcal{I}_{\text{diff}}(r) + \lambda_2 \mathcal{I}_{\text{unc}}(r), \quad (4)$$

where $\lambda_1 > \lambda_2$ prioritizes learnability-driven sampling over pure exploration.

Distribution Alignment. Let $P^*(r)$ denote the empirical relation frequency in the training set, N_{total} the total budget, and $N_t(r)$ the number of valid trajectories collected for r up to iteration t.

$$\mathcal{I}_{\text{dist}}(r) = \max\Big(0, \frac{N_{\text{total}} P^*(r) - N_t(r)}{N_{\text{total}} P^*(r) + \epsilon}\Big) \quad (5)$$

This term prioritizes relations whose trajectory count trails their target quota.

Difficulty Awareness. Beyond coverage, not all relations are equally learnable. Let $\mu_t(r) = V_t(r)/A_t(r)$ be the empirical success rate for r, where $A_t(r)$ and $V_t(r)$ count total attempts and valid trajectories. A relation is *abandoned* when $A_t(r) > A_{\text{abandon}}$ yet $\mu_t(r) < \mu_{\min}$. We model learnability via a Gaussian window centered at μ_{target}:

$$\mathcal{I}_{\text{diff}}(r) = \begin{cases} 0, & \text{abandoned,} \\ \exp\Big(-\frac{(\mu_t(r) - \mu_{\text{target}})^2}{2\sigma^2}\Big), & \text{otherwise.} \end{cases} \quad (6)$$

Gated Uncertainty Exploration. We encourage exploration of under-sampled relations while suppressing unlearnable ones by gating the UCB term with $\mu_t(r)^\gamma$:

$$\mathcal{I}_{\text{unc}}(r) = \sqrt{\frac{2\ln(\sum_{r'} N_t(r') + 1)}{N_t(r) + 1}} \cdot \mu_t(r)^\gamma. \quad (7)$$

Closed-Loop Distillation. ARD proceeds iteratively: S_t is Softmax-transformed into a sampling distribution; a batch of relations is drawn, queries are instantiated and submitted to the teacher; $N_t(r)$ and $\mu_t(r)$ are updated from trajectory validity. The three acquisition terms interact dynamically: early iterations are dominated by distribution alignment (broad coverage), mid-stage by difficulty awareness (concentrating on the learnability window),

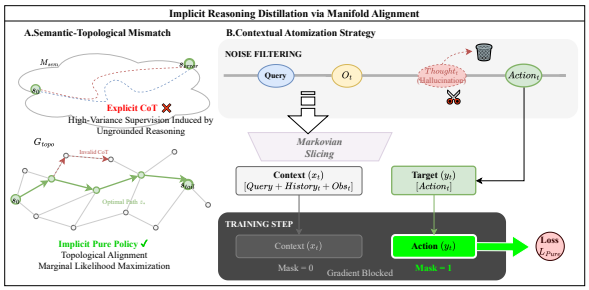


Figure 3: Implicit Reasoning Distillation (IRD). (A) Semantic–Topological Mismatch: semantically plausible teacher reasoning does not always translate into correct graph-constrained decisions. (B) Action-centric masking: gradients are restricted to grounded action tokens, reducing supervision from potentially misaligned CoT.

and late-stage by gated uncertainty (suppressing exhausted or hopeless relations). This phase transition, visualized in Appendix C, yields an adaptive curriculum under a fixed budget. Each synthesized trajectory pairs grounded actions with the teacher's CoT reasoning. As we show next, the free-form reasoning can be semantically plausible yet misaligned with the graph-constrained decision process.

3.2.2 Implicit Reasoning Distillation (IRD)

To mitigate the semantic–topological mismatch introduced by explicit CoT supervision (quantified in Appendix B; Figure 3), we propose *Implicit Reasoning Distillation (IRD)*. Rather than directly supervising the student with the teacher's full free-form reasoning trace, IRD treats reasoning as a latent variable and optimizes the policy through grounded action supervision, reducing its dependence on potentially misaligned CoT.

Theoretical Motivation. LLM reasoning can be understood as an interaction between two manifolds: the continuous semantic manifold $\mathcal{M}_{\text{sem}}$ from pretraining, and the discrete topological manifold $\mathcal{G}_{\text{topo}}$ of the knowledge graph. Teacher-generated traces z_{syn}, sampled from $\mathcal{M}_{\text{sem}}$, may diverge from the structured decisions required by $\mathcal{G}_{\text{topo}}$. We distinguish explicit topological violations, where a navigation action does not correspond to a valid KG edge, from grounded but outcome-misaligned trajectories, where all executed edges are valid but the final prediction is incorrect. Across 5,000 teacher trajectories on YAGO3-10, 89.6% fall into the latter category, while only 2.2% contain explicit topological violations (Appendix B).

This mismatch creates two problems for explicit CoT supervision. First, *spurious correlation*: when the teacher's free-form reasoning is misaligned with the graph-constrained decision while the associated action remains useful for supervision, directly optimizing the full CoT can encourage the student to model irrelevant or misleading reasoning signals. Second, *high-variance gradients*: the CoT trace spans many tokens whose gradient $\nabla \log P_\theta(z_{\text{syn}} \mid x)$ competes with the action gradient, diluting supervision on the decisions that matter. By masking CoT tokens and backpropagating only through actions, IRD reduces exposure to both sources of noise and concentrates the training signal on the state-to-action mapping.

Action-Centric Optimization. We implement IRD via step-wise behavioral cloning with an action-centric masking strategy. Given a flattened trajectory sequence $X = [x_t; y_t]$, we construct a binary mask M where $M_i = 1$ if $X_i \in \text{Tokens}(y_t)$ and 0 otherwise. The objective is:

$$\mathcal{L}_{\text{BC}}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{step}}} \sum_i M_i \log P_\theta(X_i \mid X_{<i}). \tag{8}$$

Importantly, IRD primarily serves as an action-grounded policy initialization rather than the final optimization stage. On YAGO3-10, the IRD/SFT-only policy achieves 0.324 Hits@1, which is subsequently improved to 0.568 by GRPO. By concentrating supervision on executable graph actions, IRD provides a lower-noise initialization for subsequent policy optimization.

3.3 Reinforcement Fine-Tuning with Bandit-Guided Rollout

3.3.1 Step-wise GRPO

Starting from the IRD-initialized policy π_{BC}, we refine it via Step-wise Group Relative Policy Optimization (GRPO), a critic-free algorithm using group-based relative feedback (Figure 4).

Rank-Aware Reward & Advantage. To mitigate the "hedging" phenomenon where models generate excessive candidates, we define a Rank-Aware Reward that strictly enforces Top-1 precision while penalizing path length:

$$R(\tau) = \frac{\mathbb{I}(e_{\text{gold}} \in L_{\text{pred}})}{\text{Rank}(e_{\text{gold}})} - \lambda_{\text{len}}(|\tau| - 1). \tag{9}$$

For each query, we sample a group of G trajectories. The advantage A_i for trajectory τ_i is computed by

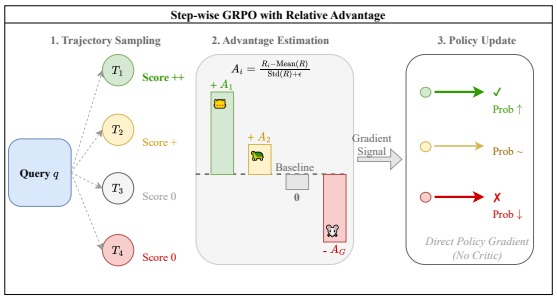


Figure 4: Step-wise GRPO with Relative Advantage. The process iterates through: (1) sampling a group of trajectories; (2) computing relative advantages based on rank-aware rewards; and (3) updating the policy directly to reinforce high-quality reasoning paths without a critic network.

normalizing rewards against the group statistics: $A_i = (R_i - \mu_R)/(\sigma_R + \epsilon)$. To address temporal credit assignment, we modulate this advantage with an exponential decay weight $\omega_{i,t} = \gamma^{T_i - t}$, assigning higher credit to actions closer to the final prediction—critical in graph navigation, where early exploration decisions determine which subgraph is reachable. The GRPO objective maximizes the weighted likelihood of step-wise actions while penalizing deviation from the reference policy π_{ref}:

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{|\tau_i|}\omega_{i,t}\Big[A_i \log \pi_\theta(a_{i,t} \mid x_{i,t}) - \beta D_{\text{KL}}\big(\pi_\theta || \pi_{\text{ref}}\big)_t\Big], \tag{10}$$

where $D_{\text{KL}}(\cdot)_t$ is the per-step KL penalty for training stability.

3.3.2 MAB-Guided Rollout Sampling

Each GRPO iteration requires sampling rollout queries. Uniform sampling wastes budget on relations the policy already masters. We extend the bandit principle to the RL phase: per-relation success rates from the preceding evaluation determine sampling weights, assigning higher weight to relations where the policy struggles. However, naively concentrating all rollout budget on the hardest relations introduces a training stability risk: in early RL iterations, when the policy is still fragile, batches dominated by near-impossible queries produce sparse, noisy rewards that can destabilize the policy before it consolidates basic graph navigation patterns.

To mitigate this, we introduce a *Relation Buffer* term that guarantees every relation a minimum sampling probability, preventing the distribution from collapsing to extreme difficulty. Formally, given the policy's per-relation success rate $\mu_t^{\text{eval}}(r)$ at RL iteration t, the sampling weight for relation r is

$$w(r) = \underbrace{\big(1 - \mu_t^{\text{eval}}(r)\big)}_{\text{difficulty}} + \underbrace{\beta}_{\text{buffer}}, \tag{11}$$

where $\beta > 0$ controls the minimum allocation share. The rollout query distribution is then

$$P(r) = \frac{w(r)}{\sum_{r'} w(r')}. \tag{12}$$

When β is small, allocation is dominated by failure rates, concentrating updates on weaknesses; when β is large, the distribution approaches uniformity, ensuring even mastered relations ($\mu_t^{\text{eval}} \to 1$) retain a non-negligible sampling probability.

Together with ARD, this forms a unified bandit-based data selection strategy spanning both training stages.

Dataset	Entity	Relation	Train	Valid	Test
YAGO3-10	123,182	37	1,079,040	5,000	5,000
NELL-995	75,492	200	154,213	543	2,818

Table 1: Summary statistics of benchmark datasets.

4 Experiment

4.1 Experimental Setup

Datasets We evaluate our method on two distinct benchmarks: YAGO3-10 (Suchanek et al.) and NELL-995 (Xiong et al., 2018). YAGO3-10 is a large-scale subset of YAGO3, characterized by a high number of entities and a long-tail relation distribution, serving as a stress test for scalability. NELL-995, derived from the Never-Ending Language Learning system, serves as a standard testbed for multi-hop reasoning. It features complex path dependencies and specific reasoning tasks that rigorously test the agent's ability to perform logical deduction over multiple steps. Table 1 summarizes the statistics of these datasets. We exclude FB15k-237 and WN18RR; justification is provided in Appendix A.

Evaluation Metrics We assess performance using standard KGC metrics: Mean Reciprocal Rank (MRR) and Hits@k ($k = 1, 10$). Hits@k measures the proportion of test triples where the ground-truth entity is ranked within the top-k candidates. We employ the *filtered ranking setting* proposed by

Bordes et al. (2013), excluding other known true triples $(h, r, t') \in \mathcal{G}_{\text{train}} \cup \mathcal{G}_{\text{valid}} \cup \mathcal{G}_{\text{test}}$ $(t' \neq t)$ during evaluation.

For LLM-based prediction, entity names returned by `predict_tail` are mapped to KG entity IDs using case-insensitive exact matching, and the ranked valid candidates are used for metric computation. For SC@10, we independently sample 10 reasoning trajectories with temperature 0.7 and aggregate their final predictions via majority voting.

Implementation Details. We employ DeepSeek-V3.2 (DeepSeek-AI, 2025) as the teacher to synthesize ARD-curated trajectories, yielding 4,498 valid trajectories for YAGO3-10 and 12,654 for NELL-995 (detailed synthesis statistics in Appendix D). The student model, Qwen3-8B (Yang et al., 2025), undergoes full-parameter SFT followed by Stepwise GRPO, both on 8×A100 GPUs with DeepSpeed ZeRO-3. Full hyperparameters are provided in Appendix D.

Table 2 details the coverage profile of the ARD-synthesized trajectories relative to the full training graphs.

Table 2: Trajectory coverage statistics. *Unique Query Triples* counts distinct $(h, r, ?)$ queries; *Unique Entities* counts distinct entities appearing in any state or observation; *Unique Edges* counts distinct (h, r, t) traversed. Percentages denote coverage of the full-graph entity set (123,182 for YAGO3-10, 75,492 for NELL-995).

	YAGO3-10	**NELL-995**
Synthesized trajectories	4,498	12,654
Unique query triples	4,439	5,996
Unique entities	8,113	7,055
(% of full graph)	(6.6%)	(9.3%)
Unique edges traversed	5,483	6,262
Relations covered	34/37	182/200
Full-graph training triples	1,079,040	154,213

On YAGO3-10, the ARD process yields 4,498 valid trajectories covering 4,439 distinct queries across 34 of 37 relations; the three uncovered relations receive negligible allocation due to persistently low teacher success rates—a consequence of $\mathcal{I}_{\text{dist}}$ and $\mathcal{I}_{\text{diff}}$ in Eq. 4 that jointly suppress unproductive exploration. On NELL-995, 12,654 trajectories span 182 of 200 relations; the remaining 18 are extreme long-tail cases where the teacher failed to produce valid reasoning chains despite extended exploration. This coverage profile highlights a key distinction from embedding-based methods: AdaG-Reason trains exclusively on relation-diverse, ARD-filtered trajectories rather than the full triple set. The resulting trajectories cover only 6.6% and 9.3% of graph entities, respectively, indicating that the student is trained from a sparse subset of the available graph supervision. Here, *sample efficiency* refers to the amount and coverage of retained supervision used for student training, rather than end-to-end computational efficiency; detailed synthesis and training costs are reported in Appendix D.5.

4.2 Main Results

Table 3 presents the performance comparison against a broad range of state-of-the-art baselines on both datasets. We include classical and contemporary embedding-based models, recent neural graph reasoning approaches, and text-based methods.

Precision and the Role of Design Choices. AdaG-Reason achieves the highest Hits@1 on both datasets (0.572 on YAGO3-10; 0.518 on NELL-995), surpassing KNOWFORMER, and reaches MRR 0.597 on YAGO3-10. This precision follows from the design choices in Section 3. ARD's bandit curriculum (Section 3.2) ensures the policy is bootstrapped from learnable trajectories rather than uniformly sampled noise. IRD's action-centric masking (Section 3.2.2) avoids directly supervising the student with potentially misaligned CoT tokens, reducing the impact of semantic–topological mismatch during distillation. GRPO with MAB-guided rollout (Section 3.3) further sharpens the policy by concentrating reinforcement signals on relations where the model is most uncertain. On NELL-995, this precision focus comes at a trade-off in MRR (0.558 vs. KNOWFORMER 0.566), reflecting the rank-aware reward's structural bias toward Top-1 accuracy. Each component addresses a specific source of optimization noise, and the consistent Top-1 accuracy across two structurally distinct benchmarks suggests that their combined effect favors decisive, high-confidence predictions.

Relational Complexity vs. Entity Scale. YAGO3-10 (123k entities, 37 relations) achieves higher absolute performance than NELL-995 (75k entities, 200 relations) under the same training paradigm. This counter-intuitive result—the larger entity set yielding better performance—aligns with the bandit formulation of ARD: the difficulty of path-based KGC is not determined by the number of entities but by the diversity of the relation space. YAGO3-10's 37 hierarchically clear re-

Table 3: Main Results on YAGO3-10 and NELL-995. The best results are highlighted in **bold**, and the second-best are underlined. ± denotes standard deviation over 3 training seeds. Direct: greedy single-path decoding; SC@10: self-consistency decoding with 10 paths.

Method	YAGO3-10			NELL-995		
	MRR	H@1	H@10	MRR	H@1	H@10
TransE (Bordes et al., 2013)	0.510	0.413	0.681	0.507	0.424	0.648
RotatE (Sun et al., 2019)	0.495	0.402	0.670	0.508	0.448	0.608
DistMult (Yang et al., 2015)	0.566	0.491	0.704	0.510	0.438	0.636
HousE (Li et al., 2022)	0.571	0.491	0.714	0.519	0.458	0.618
AdaProp (Zhang et al., 2023)	0.573	0.510	0.685	0.554	0.493	0.655
NBFNet (Zhu et al., 2022)	0.563	0.480	0.708	0.525	0.451	0.639
DRUM (Sadeghian et al., 2019)	0.531	0.453	0.676	0.532	0.460	0.662
CompGCN (Vashishth et al., 2020)	0.421	0.392	0.577	0.463	0.383	0.596
SimKGC (Wang et al., 2022)	0.211	0.141	0.351	0.501	0.426	0.653
KNOWFORMER (Liu et al., 2024)	0.615	0.547	**0.734**	**0.566**	0.502	**0.675**
KG-FIT (Jiang et al., 2024)	0.568	0.474	0.718	-	-	-
SAT (Liu et al., 2025)	0.616	0.525	0.726	-	-	-
AdaG-Reason (Direct)	0.597±0.007	0.568±0.008	0.640±0.006	0.539±0.008	0.518±0.009	0.565±0.007
AdaG-Reason (SC@10)	**0.617±0.004**	**0.572±0.005**	0.678±0.004	0.558±0.005	**0.518±0.005**	0.613±0.004

lations present a concentrated learning problem, while NELL-995's 200 diverse relations demand complex, long-range reasoning chains that stretch the capacity of both teacher and student. This observation suggests that relation diversity and reasoning depth may be more informative indicators of difficulty than entity count alone in agent-based KGC.

Self-Consistency and the Single-Path Trade-off. Direct decoding trails global embedding methods on Hits@10, reflecting the recall limitation inherent in greedy single-path inference. Self-consistency decoding (SC@10) bridges this gap without retraining: sampling 10 independent paths and aggregating via majority voting raises YAGO3-10 MRR to 0.617, surpassing KNOWFORMER, and lifts Hits@10 to 0.678. That an opt-in, training-free decoding strategy closes much of the gap suggests that a substantial portion of the Hits@10 deficit arises from the high-confidence search behavior encouraged by the rank-aware reward (Section 3.3.1), which prioritizes Top-1 precision over recall.Notably, SC@10 yields marginal H@1 improvement on both datasets (+0.004 on YAGO3-10, +0.000 on NELL-995), indicating that the policy's top-ranked prediction is already accurate and stable; the gain from self-consistency lies in recovering correct candidates that fall outside the top-1, not in correcting the top-1 itself.

4.3 Ablation Studies

We conduct comprehensive ablation studies on the YAGO3-10 dataset to validate our architectural choices and training mechanisms.

4.3.1 Impact of Reasoning Modality

We contrast our implicit reasoning paradigm against three baselines: (1) Zero-shot ReAct (using Qwen3-32B-Instruct as a comparable-scale reference; the 8B-Instruct variant was unavailable at the time of experiments); (2) Direct SFT, which treats KGC as internal memorization; and (3) Explicit Chain-of-Thought (CoT) ReAct. All fine-tuned variants use Qwen3-8B and are evaluated after the IRD/SFT stage. Results on YAGO3-10 and NELL-995 are shown in Table 4.

Table 4: Ablation on reasoning modality on YAGO3-10 and NELL-995. Models are evaluated after the IRD/SFT stage under full-parameter training.

Variant	YAGO3-10		NELL-995	
	MRR	H@1	MRR	H@1
Zero-shot ReAct	0.067	0.066	0.026	0.026
Memorization SFT	0.063	0.063	0.120	0.116
CoT ReAct SFT	0.152	0.147	0.216	0.209
AdaG-Reason SFT only	**0.324**	**0.324**	**0.339**	**0.339**

ReAct vs. Memorization. Zero-shot ReAct and Direct SFT both perform poorly, supporting the premise of our agent design: large-scale KGC is fundamentally a reasoning-and-acting task rather than one that relies solely on internal knowledge recall. Parametric memory alone is insufficient to cover the rich factual relations in large-scale knowledge graphs, supporting the POMDP formulation in Section 3.1 that requires active graph ex-

ploration.

Implicit vs. Explicit Reasoning. Across both datasets, CoT ReAct SFT substantially underperforms IRD, providing further support for the semantic–topological mismatch analysis in Section 3.2.2. Explicit reasoning traces generated by the teacher in semantic space may deviate from effective decision paths in the graph environment; forcing the student to imitate these intermediate reasoning traces can therefore introduce additional optimization noise. By restricting supervision to grounded actions, IRD enables the student to learn a more direct mapping from graph states to structurally valid actions, yielding a stronger initialization for subsequent policy optimization.

4.3.2 Impact of Learning Paradigm

We isolate the contribution of the RL stage by comparing the supervised baseline (IRD/SFT-Only) versus the full pipeline (SFT + GRPO). Results are shown in Table 5. Full-parameter SFT alone yields

Table 5: Ablation on Impact of Learning Paradigm. Full-parameter SFT vs. SFT + GRPO.

Dataset	Stage	MRR	H@1	H@10
YAGO3-10	Full SFT only	0.324	0.324	0.325
YAGO3-10	Full SFT + GRPO	0.597	0.568	0.640
YAGO3-10	+ SC@10	**0.617**	**0.572**	**0.678**
NELL-995	Full SFT only	0.339	0.339	0.340
NELL-995	Full SFT + GRPO	0.539	0.518	0.565
NELL-995	+ SC@10	**0.558**	**0.518**	**0.613**

near-identical H@1 and H@10 on both datasets, indicating that the policy gains little benefit from ranking additional candidates at this stage. GRPO substantially improves MRR and Hits@k on both datasets, demonstrating the importance of subsequent policy optimization beyond supervised initialization. The rank-aware reward (Section 3.3.1) is designed to reward precise Top-1 predictions while penalizing hedging, encouraging the policy to place greater confidence on high-ranked candidates. SC@10 further lifts YAGO3-10 MRR to 0.617, showing that additional inference-time sampling can further improve candidate ranking without retraining.

4.3.3 Impact of Bandit-Based Adaptive Sampling

We evaluate bandit-based adaptive sampling at both the data synthesis and RL rollout stages on YAGO3-10 (Table 6). At the synthesis stage, we compare topology-only BFS sampling, uniform teacher distillation, and ARD. At the RL stage, we independently vary the rollout strategy under both uniform and ARD-based SFT initializations.

Table 6: Data synthesis and RL rollout strategies on YAGO3-10.

SFT	Rollout	MRR	H@1
	SFT only		
BFS Random	–	0.047	0.043
Uniform	–	0.206	0.204
ARD	–	0.324	0.324
	SFT + GRPO		
Uniform	Uniform	0.456	0.423
Uniform	MAB	0.525	0.491
ARD	Uniform	0.526	0.505
ARD	MAB	0.597	0.568
ARD	MAB + SC@10	**0.617**	**0.572**

Effect of Adaptive Sampling. At the synthesis stage, BFS Random produces topologically valid but semantically unguided trajectories and performs poorly, highlighting the importance of teacher-guided trajectory generation. Uniform teacher sampling substantially improves over BFS Random, while ARD further improves the SFT initialization by adaptively allocating the synthesis budget toward more learnable relations. Importantly, this advantage persists after subsequent GRPO training.

At the RL stage, MAB-guided sampling consistently outperforms uniform sampling under both SFT initialization strategies, suggesting that relation-difficulty-aware rollout allocation provides additional gains beyond those obtained from adaptive distillation. Overall, adaptive allocation is beneficial at both data synthesis and RL exploration stages, with ARD and MAB-guided rollout achieving the strongest training-time performance. SC@10 further improves the final result at inference time without additional training.

ARD Allocation Behavior. The ARD bandit policy dynamically allocates the synthesis budget across three regimes: easy relations are exploited early and throttled upon convergence, medium-difficulty relations receive the majority of investment, and persistently hard relations are suppressed to prevent low-quality trajectories from diluting the training signal. The complete analysis and visualization are provided in Appendix C.

4.4 Reasoning Depth Analysis

To analyze the actual reasoning depth of the learned policy, we group test trajectories by the number of graph-navigation steps executed before the final prediction and report the trajectory ratio and Hits@1 at each depth.

Table 7: Performance across reasoning depths. Ratio denotes the proportion of test trajectories terminating at each depth.

Steps	YAGO3-10		NELL-995	
	Ratio	H@1	Ratio	H@1
1	5.7%	0.341	4.1%	0.291
2	23.2%	**0.769**	17.6%	**0.688**
3	33.9%	0.738	30.5%	0.651
4	23.9%	0.436	27.1%	0.491
5	13.4%	0.188	20.7%	0.228

Across both datasets, most trajectories involve multi-step graph navigation, while only a small fraction terminate after a single step. Performance is strongest at intermediate depths of 2–3 steps and degrades noticeably as trajectories become longer. This pattern indicates that the policy can effectively exploit moderate-depth multi-hop paths, while longer reasoning chains remain more susceptible to path uncertainty and error accumulation.

5 Conclusion

We present AdaG-Reason, a framework where LLM agents acquire graph reasoning through actively curated demonstrations (ARD), action-grounded supervision (IRD), and trial-and-error refinement (Step-wise GRPO). Our analysis identifies a semantic–topological mismatch in teacher supervision: explicit topological violations are rare (2.2%), whereas 89.6% of analyzed trajectories exhibit implicit semantic misalignment. This finding motivates IRD's action-centric supervision, which avoids direct imitation of the teacher's full CoT. Across YAGO3-10 and NELL-995, AdaG-Reason consistently improves Top-1 prediction through adaptive data selection and policy optimization. On YAGO3-10, it achieves state-of-the-art Hits@1 (0.572) while training on trajectories covering only 6.6% of graph entities.

Acknowledgments

This work was supported by the Beijing Natural Science Foundation (7254504), the Noncommunicable Chronic Diseases–National Science and Technology Major Project (2024ZD0532900), the National Natural Science Foundation of China (No. 32670888), and the CAMS Innovation Fund for Medical Sciences (CIFMS, No. 2021-I2M-1-057).

Limitations

ARD relies on the teacher producing at least a minimal number of valid trajectories per relation; relations where the teacher consistently fails receive little or no allocation (3/37 on YAGO3-10 and 18/200 on NELL-995), limiting the framework on queries beyond the teacher's reasoning competence. Although the retained supervision used for student training is sparse, ARD still incurs substantial teacher-side synthesis cost because trajectory generation requires repeated API calls and filtering.

The current framework is primarily designed and optimized for graph reasoning settings represented by YAGO3-10 and NELL-995, where explicit multi-step navigation and relational composition play an important role. Its effectiveness may not directly transfer to datasets with substantially different graph structures or reasoning characteristics, such as regimes dominated by short-range associations, highly hierarchical relations, or different connectivity patterns. Adapting the sampling strategy, navigation policy, or training objective to such graph regimes remains an important direction for future work.

The single-path formulation favors Top-1 accuracy at the expense of candidate recall; although SC@10 partially mitigates this trade-off at inference time, it introduces additional latency. More generally, AdaG-Reason incurs higher inference cost than embedding-based methods because each query requires sequential model interaction and multiple forward passes.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and 26 others. 2022. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*.

Peter Auer, Nicol O Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 27872795, Red Hook, NY, USA. Curran Associates Inc.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. *arXiv preprint*. ArXiv:1711.05851 [cs].

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. *arXiv preprint*. ArXiv:1707.01476 [cs].

Pengcheng Jiang, Lang Cao, Cao Xiao, Parminder Bhatia, Jimeng Sun, and Jiawei Han. 2024. Kg-fit: Knowledge graph fine-tuning upon open-world knowledge. *Preprint*, arXiv:2405.16412.

Deren Lei, Gangrong Jiang, Xiaotao Gu, Kexuan Sun, Yuning Mao, and Xiang Ren. 2020. Learning Collaborative Agents with Rule Guidance for Knowledge Graph Reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8541–8547, Online. Association for Computational Linguistics.

Rui Li, Jianan Zhao, Chaozhuo Li, Di He, Yiqi Wang, Yuming Liu, Hao Sun, Senzhang Wang, Weiwei Deng, Yanming Shen, Xing Xie, and Qi Zhang. 2022. House: Knowledge graph embedding with householder parameterization. *Preprint*, arXiv:2202.07919.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253, Brussels, Belgium. Association for Computational Linguistics.

Junnan Liu, Qianren Mao, Weifeng Jiang, and Jianxin Li. 2024. Knowformer: Revisiting transformers for knowledge graph reasoning. *Preprint*, arXiv:2409.12865.

Yu Liu, Yanan Cao, Xixun Lin, Yanmin Shang, Shi Wang, and Shirui Pan. 2025. Enhancing large language model for knowledge graph completion via structure-aware alignment-tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20970–20984, Suzhou, China. Association for Computational Linguistics.

Ying Ma, Owen Burns, Mingqiu Wang, Gang Li, Nan Du, Laurent El Shafey, Liqiang Wang, Izhak Shafran, and Hagen Soltau. 2025. Knowledge Graph Reasoning with Self-supervised Reinforcement Learning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:1508–1519. ArXiv:2405.13640 [cs].

Daniel Russo and Benjamin Van Roy. 2014. Learning to optimize via information directed sampling. *CoRR*, abs/1403.5556.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. Drum: End-to-end differentiable rule mining on knowledge graphs. *Preprint*, arXiv:1911.00055.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. *arXiv preprint*. ArXiv:1703.06103 [stat].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint*. ArXiv:1707.06347 [cs].

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint*. ArXiv:2402.03300 [cs].

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-Walk: Learning to Walk over Graphs using Monte Carlo Tree Search. *arXiv preprint*. ArXiv:1802.04394 [cs].

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint*. ArXiv:2303.11366 [cs].

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. *arXiv preprint*. ArXiv:1902.10197 [cs].

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025. Paths-over-Graph: Knowledge Graph Empowered Large Language Model Reasoning. *arXiv preprint*. ArXiv:2410.14211 [cs].

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. *CoRR*, abs/1606.06357.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. *arXiv preprint*. ArXiv:1911.03082 [cs].

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, and Yi Chang. 2021. Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748. ArXiv:2004.14781 [cs].

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. SimKGC: Simple Contrastive Knowledge Graph Completion with Pre-trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, Dublin, Ireland. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint*. ArXiv:2201.11903 [cs].

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2018. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. *arXiv preprint*. ArXiv:1707.06690 [cs].

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *Preprint*, arXiv:1412.6575.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *arXiv preprint*. ArXiv:1909.03193 [cs].

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint*. ArXiv:2210.03629 [cs].

Yongqi Zhang, Zhanke Zhou, Quanming Yao, Xiaowen Chu, and Bo Han. 2023. Adaprop: Learning adaptive propagation for graph neural network based knowledge graph reasoning. *Preprint*, arXiv:2205.15319.

Qi Zhao, Hongyu Yang, Qi Song, Xinwei Yao, and Xiangyang Li. 2025. KnowPath: Knowledge-enhanced Reasoning via LLM-generated Inference Paths over Knowledge Graphs. *arXiv preprint*. ArXiv:2502.12029 [cs].

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2022. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Preprint*, arXiv:2106.06935.

A Applicability Scope and Dataset Selection

In this work, we primarily study explicit logical reasoning over graph structures, where inference requires multi-hop traversal and structured decision-making, rather than implicit semantic similarity matching. Accordingly, we conduct experiments on YAGO3-10 and NELL-995, which are commonly used benchmarks for path-based and rule-like reasoning. In this appendix, we clarify the scope of our evaluation and discuss the considerations behind excluding FB15k-237 and WN18RR.

Reasoning Type: Logical Inference vs. Semantic Correlation. FB15k-237 contains a large number of relations whose predictions can often be resolved through direct semantic associations between entities (e.g., `/award/award_nominee` $\leftrightarrow$ `/award/award_winner`). Prior work has shown that embedding-based models (e.g., RotatE, TuckER) are particularly effective in such settings by capturing symmetric or anti-symmetric patterns in a continuous latent space. In contrast, our method focuses on explicit path-based reasoning, where correct predictions depend on composing multiple relational steps (e.g., $A \xrightarrow{\text{bornIn}} B \xrightarrow{\text{locatedIn}} C \Rightarrow A \xrightarrow{\text{nationality}} C$). YAGO3-10 and NELL-995 contain a higher proportion of such multi-hop, rule-like relational patterns, making them more suitable for evaluating sequential decision-making agents. As a result, we do not expect the advantages of explicit reasoning policies to be fully reflected on FB15k-237.

Graph Topology and Relational Diversity. WN18RR exhibits a predominantly hierarchical structure with a low average node degree, where reasoning is largely constrained to hypernym and hyponym relations. While this dataset is well-suited for studying hierarchical representation learning, it provides limited relational diversity for evaluating general-purpose graph navigation agents. By contrast, YAGO3-10 and NELL-995 exhibit heterogeneous and long-tailed relation distributions, requiring models to generalize from frequent relations to sparsely observed ones. This property aligns with our focus on sample-efficient reasoning under sparse supervision, which constitutes a central motivation of our work.

Scope of LLM-based Agents. Our framework is designed for scenarios in which reasoning re-

quires explicit multi-step decision-making under structural constraints. In datasets where answers are frequently reachable within one hop or can be inferred via surface-level semantic similarity, the computational overhead of LLM-based agents may not translate into clear empirical gains. Therefore, we restrict our evaluation to benchmarks where structured reasoning is expected to play a more prominent role. We leave a systematic study of extending AdaG-Reason to other dataset regimes, including dense or hierarchy-dominated graphs, to future work.

B Semantic-Topological Mismatch Quantification

To quantify semantic–topological mismatch in teacher-generated trajectories, we analyze 5,000 reasoning traces produced by DeepSeek-V3.2 on YAGO3-10. The classification is deterministic and requires no manual labeling. It is based on whether each executed `follow_edge` action corresponds to an edge available in the current KG observation and whether the terminal prediction matches the ground-truth tail. Each trajectory is assigned to one of the following categories:

- **Successful** (207, 4.1%): The trajectory produces the correct target entity through graph-based reasoning.
- **Explicit Topological Violation** (109, 2.2%): At least one executed `follow_edge` action attempts to traverse an edge that is not present in the current KG observation.
- **Implicit Semantic Misalignment** (4,480, 89.6%): All executed `follow_edge` actions correspond to valid KG edges, but the trajectory ultimately produces an incorrect tail prediction.
- **Single-Step Guess** (204, 4.1%): The agent invokes `predict_tail` without graph traversal.

The breakdown reveals a marked asymmetry: explicit violations of graph structure are rare (2.2%), whereas the dominant failure mode consists of trajectories that remain grounded in valid graph edges but terminate at an incorrect entity (89.6% of all analyzed trajectories). This asymmetry motivates IRD (Section 3.2.2): rather than requiring the student to imitate the teacher's full intermediate reasoning text, IRD restricts supervision to grounded, executable actions, reducing the influence of reasoning signals that are not aligned with the final graph decision.

C ARD Resource Allocation Analysis

Figure 5 illustrates how the MAB policy optimizes the synthesis budget on NELL-995. The joint analysis of temporal evolution (Panel A) and final efficiency (Panel B) reveals a strategic transition from correctness to utility via three mechanisms:

Saturation-Aware Throttling. Easy relations are exploited early but throttled upon convergence (Panel A, Red), producing a counter-intuitive inversion where the highest-performing tasks receive minimal final investment (Panel B, Top-Left) to prevent overfitting on trivial patterns.

Dynamic Stratification. Resources shift to the medium-difficulty learnability window (Panel A, Green); the majority of the budget concentrates on these high-gradient tasks (Panel B, Warm Colors), maximizing their contribution to the student model.

Cost-Effective Pruning. Hard relations undergo active suppression (Panel A, Gray) into sustained low-resource exploration (Panel B, Bottom), preventing high-uncertainty failures from diluting data quality while maintaining coverage of long-tail knowledge.

D Implementation Details

D.1 Hardware and Environment

All training experiments were conducted on 8 $\times$ NVIDIA A100 80G GPUs using DeepSpeed ZeRO-3. The software environment includes PyTorch 2.6, CUDA 12.6, and HuggingFace Transformers. For GRPO results, we report the mean and standard deviation over 3 independent runs with different random seeds (42, 123, 456). Due to the non-deterministic nature of the DeepSeek API, teacher-generated trajectories may vary across API calls; we will therefore release all synthesized trajectories alongside our code to facilitate reproducibility.

D.2 Active Reasoning Distillation (ARD)

Hyperparameters. In the composite acquisition function (Eq. 4), we set the coefficients for difficulty and uncertainty as $\lambda_1 = 1$ and $\lambda_2 = 0.3$, giving distribution alignment and difficulty equal emphasis while moderating the uncertainty-driven exploration to prevent unproductive sampling from dominating the acquisition score.

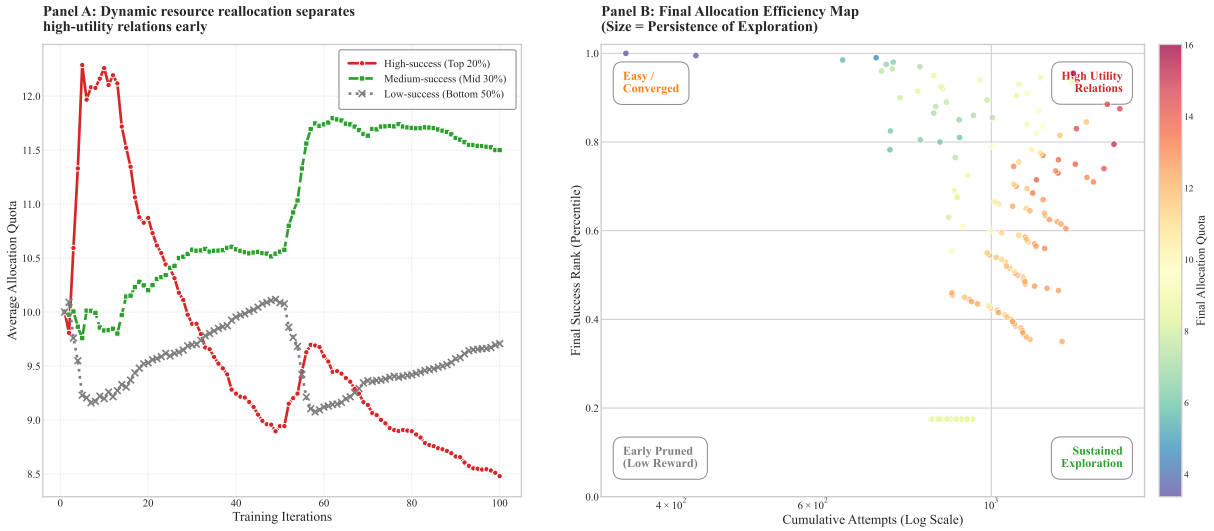


Figure 5: Visualization of Dynamic Resource Allocation in ARD. Panel A traces the temporal evolution of allocation quotas, illustrating the strategic shift from early exploration of simple relations to a focused exploitation of learnable, intermediate-difficulty ones. Panel B maps the final efficiency landscape.

Synthesis Statistics. We employed DeepSeek-V3.2 as the teacher model. The synthesis process details are as follows:

- **YAGO3-10:** We set the target retained-trajectory budget to 5,000. At each ARD iteration, a candidate pool of 1,000 queries was constructed and adaptively allocated across relations. The synthesis process consumed approximately 200,000 teacher API calls before early stopping, yielding 4,498 valid trajectories.
- **NELL-995:** We set the target retained-trajectory budget to 20,000. Each ARD iteration used a candidate pool of 1,000 queries. Due to the increasing concentration of sampling on difficult long-tail relations, the marginal yield of valid trajectories gradually diminished, and the process was terminated early after approximately 800,000 teacher API calls, yielding 12,654 valid trajectories.

D.3 Implicit Reasoning Distillation (IRD)

In the Supervised Fine-Tuning (SFT) phase, we apply **full-parameter fine-tuning** to the student model (Qwen3-8B) using DeepSpeed ZeRO-3. The detailed configuration is:

- **Optimization:** AdamW optimizer with a learning rate of 1×10^{-5} and a linear learning rate scheduler.
- **Regularization:** Weight decay of 0.02.
- **Training Config:** Effective batch size of 32, trained for 3 epochs (YAGO3-10 and NELL-995) or 5 epochs (FB15k-237).

D.4 Step-wise GRPO

For the reinforcement learning stage, we optimize all model parameters (no adapters) under DeepSpeed ZeRO-3. The specific hyperparameters are:

- **Group Size G:** 16 (sampling 16 trajectories per query).
- **KL Coefficient β:** 0.04 (to penalize deviation from the reference policy).
- **Temporal Decay γ:** 0.8 (for step-wise credit assignment).
- **Learning Rate:** Constant 1×10^{-6}.
- **Training Duration:** 10 iterations of the Stage 1–Stage 3 cycle, with 500 queries sampled per iteration.
- **Reward Design:** We use the rank-aware reward and step-wise credit assignment defined in Section 3.3.1.
- **Rollout Sampling:** We use the relation-difficulty-aware sampling distribution defined in Section 3.3.1.

D.5 Cost Analysis

Table 8 summarizes the resource consumption of AdaG-Reason across trajectory synthesis and student training. Teacher API calls denote the cumulative number of model calls made during ARD before early stopping, rather than the target number of retained trajectories or the candidate query pool size.

Table 8: Resource consumption for trajectory synthesis and student training. GPU hours include the full-parameter SFT and GRPO stages.

Resource	YAGO3-10	NELL-995
Teacher API calls	∼200K	∼800K
Retained trajectories	4,498	12,654
SFT GPU hours	5.0	11.5
GRPO GPU hours	160.0	80.0
Total training GPU hours	165.0	91.5

The synthesis cost reflects the iterative nature of ARD: the target retained-trajectory budgets are 5,000 and 20,000 for YAGO3-10 and NELL-995, respectively, while the actual number of teacher calls is determined by adaptive sampling and early stopping. Thus, the low coverage of the retained trajectories should be interpreted as efficiency in the amount of supervision consumed by the student, rather than low end-to-end synthesis cost.

At inference time, Direct decoding requires approximately 1.6 seconds per query, whereas SC@10 requires approximately 5.0 seconds per query, corresponding to about 3.1× the latency. SC@10 incurs no additional training cost and is used only as an optional inference-time strategy to improve candidate recall.

E Prompt Design

The local graph state x_t is serialized into a natural language prompt to guide the LLM agent. Below is the template used for both the Teacher (during ARD) and the Student (during Inference). Note that for the Student model in the *Implicit Reasoning* setting, the output is strictly constrained to the action space tokens.

```
You are an agent that searches a Knowledge Graph
     for a missing tail entity.
Your task: Through multi-step graph lookup,
    decide whether to continue exploring (using
    follow_edge) based on explored paths and
    visible adjacent relationships.
If the current context is sufficient for a final
     decision, execute predict_tail. You may
    predict any entity from the set of already
    explored entities.

You are given the following information:
Query (head, relation): ({e1q_name}, {rq_name})
Current traversal node (et): {et_name}

You observe the following local subgraph (
    context):
{context_text}

Allowed Actions (choose ONE in one answer):
- follow_edge('relation_name','tail_entity_name
    ')
   -> Move one step along a visible edge from
       the current node et to its neighbor.
   -> The relation_name and tail_entity_name
       must be chosen from the current
       observation's neighbors.
- predict_tail(['tail1','tail2', ...])
   -> Finally return an ordered list of tail
       entity names, at most 20 items.

{output_format}

Note:
- Only response a specific Action, no other
    words.
- When using predict_tail, list results most-
    likely-first that appears in the context and
     past paths, max 20.
- When using follow_edge, pick a neighbor that
    appears in the context.
```

Revisiting RAG Retrievers: An Information Theoretic Benchmark

Wenqing Zheng and Dmitri Kalaev and Noah Fatsi and Daniel Barcklow
and Owen Reinert and Igor Melnyk and Senthil Kumar and C. Bayan Bruss
Capital One
wenqing.zheng@capitalone.com

Abstract

Retrieval-Augmented Generation (RAG) systems rely critically on the retriever module to surface relevant context for large language models. Although numerous retrievers have recently been proposed, each built on different ranking principles such as lexical matching, dense embeddings, or graph citations, there remains a lack of systematic understanding of how these mechanisms differ and overlap. Existing benchmarks primarily compare entire RAG pipelines or rely on aggregate ranking metrics (Recall, MRR, nDCG) that assume item-level independence, providing little guidance on redundancy or complementarity between retrievers. This work presents MIGRASCOPE, a Mutual Information based RAG Retriever Analysis Scope. We introduce principled metrics grounded in information theory to quantify retrieval quality, redundancy/synergy, and marginal contribution. Although the framework is designed to be applicable to any setting where a ground-truth answer can construct a pseudo chunk distribution, the present empirical study focuses on multi-hop question answering —a demanding setting where retriever complementarity is especially pronounced. On four corpora and thirteen SOTA retrievers spanning lexical, dense, and graph-based families, we show that (i) the MI-based divergence metric captures retriever quality dimensions not reflected by Recall or MRR, (ii) graph-based retrievers are redundant with each other while BM25 remains informationally complementary, and (iii) in the multi-hop QA setting, a carefully chosen ensemble guided by MI-based attribution can outperform any single retriever. We position these findings as task-specific insights and provide a reusable framework for retriever analysis that practitioners can re-estimate on their own target domains.

1 Introduction

Retrieval-Augmented Generation (RAG) has emerged as a dominant paradigm for knowledge-intensive NLP tasks (Lewis et al., 2020). By retrieving relevant external context to inform large language models (LLMs), RAG systems dramatically improve factual accuracy and reduce hallucination (Shuster et al., 2021). The retriever module is the core component responsible for selecting which evidence is passed to the generator, and its effectiveness directly limits downstream performance. A rich ecosystem of retrievers has flourished, including lexical-based methods such as BM25 (Robertson et al., 2009), dense embedding models trained for semantic similarity (Karpukhin et al., 2020), graph-based retrievers designed for multi-hop and global reasoning (Edge et al., 2024), and increasingly hybrid retrieval systems (Sarmah et al., 2024). Each introduces distinct assumptions and ranking mechanisms, yet the field lacks a unified framework to compare them rigorously. Many existing benchmarks focus on new datasets or evaluation metrics applied to full RAG pipelines. While these studies provide valuable insights into overall system quality, they rarely isolate retrieval performance. Some retriever-specific evaluation work has been done, covering heterogeneous domain generalization (Thakur et al., 2021), massive embedding leaderboards (Muennighoff et al., 2023), and complex-objective tasks (Wang et al., 2024b). However these frameworks focus on aggregate ranking-based metrics such as recall, MRR, and nDCG which assume independence between retrieved items and thus fail to explain how different retrievers complement or supersede each other. Recent ensemble-RAG efforts mostly concentrate on hierarchical integration of the full pipeline without disentangling the unique contributions of the retriever itself. As a result, practitioners have limited principled tools for quantifying which retrievers are strong, which are redundant, and which offer complementary value on a given dataset. We argue that revisiting the retriever module with dedicated evaluation tools is essential to unlocking the next

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 393–408
October 28, 2026 ©2026 Association for Computational Linguistics

phase of RAG progress.

To address these gaps, we introduce MIGRASCOPE, an information-theoretic framework for analyzing and benchmarking retrievers in RAG. We model each retriever's ranking signal as a noisy view of an underlying chunk probability distribution conditioned on ground-truth answers. Building on Mutual Information (MI), we define a retriever quality score that computes the diverging level from the presumed ideal chunk distribution. This score generalizes across lexical, dense, and graph-based retrieval mechanisms, enabling principled comparison and calibration.

We further develop MI-grounded synergy and redundancy metrics to quantify how multiple retrievers interact - whether they contribute complementary evidence or duplicate signals - and provide visual diagnostics to reveal overlap structure. These tools support task-specific marginal contribution estimation and guide retriever analysis and shortlist construction within a target retrieval setting.

Finally, we translate these analyses into practice with a lightweight ensemble that uses MI-based attribution to select and weight retrievers with minimal supervision. Applied across major RAG corpora and a broad suite of state-of-the-art (SOTA) retrievers, the framework yields new insights into contribution patterns, redundancy, and robustness, and the ensemble consistently outperforms strong single retrievers in our evaluated multi-hop QA setting. This paper makes the following contributions:

- Introduce MIGRASCOPE, an MI based retriever quality score that normalizes across ranking paradigms (lexical, dense, graph) to quantify retrieval relevance and utility.
- Define and visualize MI-grounded synergy and redundancy metrics for multiple retrievers, revealing complementary and overlapping signal structure within ensembles.
- Develop a retriever ensemble framework with MI-based attribution to estimate each retriever's marginal contribution within a task-specific retrieval setting.
- Benchmark a broad set of graph-RAG and standard RAG retrievers using the proposed methodology, yielding new insights on contribution patterns, robustness, and design guidance for multi-hop QA RAG systems.

Our results suggest a need to shift community focus from endlessly inventing new retrievers toward understanding how existing ones differ, overlap, and cooperate within a target retrieval environment. Through an information-theoretic lens, we uncover structure and simplicity hidden beneath the diversity of modern retrieval methods, paving the way for more reliable and interpretable RAG systems.

2 Related Work

Our work is positioned at the intersection of three active research areas: RAG benchmarking, RAG ensemble methods, and the application of information-theoretic principles to large language models.

2.1 RAG Benchmarking

Recent work on RAG evaluation has primarily focused on establishing new datasets, defining system-level metrics, or analyzing operational costs, rather than systematically comparing the retrievers themselves. Several benchmarks have been introduced to test RAG systems on complex tasks, such as multi-hop reasoning over structured documents (Khang et al., 2025) or domain-specific GraphRAG applications (Xiao et al., 2025). Others have focused on creating datasets for specific industrial domains, such as finance (Wang et al., 2024a), or on proposing new evaluation methodologies, like assessing LLM robustness and information integration (Chen et al., 2024b) or developing explainable metrics for industrial settings (Friel et al., 2024).

To evaluate retrievers in isolation, the Information Retrieval community has developed rigorous benchmarks covering heterogeneous domain generalization (Thakur et al., 2021), massive embedding leaderboards (Muennighoff et al., 2023), and complex user objectives (Wang et al., 2024b). However, these frameworks typically rely on aggregate ranking metrics such as nDCG and Recall. While effective for static leaderboards, these metrics treat document utility as additive, implicitly assuming independence between retrieved items. Consequently, they fail to quantify redundancy or measure the conditional information gain required to understand how distinct retrievers complement or supersede one another in an ensemble.

While valuable, these contributions often treat the retriever as a component within a larger pipeline. The focus remains on dataset design or system-level assessment. Other studies analyze important, but orthogonal, aspects. For instance, Lin (2024) provide practical guidance on the operational trade-offs of different retrieval infrastructures (e.g., HNSW vs. inverted indexing) but do not offer prescriptive insights on retriever algorithm se-

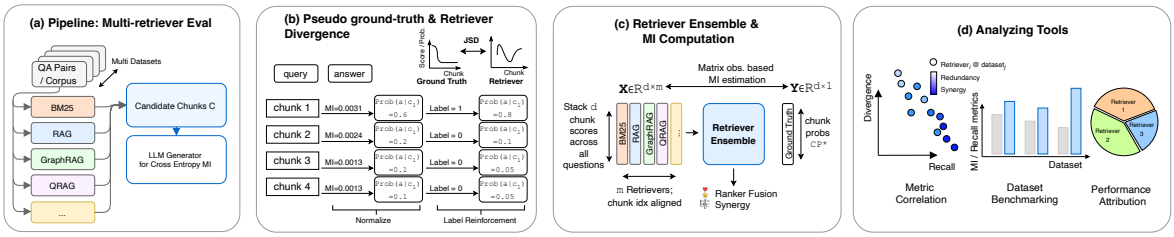


Figure A: The proposed MIGRASCOPE analyzing framework

lection. Similarly, Zhang et al. (2025a) compare different LLMs as dense retrievers, but limit their analysis to this single family.

Across these studies, there remains a significant gap in understanding the fundamental differences, redundancies, and complementary strengths of the various retriever families (e.g., dense, sparse, hybrid, and graph-based). Our work directly addresses this gap by proposing a framework to benchmark the retrievers themselves.

2.2 RAG Ensemble Methods

As RAG systems have matured, ensemble and multi-retriever architectures have become increasingly common. Many of these efforts focus on sophisticated pipeline orchestration. For example, Zhang et al. (2025b) propose a hierarchical system with a planner that orchestrates multiple low-level searchers, while Wu et al. (2025) introduce a modular framework that composes distinct steps like question decomposition and retrieval. Other approaches integrate heterogeneous data sources, such as combining web search with knowledge graphs (Xie et al., 2024) or creating adaptive hybrid systems for specific domains like law (Kalra et al., 2024).

A separate line of work seeks to optimize the information passed to the generator. This includes methods for decoupling retrieval and generation representations (Yang et al., 2025) or explicitly balancing relevance and semantic diversity in the retrieved document set (Rezaei and Dieng, 2025). Chen et al. (2025) explicitly analyze ensembles through an information-theoretic lens, arguing that they increase the total useful information. However, their work stops short of providing an operational recipe for selecting or weighting retrievers. These studies primarily offer architectural solutions or in-domain systems, but do not provide a principled, data-driven method for understanding *which* retrievers to ensemble or *how* to combine them based on their quantified marginal contribution.

2.3 Information-Theoretic Analysis in RAG

Information theory provides a powerful, formal toolkit for quantifying concepts like uncertainty, relevance, and redundancy, which are central to RAG. Recently, these tools have been applied to various parts of the RAG pipeline. For instance, Zhu et al. (2024) use an information bottleneck formulation to filter noise from retrieved contexts. Liu et al. (2024) propose using Pointwise Mutual Information (PMI) as a probabilistic gauge for RAG performance, while others have used divergence metrics to measure attribution (Li et al., 2025) or detect hallucinations (Halperin, 2025). These methods successfully apply information-theoretic concepts to *diagnose* or *optimize* a specific RAG component, such as context pruning (Deng et al., 2025) or detecting data memorization (Anh et al., 2025).

However, this body of work has not yet been directed at the problem of *retriever evaluation*. While Yumuşak (2025) outline a theoretical framework for RAG information flow, it lacks the empirical benchmarking to reveal how different SOTA retrievers actually perform under these metrics. To our knowledge, no existing work uses mutual information and information divergence to create a comparative framework that quantifies the quality, redundancy, and marginal contribution of distinct retriever families.

In summary, the field currently lacks a unified framework for comparing retriever modules directly, a principled method for designing retriever ensembles based on redundancy, and an empirical application of information theory to benchmark retriever contributions. Our work, MIGRASCOPE, is designed to fill these precise gaps.

3 Analytical Framework

3.1 Preliminaries: GraphRAG Formulations

Let a query be $q \in \mathcal{Q}$, a candidate chunk set be $C = \{c_1, \ldots, c_N\}$, and an answer be $a \in \mathcal{A}$. A retriever r maps q to a scored or ranked subset $\mathcal{C}_q \subseteq C$ as the context. The indices of chunks

subset $\mathcal{C}_q$ are denoted as $\mathcal{I}_q$. The retriever gives each chunk a score, combining into a scored list $\{(c_j, s_j)\}_{j\in\mathcal{I}_q}$. A generator then produces the answer a conditioned on $(q, \mathcal{C}_q)$. We formulate a simplified version of RAG and GraphRAGs here. More details are described in Appendix A.

Vanilla RAG. The retrieval $\mathcal{C}_q$ relies solely on vector similarity to select the top-k chunks:

$$\mathcal{C}_q = \mathrm{Top}_k(\{c_j\}_{j=1}^N, \mathbf{Sim}(\mathbf{Embed}(q), \mathbf{Embed}(c_j))) \tag{1}$$

GraphRAG (General Form). GraphRAG introduces a knowledge graph G constructed from the corpus. Retrieval is a two-stage process: graph anchoring and context mapping.

$$G = \mathbf{Construct}(C) \tag{2}$$

$$\mathcal{C}_q = \mathbf{Map}_{G\to C}(\mathbf{Anchor}(G, q)) \tag{3}$$

3.2 Information-Theoretic View of RAG

Mutual information (MI). For random variables X, Y with joint $p(x, y)$ and marginals $p(x), p(y)$,

$$I(X;Y) = \mathbb{E}_{p(x,y)}\left[\log \frac{p(x,y)}{p(x)\,p(y)}\right] = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X). \tag{4}$$

MI measures the reduction in uncertainty of one variable given the other; it is zero iff X and Y are independent.

Divergence. In this work we apply Jensen–Shannon divergence (JSD) which is symmetric, bounded in $[0, \log 2]$, and well-defined even with disjoint supports.

$$\mathrm{KL}(P \parallel Q) = \mathbb{E}_p\left[\log \frac{p}{q}\right],$$
$$\mathrm{JSD}(P, Q) = \tfrac{1}{2}\,\mathrm{KL}\big(P\|M\big) + \tfrac{1}{2}\,\mathrm{KL}\big(Q\|M\big), \quad M = \tfrac{1}{2}(P + Q). \tag{5}$$

Pointwise mutual information in RAG. Given query q and a random multiset of retrieved chunks $\mathcal{C}_q$, define the pointwise mutual information w.r.t. the answer a as

$$\mathrm{PMI}(a; \mathcal{C}_q \mid q) = \log \frac{p(a \mid \mathcal{C}_q, q)}{p(a \mid q)}. \tag{6}$$

Exact evaluation is intractable because the answer a is generative with a combinatorially large support. In practice we approximate with a fixed ground-truth answer a from the dataset:

$$\widehat{\mathrm{PMI}}(a; \mathcal{C}_q \mid q) \approx \log p(a \mid \mathcal{C}_q, q) - \log p(a \mid q). \tag{7}$$

We estimate $\log p(a \mid \cdot)$ with the LLM token-level cross-entropy (log perplexity). For a tokenization $a = (t_1, \ldots, t_T)$ and model condition Z,

$$\log p(a \mid Z) = \sum_{t=1}^{T} \log p_\theta(t_t \mid t_{<t}, Z)$$
$$\iff \quad \mathrm{CE}(a \mid Z) = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(t_t \mid t_{<t}, Z). \tag{8}$$

where θ represents the generative LLM parameters. Large PMI between the answer a and context $\mathcal{C}_q$ corresponds to large perplexity drop under retrieved context $Z = (q, \mathcal{C}_q)$ relative to no context $Z = (q)$. PMI, then, quantifies how much a retrieved set of chunks C_q, for a given question q, reduces uncertainty in the ground-truth answer a.

3.3 Pseudo Ground-Truth Chunk Probability

We define a pseudo ground truth probability over the retrieved chunks that reflects how well each chunk supports the known answer. Related chunk attribution techniques utilize LLM cross-entropy to gauge chunk importance, either by calculating Information Bottleneck scores (Zhu et al., 2024) or by measuring the performance degradation when a specific chunk is removed from the context (Deng et al., 2025). However, such methods often introduce undesirable complexity. Ablation-based approaches suffer from computational overhead and bias induced by chunk order, as LLMs have been shown to disproportionately favor the first and last chunks in a sequence (Liu et al., 2024). Meanwhile, Information Bottleneck methods require tuning sensitive hyperparameters and optimizing auxiliary reconstruction objectives. To avoid these pitfalls, we apply a simple, independent scoring approach to measure the chunk self-attribution score. Thus we have the pseudo ground-truth chunk probability CP^*:

$$\mathrm{CP}^*(c \mid q, a) = \mathrm{Softmax}_{\mathcal{C}_q}(\log p_\theta(a \mid q, c)) \tag{9}$$

where $\mathrm{Softmax}_{\mathcal{C}_q}$ normalizes the chunk probabilities based on the logits over the subset of retrieved chunks $\mathcal{C}_q$.

Golden chunk reinforcement. If the dataset provides golden supporting chunks $G_q \subseteq \mathcal{I}_q$, we optionally reinforce them by a scalar $\gamma > 1$, i.e. amplify the dataset labeled ground truth chunks then re-normalize [1]:

$$\widetilde{\mathrm{CP}}^*(c \mid q, a) = \frac{\mathbf{1}[c \in G_q]\gamma \cdot \mathrm{CP}^*(c \mid q, a) + \mathbf{1}[c \notin G_q] \cdot \mathrm{CP}^*(c \mid q, a)}{\sum_{j \in \mathcal{I}_q} (\mathbf{1}[c_j \in G_q]\gamma + \mathbf{1}[c_j \notin G_q])\mathrm{CP}^*(c_j \mid q, a)}. \tag{10}$$

We use $\widetilde{\mathrm{CP}}^*$ as the default pseudo target distribution to produce the following retriever analysis.

3.4 Retriever Divergence Score

Once we have the pseudo ground-truth chunk probability CP^* as the standard to compare with, we can easily measure the quality of any retriever r by comparing the scores produced by this retriever and CP^*. We first convert the retriever scores $s_r(c)$ to probabilities:

$$p_r(c \mid q) = \frac{\exp(s_r(c)/\tau)}{\sum_{j \in \mathcal{I}_q} \exp(s_r(c_j)/\tau)} \tag{11}$$

where $\tau > 0$ is a temperature-softmax normalization. The quality of the retriever r can be defined by the divergence of r from the pseudo target:

$$\mathrm{Div}(r) \;=\; \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathrm{JSD}\Big(p_r(\cdot \mid q) \;\Big\|\; \widetilde{\mathrm{CP}}^*(\cdot \mid q, a_q)\Big). \tag{12}$$

3.5 Ensembling Multiple Retrievers

Assume m retrievers $\{r_i\}_{i=1}^m$ produce scores $\{s_i(c)\}_{i=1}^m$. There are multiple ways that these retrievers can be fused. We list and compare 12 ways to merge the rankers and show results in Section 4. We do not aim to claim novelty on the ranker fusion approaches; we leverage existing approaches as detailed in Appendix B.1.

3.6 Marginal Contributions and Redundancy Analysis

In the attempt to benchmark and analyze different retrievers, we utilize well-established tools from statistical signal processing to quantify the contribution and redundancy levels.

Setup. Given a batch of queries $\{q_i\}_{i=1}^{|\mathcal{Q}|}$, we stack the retriever's scores and CP^* to obtain observation X and estimation target Y.

For each query, we use $\widetilde{\mathrm{CP}}^*$ to produce a list of top K chunk scores, and concatenate them into target vector $Y \in \mathbb{R}^{K|\mathcal{Q}|}$

[1] When golden chunk support exists, the chunk distribution is ideally one-hot. The CP^* softens the hard distribution. For better alignment with dataset labels, we empirically pick large γ.

On the other hand, each of the m retrievers also produces scores for these top K chunks. We stack them into the observation matrix $X \in \mathbb{R}^{K|\mathcal{Q}| \times m}$.

Utility. Define the set function

$$F(S) = I\big(Y; X_S\big), \tag{13}$$

which measures predictive information about Y contained in the subset S of retrievers. This is distinct from Fisher information.

Independent marginal contribution. Denote the collection of m retrievers as $\mathcal{R}$. For retriever i, its independent marginal contribution is:

$$\begin{aligned} C_i \;&\stackrel{\text{def}}{=}\; I\big(Y; X_i \mid X_{\mathcal{R}/\{i\}}\big) \\ &=\; H\big(Y \mid X_{\mathcal{R}/\{i\}}\big) - H\big(Y \mid X_{\mathcal{R}}\big), \end{aligned} \tag{14}$$

where $X_{\mathcal{R}/\{i\}}$ denotes all columns of X excluding the column corresponding to retriever i, $X_{\mathcal{R}}$ denotes the full observation matrix. Large C_i then captures the unique information conveyed by retriever i. Values near zero indicate redundancy given the others.

Shapley value for fair attribution. Define Shapley values over the coalition game $(\{1, \ldots, m\}, F)$:

$$\phi_i = \sum_{S \subseteq \{1,\ldots,m\} \setminus \{i\}} \frac{|S|!\,(m - |S| - 1)!}{m!} \left[F(S \cup \{i\}) - F(S)\right]. \tag{15}$$

Shapley values fairly allocate the total utility $F(\{1{:}m\}) - F(\varnothing)$ across retrievers by $\Sigma_{i=1}^m \phi_i = F(\{1{:}m\}) - F(\varnothing)$. They average marginal information gains over all insertion orders, which resolves order dependence.

Redundancy vs synergy. For any pair (i, j), define the interaction information

$$\mathrm{II}(Y; X_i; X_j) \;=\; I(Y; X_i) - I\big(Y; X_i \mid X_j\big). \tag{16}$$

Positive II indicates redundancy; negative II indicates synergy, i.e., complementary information that emerges only jointly. Accordingly, we define a surrogate distance between retrievers r_i and r_j as

$$d(r_i, r_j) = \exp\big(-\mathrm{II}(Y; X_i; X_j)\big). \tag{17}$$

We further leverage this score to position each retriever on a *redundancy–synergy spectrum*, and perform clustering analysis to reveal the structural grouping of retrievers. Results are presented in Section 4.6.

Retriever Selection Principle. We apply the tractable redundancy/synergy scores in Eq. (16) to decide whether to include certain RAG retriever for ensemble. We follow the principle that multicollinearity harms estimation stability but independence helps. For any sources X_1, X_2 and target Y, $I(Y; X_1, X_2) = I(Y; X_1) + I(Y; X_2|X_1)$. If X_2 is redundant given X_1, then $I(Y; X_2|X_1) = 0$. If X_2 is complementary, this conditional term is positive, so the pair contains strictly more information than either alone.

3.7 Estimating Entropy and Mutual Information from Vector Observations

Assuming X and Y are jointly Gaussian yields a closed-form mutual information (MI) estimator (see Appendix D.1). However, this assumption is theoretically tenuous in our setting and, empirically, leads to systematic underestimation of MI by failing to capture nonlinear dependencies between the variables.

Therefore, we accept the distributions of X and Y as unknown, then train a predictive model $\hat{p}(y \mid x)$ and compute $\widehat{H}(Y \mid X) = \frac{1}{n}\sum_i H(\hat{p}(\cdot \mid x_i))$ for discrete Y, or approximate continuous Y with a Gaussian residual model where $\widehat{H}(Y \mid X) = \frac{1}{2}\log(2\pi e\,\widehat{\sigma}^2_{\text{res}})$. We use XGB for regression.

4 Results

4.1 Benchmarking RAG Retrievers

In this work, we focus on the effect of retriever mechanisms rather than encoder, hence the encoder remains fixed and same for all RAG approaches in our work. We use BGE-M3 (Chen et al., 2024a) indiscriminatively. We benchmark a dozen of SOTA RAG retrievers, including chunk similarity based (vanilla) retriever, lexical retrievers, graph retrievers, and chunk decomposition based retrievers, etc. For the seven G-series graph retrievers, the encoder and indexing backbone are shared, while the graph construction differs: windowed co-occurrence, thresholded co-occurrence, hierarchical provenance, naive co-occurrence, community, relationship-global, and global co-occurrence. We additionally compare with HippoRAG (Gutiérrez et al., 2025) and LightRAG (Guo et al., 2024).

4.2 Datasets and Split

We evaluate the retrievers on four multi-hop QA corpora: HotpotQA, MuSiQue, 2WikiMultiHopQA, TriviaQA. For each corpus, we uniformly sample 1,000 QA pairs.

Benchmarking results for these retrievers are presented in Table 1, which shows the traditional metrics (Recall@1, MRR) as well as the newly derived divergence metric from Eq. (12). Next, we leverage the proposed tools to address four key research questions.

- RQ1: How does the new MI based metric align with previous MRR/Recall metrics?
- RQ2: How does the selection of hyperparameters impact MI?
- RQ3: Is it possible to combine multiple weak retrievers into a stronger retriever, and how are they contributing individually?
- RQ4: Among the SOTA retrievers, how redundant are they?

More results are presented in Appendix C, including variances, computational costs, ranking robustness, case study of MI/Recall metric differences, MIGRASCOPE analyzing workflow, etc.

4.3 Correlation Analysis (RQ1)

This experiment aims to check the level of alignment between the proposed divergence metric and traditional metrics. For each (retriever, dataset) pair, we plot the divergence score against standard evaluation metrics (Recall@1, MRR) in Figure 1.

The first observation is that recall and *Div* scores are negatively correlated in general (strong retrievers tend to have lower divergence). There are also systematic deviations, as multiple retrievers achieve similar recall but differ significantly in divergence. This confirms the complementary nature of our divergence metric and its ability to diagnose failure modes not captured by ranking accuracy alone.

These deviations also affect practical selection decisions. On 2Wiki, rank-based metrics among strong graph retrievers can be contradictory, while *Div* selects the retriever that later contributes more strongly in the best ensemble. On MuSiQue, several graph variants are marginally stronger than G-hierarchical under Recall/MRR, yet *Div* identifies G-hierarchical as the better ensemble component.

4.4 Sensitivity of MI Hyperparameters (RQ2)

We sweep top-K, anchor construction, and golden-chunk reinforcement strength γ; the full plot is deferred to Appendix C.3 to conserve space. The

Method	2Wiki			HotpotQA			MuSiQue			TriviaQA		
	Rec	MRR	Div	Rec	MRR	Div	Rec	MRR	Div	Rec	MRR	Div
G-window	0.935	0.959	0.110	0.871	0.916	0.118	0.822	0.806	0.125	0.891	0.922	0.102
G-threshold	0.936	0.960	0.120	0.921	0.924	0.121	0.725	0.805	0.122	0.944	0.922	0.101
G-hierarchical	0.934	0.958	0.117	0.872	0.916	0.120	0.724	0.803	0.129	0.881	0.914	0.098
G-naive	0.957	0.934	0.122	0.876	0.920	0.125	0.729	0.808	0.134	0.892	0.933	0.144
G-community	0.935	0.959	0.125	0.873	0.917	0.123	0.727	0.807	0.130	0.891	0.919	0.119
G-relationship	0.934	0.958	0.128	0.873	0.917	0.124	0.728	0.807	0.129	0.845	0.922	0.127
G-global	0.937	0.961	0.125	0.880	0.922	0.128	0.726	0.805	0.131	0.835	0.955	0.121
HippoRAG	0.931	0.960	0.142	0.892	0.921	0.128	0.833	0.810	0.120	0.921	0.920	0.119
RAG	0.911	0.921	0.133	0.851	0.889	0.130	0.711	0.768	0.159	0.860	0.891	0.125
LLM	0.913	0.944	0.131	0.824	0.908	0.138	0.807	0.801	0.155	0.882	0.914	0.129
QRAG	0.889	0.919	0.147	0.713	0.792	0.133	0.681	0.753	0.166	0.829	0.881	0.144
LightRAG	0.785	0.863	0.155	0.507	0.644	0.162	0.563	0.659	0.199	0.877	0.820	0.157
BM25	0.486	0.605	0.252	0.620	0.739	0.238	0.415	0.507	0.302	0.420	0.569	0.289

Table 1: RAG Retriever Benchmarking with MI metric

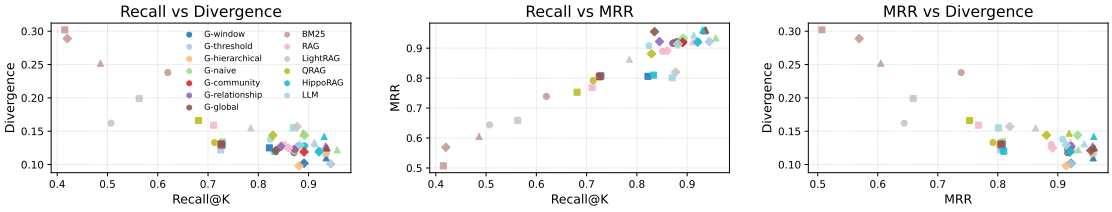


Figure 1: Metric Correlations Across Datasets, Legend by Retriever

same qualitative pattern remains: larger γ makes *Div* more recall-like, while larger top-K reduces Pearson alignment because retrievers increasingly differ from the pseudo target away from the top evidence. Importantly, retriever ordering is more stable than raw Pearson values alone suggest. On HotpotQA, the Spearman correlation of *Div*-induced retriever rankings across the sweep stays within $[0.748, 0.910]$, with per-setting means between 0.841 and 0.852 (Table 4). This supports using *Div* primarily as a relative diagnostic for model selection rather than an absolute standalone score.

4.5 Retriever Ensemble and Shapley Contribution Values (RQ3)

We conduct a series of experiments in the attempt to find a subset of combined retrievers outperforming any single strongest retriever, as well as visualizing the synergy across different retrievers.

Our retriever ensemble is based on the divergence metric. For every given dataset, we first run retrieval procedure with every retriever to get a large top-K chunk list for 1000 QA pairs. Then we select a list of retriever subset to perform ensemble search in the unified search space. The best fusion rule is selected on the **20% training split** by maximizing held-out MI on a nested 5-fold cross-validation within that split; no test labels are touched during fusion-rule selection. The ensembled retriever is then evaluated on the **remaining 80% test QA pairs**. Operationally, this also yields a practitioner-facing shortlist procedure: score retrievers individually by *Div*, remove dominated or highly redundant candidates, add retrievers greedily by marginal MI/Shapley gain, and tune only a small family of fusion rules on the resulting shortlist (Appendix C.5).

We break down this research question into several key points for a better comprehensive understanding. **RQ3.1** How much are these retrievers contributing to the ensembled results? **RQ3.2** How much information does the best ensemble strategy provide? What if it deviates from the best configuration? **RQ3.3** Ensemble stability: if more retrievers are added to the best ensemble configuration, or some retrievers are dropped, how does it impact the performance?

4.5.1 Best Recall by Shapley Shares (RQ3.1)

We compute the Shapley attribution values for every retriever within the best retriever set in Figure 3.1. The height of the bar plot means the overall recall performance relative to 1.0. From this figure we can see that the strongest retriever also tends to contribute the most. However, the winner is not the ensemble of the top-performing GraphRAG retrievers, but it often includes worse retrievers such as BM25. This hints that different GraphRAG retriev-

ers may contain redundant components that harm synergy.

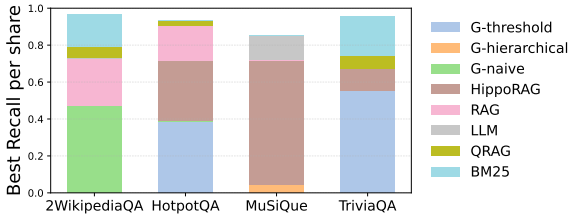


Figure 3.1: Shapley share for the best ensemble configs. Variances see Appendix C.1.

4.5.2 Perturbation Impact on MI (RQ3.2)

We evaluate RQ3.2 by computing $I(\widetilde{\mathrm{CP}}^*; X_S)$ from Eq. 13 for the best retriever subset and its perturbed variations, where S is some retriever ensemble set. The results are plotted in Figure 3.3. In this figure, the y-axis labels are different retriever subsets, ranked by the MI. As can be seen in the figure, the few top performing retriever subsets show similar performance. However, when the perturbation of retrievers are too large, the performance drops drastically.

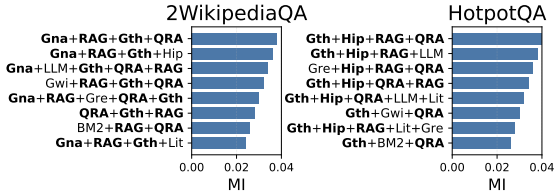


Figure 3.3: MI under ensemble perturbation.

4.5.3 Perturbation Impact on Recall (RQ3.3)

We answer RQ3.3 by first identifying the best retriever subset for ensemble, then gradually add or drop retrievers on top of this best subset, and finally concatenate all results into a continuous line. We plot the results in Figure 3.2.

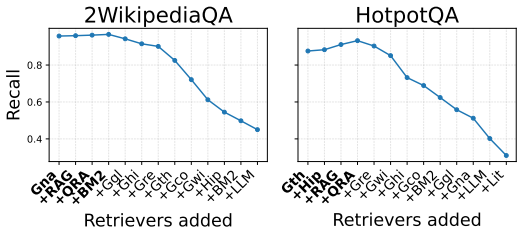


Figure 3.2: Recall under ensemble perturbation.

As can be seen in Figure 3.2, if the retriever subset is chosen wisely, it performs better than single best retriever. As more retrievers are added, redundancy dominates over synergy, hence the performance dropped drastically.

4.6 Retriever Redundancy Spectrum (RQ4)

We formulate the visualization of the m retrievers as a geometric embedding problem. We first pre-compute the pairwise redundancy for all retrievers. By interpreting the pairwise redundancy as a measure of similarity, we derive a distance metric where highly redundant pairs correspond to short distances as in Eq. 17. Using Classical Multidimensional Scaling (MDS), we map the high-dimensional relationship matrix into a 2D Euclidean space. This mapping preserves the global relational structure, allowing us to visualize retriever clusters based on their informational synergy, as shown in Figure 4.

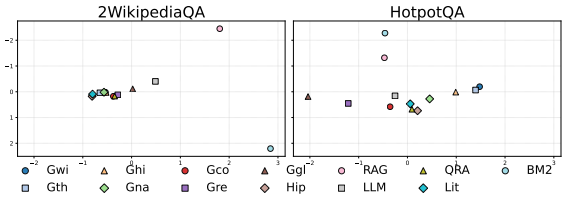


Figure 4: Retriever redundancy/synergy spectrum.

As can be observed in the figure, GraphRAG approaches are similar to each other and further away from vanilla RAG or lexical BM25 retriever. This implies that redundancy might be clustered by approach types (graph vs. lexical vs. vanilla).

5 Conclusion

We introduced MIGRASCOPE, an information-theoretic framework for evaluating RAG retrievers with principled metrics for quality, redundancy, and marginal contribution. By modeling rankings as noisy views of an answer-conditioned chunk distribution, it provides a task-specific way to measure retriever quality, redundancy, and marginal utility, and shows on multi-hop QA benchmarks that well-designed ensembles can outperform individual retrievers. Despite current limitations, MIGRASCOPE offers new insight into retriever interactions and practical tools for building more robust and efficient RAG systems. Future work should integrate generation-side metrics, study retriever-specific internal mechanisms, improve computational efficiency for large-scale deployment, and evaluate the framework beyond multi-hop QA to test how redundancy and complementarity vary across domains.

Limitations

As a pioneering effort in applying mutual information to retriever evaluation, MIGRASCOPE has certain limitations that suggest opportunities for refinement. The MI-based metrics, while theoretically grounded, are less familiar than conventional metrics like Recall@K or MRR, and their interpretation may require additional context for practitioners. The framework depends on hyperparameters and estimator choices, which can influence numerical stability and relative rankings, though sensitivity analysis can mitigate these effects. Additionally, the reported MI values are approximations influenced by model assumptions, estimator errors, and practical truncations, and should be viewed as consistent estimates under specific settings. The present instantiation is also chunk-local: $\widetilde{\mathrm{CP}}^{*}$ scores each chunk independently, so it does not explicitly model evidence that is useful only jointly as a chunk pair or set. Finally, our empirical study is limited to multi-hop QA, so the redundancy and complementarity structures reported here should be interpreted as task-specific and re-estimated for new domains rather than assumed to transfer unchanged; accordingly, the current results should not be read as universal retriever-selection rules across RAG settings.

Additionally, we do not compare against ensemble-diversity metrics from the classifier ensemble literature (e.g., Q-statistic, disagreement measure) because these are defined for binary prediction settings and do not generalize to graded ranked retrieval lists; the conceptual mismatch would make any such comparison misleading. Similarly, while we evaluate 12 ranker fusion strategies and identify the best per dataset, we do not claim novelty over any individual fusion rule. The contribution is the *selection criterion* (MI-based attribution) rather than the fusion mechanism itself. The standard RRF baseline (Cormack et al., 2009) is included among the 12 strategies and consistently underperforms the MI-guided weighted fusion, supporting the value of principled retriever selection over naïve rank aggregation.

References

Le Vu Anh, Nguyen Viet Anh, Mehmet Dik, and Luong Van Nghia. 2025. Repcs: Diagnosing data memorization in llm-powered retrieval-augmented generation. *arXiv preprint arXiv:2506.15513*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024b. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Yifei Chen, Guanting Dong, Yutao Zhu, and Zhicheng Dou. 2025. Revisiting rag ensemble: A theoretical and mechanistic analysis of multi-rag system collaboration. *arXiv preprint arXiv:2508.13828*.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.

Jiale Deng, Yanyan Shen, Ziyuan Pei, Youmin Chen, and Linpeng Huang. 2025. Influence guided context selection for effective retrieval-augmented generation. *arXiv preprint arXiv:2509.21359*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2024. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Igor Halperin. 2025. Prompt-response semantic divergence metrics for faithfulness hallucination and misalignment detection in large language models. *arXiv preprint arXiv:2508.10192*.

Rishi Kalra, Zekun Wu, Ayesha Gulley, Airlie Hilliard, Xin Guan, Adriano Koshiyama, and Philip Treleaven. 2024. Hypa-rag: A hybrid parameter adaptive retrieval-augmented generation system for ai legal and policy applications. *arXiv preprint arXiv:2409.09046*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi

Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Minsoo Khang, Sangjun Park, Teakgyu Hong, and Dawoon Jung. 2025. Crest: A comprehensive benchmark for retrieval-augmented generation with complex reasoning over structured documents. *arXiv preprint arXiv:2505.17503*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Ruizhe Li, Chen Chen, Yuchen Hu, Yanjun Gao, Xi Wang, and Emine Yilmaz. 2025. Attributing response to context: A jensen-shannon divergence driven mechanistic study of context attribution in retrieval-augmented generation. *arXiv preprint arXiv:2505.16415*.

Jimmy Lin. 2024. Operational advice for dense and sparse retrievers: Hnsw, flat, or inverted indexes? *arXiv preprint arXiv:2409.06464*.

Tianyu Liu, Jirui Qi, Paul He, Arianna Bisazza, Mrinmaya Sachan, and Ryan Cotterell. 2024. Pointwise mutual information as a performance gauge for retrieval-augmented generation. *arXiv preprint arXiv:2411.07773*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.

Mohammad Reza Rezaei and Adji Bousso Dieng. 2025. Vendi-rag: Adaptively trading-off diversity and quality significantly improves retrieval augmented generation with llms. *arXiv preprint arXiv:2502.11228*.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. 2024. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 608–616.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Shuting Wang, Jiejun Tan, Zhicheng Dou, and Ji-Rong Wen. 2024a. Omnieval: An omnidirectional and automatic rag evaluation benchmark in financial domain. *arXiv preprint arXiv:2412.13018*.

Xiaoyue Wang, Jianyou Wang, Weili Cao, Kaicheng Wang, Ramamohan Paturi, and Leon Bergen. 2024b. Birco: A benchmark of information retrieval tasks with complex objectives. *arXiv preprint arXiv:2402.14151*.

Ruofan Wu, Youngwon Lee, Fan Shu, Danmei Xu, Seung-won Hwang, Zhewei Yao, Yuxiong He, and Feng Yan. 2025. Composerag: A modular and composable rag for corpus-grounded multi-hop question answering. *arXiv preprint arXiv:2506.00232*.

Yilin Xiao, Junnan Dong, Chuang Zhou, Su Dong, Qianwen Zhang, Di Yin, Xing Sun, and Xiao Huang. 2025. Graphrag-bench: Challenging domain-specific reasoning for evaluating graph retrieval-augmented generation. *arXiv preprint arXiv:2506.02404*.

Weijian Xie, Xuefeng Liang, Yuhui Liu, Kaihua Ni, Hong Cheng, and Zetian Hu. 2024. Weknow-rag: An adaptive approach for retrieval-augmented generation integrating web search and knowledge graphs. *arXiv preprint arXiv:2408.07611*.

Peiru Yang, Xintian Li, Zhiyang Hu, Jiapeng Wang, Jinhua Yin, Huili Wang, Lizhi He, Shuai Yang, Shangguang Wang, Yongfeng Huang, and 1 others. 2025. Heterag: A heterogeneous retrieval-augmented generation framework with decoupled knowledge representations. *arXiv preprint arXiv:2504.10529*.

Semih Yumuşak. 2025. An information-theoretic framework for retrieval-augmented generation systems. *Electronics*, 14(15):2925.

Hengran Zhang, Keping Bi, and Jiafeng Guo. 2025a. A comparative study of specialized llms as dense retrievers. *arXiv preprint arXiv:2507.03958*.

Zhuocheng Zhang, Yang Feng, and Min Zhang. 2025b. Levelrag: Enhancing retrieval-augmented generation with multi-hop logic planning over rewriting augmented searchers. *arXiv preprint arXiv:2502.18139*.

Kun Zhu, Xiaocheng Feng, Xiyuan Du, Yuxuan Gu, Weijiang Yu, Haotian Wang, Qianglong Chen, Zheng Chu, Jingchang Chen, and Bing Qin. 2024. An information bottleneck perspective for effective noise filtering on retrieval-augmented generation. *arXiv preprint arXiv:2406.01549*.

A Detailed RAG/GraphRAG Formulations

This section formalizes the graph construction and retrieval variants used in our evaluation, emphasizing succinct definitions and avoiding redundant statements.

A.1 Method 1: Windowed Co-occurrence Graph

The **G-Window** method limits co-occurrence edges to entities within a fixed offset in the ordered entity list $\mathcal{E}_c$ for any chunk c.

$$E_{\text{Window}} = \Big\{ \{e_i, e_j\} \mid e_i, e_j \in \mathcal{E},\ i \neq j,\ \exists c \in C, \exists k, l : (\mathcal{E}_{c,k} = e_i \wedge \mathcal{E}_{c,l} = e_j \wedge |k - l| \leq \text{window\_size}) \Big\} \quad (18)$$

A.2 Method 2: Thresholded Co-occurrence Graph

The **G-Threshold** method filters incidental co-occurrences by requiring a minimum count across the corpus:

$$E_{\text{Threshold}} = \{ \{e_i, e_j\} \mid e_i, e_j \in \mathcal{E}, i \neq j, \sum_{c \in C} \mathbf{I}(\{e_i, e_j\} \subseteq \mathcal{E}_c) \geq \text{min\_cooc} \} \quad (19)$$

A.3 Method 3: Hierarchical and Algorithmic Graph Construction Methods

G-Hierarchical constructs a heterogeneous directed graph $G = (V, E)$ with document, chunk, and entity nodes, enabling explicit provenance.

- $(\text{doc} : d, \text{chunk} : c)$ with type: CONTAINS.
- $(\text{chunk} : c, \text{entity} : e)$ with type: MENTIONS for $e \in \mathcal{E}_c$.
- undirected $\{e_i, e_j\}$ with type: CO_OCCURS for $e_i, e_j \in \mathcal{E}_c$.

A.4 Method 4: Naive Co-occurrence Graph

G-Naive establishes an undirected edge between any two entities that co-occur in at least one chunk:

$$E_{\text{Naive}} = \{ \{e_i, e_j\} \mid e_i, e_j \in \mathcal{E},\ i \neq j,\ \exists c \in C : \{e_i, e_j\} \subseteq \mathcal{E}_c \} \quad (20)$$

A.5 Method 5: Community Graph

G-Community first forms $G_{\text{temp}} = (\mathcal{E}, E_{\text{Threshold}})$, then applies Louvain to obtain clusters $\mathcal{S}_k$:

$$\{\mathcal{S}_1, \mathcal{S}_2, \dots\} = \textbf{Louvain}(G_{\text{temp}}, \text{weight: cooccurrence}) \quad (21)$$

Community nodes v_k are added with edges $E_{\text{Comm}} = \{(e, v_k) \mid e \in \mathcal{S}_k, \text{type: BELONGS\_TO}\}$.

A.6 Method 6: Relationship Graph

G-Relationship-Global focuses on relationships among significant entities $\mathcal{E}_{\text{sig}}$ (frequency $\geq$ MIN_ENTITY_OCCURRENCE), aggregating their global context:

$$C_e = \{c \in C \mid e \in \mathcal{E}_c\} \quad (22)$$

Directed relationships are extracted by an LLM over Context(C_e):

$$E_{\text{RelGlobal}} = \bigcup_{e \in \mathcal{E}_{\text{sig}}} \mathbf{LLM}_{\text{extract\_global}}(\text{Context}(C_e), e, \mathcal{E}_{\text{cand}}) \quad (23)$$

A.7 Method 7: Global Co-occurrence Graph

G-Global derives the corpus-level unique set of co-occurring entity pairs (i.e., the non-attributed, de-duplicated union of chunk-level co-occurrences; equivalent in edge set to Method 4 but without per-chunk metadata).

A.8 Method 8: HippoRAG

HippoRAG (Gutiérrez et al., 2025) models a heterogeneous memory graph $G_{\text{mem}} = (V, E)$:

1. **Graph Construction:** $V = \mathcal{E} \cup \mathcal{F} \cup C$, where $\mathcal{F}$ are OpenIE triples. Edges include containment and synonymy (via $\mathbf{Embed}$ similarity).
2. **Fact Reranking:** Retrieve fact candidates $\mathcal{F}_{\text{cand}} = \mathbf{VDB}_{\mathcal{F}}(q)$ and refine with an LLM reranker:
$$\mathcal{F}_q = \mathbf{LLM}_{\text{Rerank}}(\mathcal{F}_{\text{cand}}, q) \quad (24)$$
3. **PPR Initialization:** Extract query entities from $\mathcal{F}_q$ and form $\mathbf{p}_0$:
$$\mathbf{p}_0 = \mathbf{InitDistribution}(\mathbf{Map}_{\mathcal{F} \to \mathcal{E}}(\mathcal{F}_q)) \quad (25)$$
4. **PPR Graph Search:** Run PPR on G_{mem}:
$$\mathbf{PPR}_{\text{scores}} = \mathbf{PPR}(G_{\text{mem}}, \mathbf{p}_0) \quad (26)$$
5. **Context Selection:** Select top-scoring chunks:
$$\mathcal{C}_q = \text{Top}_{k_C}(\{c_j\}_{j \in C}, \mathbf{PPR}_{\text{scores}}[c_j]) \quad (27)$$

A.9 Method 9: RAG

Standard RAG using embedding-based retrieval over chunks followed by generation.

A.10 Method 10: LLM Reranking

LLM reranking orders candidate chunks $\mathcal{C}_{\text{cand}}$ with the LLM:

$$R_{\mathbf{LLM}} = \mathbf{LLM}_{\text{rank}}(q, \mathcal{C}_{\text{cand}}) \tag{28}$$

The final context $\mathcal{C}_q$ is the top k_C by $R_{\mathbf{LLM}}$. In our tests, candidates were produced via the windowed co-occurrence graph, but the reranker is graph-agnostic.

A.11 Method 11: Question as Sub-chunk Summary (QRAG)

Graph Construction (Synthetic Questions). For each chunk c_i, generate a set of synthetic questions:

$$\mathbf{Q}_i = \{q_{i,1}, \ldots, q_{i,M_i}\} = \mathbf{LLM}_{\text{Gen}}(c_i) \tag{29}$$

Retrieval Scoring. Score c_i by the maximum similarity between the query and its synthetic questions:

$$s_i = \max_{q_{i,j} \in \mathbf{Q}_i} \mathbf{Sim}(\mathbf{Embed}(q), \mathbf{Embed}(q_{i,j})) \tag{30}$$

Final Context. Select the top k chunks:

$$\mathcal{C}_q = \text{Top}_k(\{c_i\}_{i=1}^N, s_i) \tag{31}$$

A.12 Method 12: LightRAG

Graph Construction. Build a directed, labeled graph $G_{\text{LightRAG}} = (\hat{\mathcal{E}}, \hat{\mathcal{R}})$ from LLM-extracted triples, applying deduplication and refinement:

$$G_{\text{LightRAG}} = \mathbf{Dedupe} \circ \mathbf{Prof}(\mathcal{V}_{\text{init}}, \mathcal{E}_{\text{init}}) \tag{32}$$

Retrieval.

1. **Search & Neighborhood Expansion:** Retrieve $\mathcal{E}_{\text{sim}} = \mathbf{VDB}_{\mathcal{E}}(q)$, $\mathcal{R}_{\text{sim}} = \mathbf{VDB}_{\mathcal{R}}(q)$, then expand to $\mathcal{E}_{\text{hood}}$, $\mathcal{R}_{\text{hood}}$ via graph traversal.
2. **Token Truncation:** Filter and truncate to a token budget $\mathbf{TB}$:

$$\mathcal{E}_{\text{final}}, \mathcal{R}_{\text{final}} = \mathbf{Truncate}(\mathbf{Filter}(\mathcal{E}_{\text{sim}} \cup \mathcal{E}_{\text{hood}}, \mathcal{R}_{\text{sim}} \cup \mathcal{R}_{\text{hood}}), \mathbf{TB}) \tag{33}$$

3. **Chunk Merging:** Map $\mathcal{E}_{\text{final}}, \mathcal{R}_{\text{final}}$ back to source chunks and merge.
4. **Context Formatting:** Produce the final context:

$$\mathcal{C}_q = \mathbf{Format}(\mathcal{E}_{\text{final}}, \mathcal{R}_{\text{final}}, \mathbf{Merge}_{\mathcal{ER} \to C}(\mathcal{E}_{\text{final}}, \mathcal{R}_{\text{final}})) \tag{34}$$

A.13 Method 13: BM25

Unmodified lexical BM25 retrieval.

B Score normalization and temperature selection.

Because different retrievers produce scores on incompatible scales—BM25 outputs unbounded term-frequency weighted integers, cosine-similarity retrievers output values in $[-1, 1]$, and PPR-based retrievers (e.g., HippoRAG) output stationary distribution probabilities—we apply per-retriever z-score normalization before the temperature softmax:

$$\tilde{s}_r(c) = \frac{s_r(c) - \mu_r}{\sigma_r + \epsilon}, \quad p_r(c \mid q) = \text{softmax}_{\mathcal{C}_q}(\tilde{s}_r(c)/\tau), \tag{35}$$

where μ_r and σ_r are the per-retriever mean and standard deviation of scores over all retrieved chunks for that query, and $\epsilon = 10^{-6}$ avoids division by zero. The temperature τ is set to a single shared value $\tau = 1.0$ after normalization; because z-score normalization equalizes score variance across retrievers, this shared τ does not introduce a retriever-specific bias.

The temperature τ and z-score normalization are similarly stable: varying $\tau \in \{0.5, 1.0, 2.0\}$ after normalization changes Spearman retriever rankings by less than 0.05, confirming that Div functions as a robust relative diagnostic rather than an absolute scale-sensitive score.

B.1 Ranker Fusion Details

We formulate the ranker fusion search space as follows.

Score standardization. We apply per-retriever z-score normalization

$$\tilde{s}_i(c) = \frac{s_i(c) - \mu_i}{\sigma_i + \epsilon}, \quad \mu_i = \text{mean}_c\, s_i(c), \quad \sigma_i = \text{std}_c\, s_i(c). \tag{36}$$

Weighted linear fusion.

$$\hat{s}(c) = \sum_{i=1}^{m} w_i\, \tilde{s}_i(c), \qquad w_i \geq 0,\ \sum_i w_i = 1. \tag{37}$$

Weights can be set from divergence $w_i \propto \exp(-\lambda\, \text{Div}(r_i))$, or learned by minimizing a surrogate loss such as cross-entropy to $\widetilde{\text{CP}}^*$:

$$\min_w \sum_q \text{CE}\Big(\widetilde{\text{CP}}^*(\cdot \mid q, a_q),\ \text{softmax}(\hat{s}(\cdot)/\tau)\Big). \tag{38}$$

Temperature fusion of probabilities. Convert each retriever to a calibrated distribution

$$p_i(c \mid q) = \mathrm{softmax}\left(\frac{\tilde{s}_i(c)}{\tau_i}\right), \qquad \hat{p}(c \mid q) = \frac{\prod_{i=1}^{m} p_i(c \mid q)^{w_i}}{\sum_j \prod_{i=1}^{m} p_i(c_j \mid q)^{w_i}} \quad (39)$$

which is a log-opinion pool.

Z-score fusion

$$\phi_r(c) = \frac{s_r(c) - \mu_r}{\sigma_r + \epsilon}, \qquad S(c) = \sum_{r=1}^{m} w_r \, \phi_r(c) \quad (40)$$

Z-score fusion (40) aligns scales using per-list mean μ_r and std σ_r.

Logit pooling

$$S(c) = \sum_{r=1}^{m} w_r \, \mathrm{logit}\big(\hat{p}_r(c)\big), \qquad \mathrm{logit}(p) = \log \frac{p}{1-p} \quad (41)$$

Logit pooling (41) adds log-odds, equivalent to naive Bayesian evidence combination under independence.

Noisy-or

$$\hat{p}(c) = 1 - \prod_{r=1}^{m} \big(1 - \hat{p}_r(c)\big)^{w_r} \quad (42)$$

Noisy-or (42) estimates the probability that at least one retriever deems c relevant.

Adaptive weights from divergence.

$$w_r(q, a) = \frac{\big(\epsilon + \mathrm{Div}(r \mid q, a)\big)^{-1}}{\sum_{j=1}^{m} \big(\epsilon + \mathrm{Div}(j \mid q, a)\big)^{-1}} \quad (43)$$

Inverse-divergence weights (43) emphasize better-aligned retrievers.

Reciprocal Rank Fusion (RRF). Let $\mathrm{rank}_i(c)$ be the rank of c under r_i, then

$$\hat{s}(c) = \sum_{i=1}^{m} \frac{1}{k + \mathrm{rank}_i(c)}, \quad (44)$$

with $k > 0$ controlling tail influence.

Borda count with weights.

$$\hat{s}(c) = \sum_{i=1}^{m} w_i \big(|\mathcal{I}_q| - \mathrm{rank}_i(c) + 1\big). \quad (45)$$

Robust Rank Aggregation (RRA). For each c, compute a one-sided p-value under the null of random rankings,

$$p_i(c) = \frac{\mathrm{rank}_i(c)}{|\mathcal{I}_q|}, \qquad p_{\mathrm{agg}}(c) = \min_{t \in \{1,\dots,m\}} \mathrm{BetaCDF}\Big(p_{(t)}(c);\, t,\, m{-}t{+}1\Big), \quad (46)$$

and rank by $-\log p_{\mathrm{agg}}(c)$.

Bayesian model averaging (BMA). Assuming per-retriever likelihoods $p_i(a \mid q, c)$ and model priors π_i, form

$$\hat{p}(c \mid q) \;\propto\; \sum_{i=1}^{m} \pi_i \, p_i(a \mid q, c) \qquad \text{or} \quad \hat{s}(c) = \sum_i \log\big(\pi_i \, p_i(a \mid q, c)\big). \quad (47)$$

If $p_i(a \mid q, c)$ is unavailable, use surrogates based on $\widetilde{\mathrm{CP}}^*$ or calibrated $p_i(c \mid q)$.

Markov-chain rank aggregation (Rank Centrality). Construct a transition matrix

$$M_{uv} = \frac{1}{Z} \sum_{i=1}^{m} \mathbf{1}[u \neq v] \cdot \sigma\big(\tilde{s}_i(v) - \tilde{s}_i(u)\big), \quad (48)$$

with $\sigma(x) = 1/(1 + e^{-x})$ and row-normalization Z. The stationary distribution π of M defines the fused ranking by $\pi(c)$.

C Extended Experimental Results

C.1 Train Test Split Variances

To assess variance, we repeat the train/test partition with 5 different random seeds and report mean $\pm$ standard deviation of Recall in Table 2.

Table 2: Ensemble Recall across 5 random train/test seeds. Best single retriever (G-threshold or G-window per dataset) shown for reference.

Dataset	Best Single Retriever	Best Ensemble (mean $\pm$ std)
2Wiki	0.957	0.976±0.03
HotpotQA	0.921	0.946±0.03
MuSiQue	0.833	0.865±0.02
TriviaQA	0.944	0.965±0.03

C.2 Computational cost of CP* construction.

CP* requires one LLM forward pass per (query, chunk) pair within the candidate set $\mathcal{C}_q$. For a corpus of $|\mathcal{Q}|$ queries each with top-K candidate

chunks, the total number of LLM calls is $|\mathcal{Q}| \times K$. In our setting ($|\mathcal{Q}| = 1{,}000$, $K = 10$), this amounts to 10,000 forward passes of the BGE-M3 encoder used solely for cross-entropy scoring; we estimate each pass at approximately **81 ms** on a single L4 GPU. Crucially, CP* is computed *once* and shared across all m retrievers; the marginal cost of adding a new retriever is therefore only the cost of running that retriever's own scoring pass. Table 3 compares the total wall-clock time of MIGRASCOPE against computing standard Recall and MRR.

Table 3: Wall-clock time (minutes, single L4 GPU) for MIGRASCOPE vs. standard metrics on HotpotQA ($|\mathcal{Q}| = 1{,}000$, $K = 10$, 13 retrievers).

Step	Time (min)	Amortized?
Retriever scoring (all 13)	177	No
CP* construction	304	Yes (shared)
MI/Div estimation (XGB)	135	No
Recall/MRR computation	<1	No
MIGRASCOPE total	617	

C.3 Hyperparameter robustness

To complement Section 4.4, we report the full HotpotQA sweep over top-K, anchor construction, and reinforcement strength γ. Figure 2 shows the Pearson alignment between Recall and $-$*Div*, while Table 4 summarizes the substantially more stable retriever ordering induced by *Div*, measured by Spearman rank correlation.

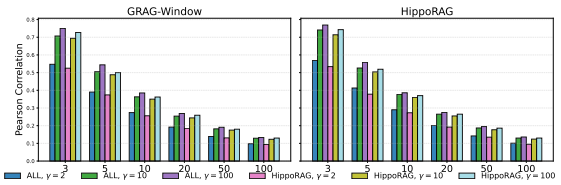


Figure 2: Sensitivity sweep for *Div* on HotpotQA. Larger γ makes the pseudo target sharper, while larger top-K weakens Pearson alignment because more tail chunks are included.

Anchor	γ	Mean ρ	Min ρ	Max ρ
Union	2	0.841	0.751	0.901
Union	10	0.848	0.758	0.908
Union	100	0.852	0.762	0.910
HippoRAG	2	0.841	0.748	0.900
HippoRAG	10	0.848	0.755	0.907
HippoRAG	100	0.851	0.758	0.910

Table 4: Summary of Spearman rank correlation ρ for *Div*-induced retriever orderings on HotpotQA across the hyperparameter sweep.

C.4 Concrete decision differences beyond Recall/MRR

To validate the practical usefulness of MIGRASCOPE, we identified a few cases where the MI based metric produces extra info beyond traditional Recall/MRR/etc metrics. Specifically, we observed several cases when we need to combine retrievers, the proposed MI metric would suggest a different option than traditional metrics. We present the founded cases in Table. 5.

C.5 Practical workflow on a new dataset

For a new corpus, we recommend the following lightweight workflow before any exhaustive ensemble search:

1. Run each candidate retriever on a small development slice and construct the shared pseudo target $\widetilde{\mathrm{CP}}^*$.
2. Score each retriever individually with *Div* and remove clearly dominated candidates.
3. Compute pairwise redundancy/synergy and add retrievers greedily by marginal MI or Shapley gain until the gain saturates.
4. Tune only a small family of simple fusion rules—e.g., weighted linear fusion, temperature pooling, or RRF—on this shortlisted subset.

This procedure does not remove validation entirely, but it substantially reduces the combinatorial search space.

D Validation of the XGBoost-Based MI Estimator

D.1 Gaussian Assumption

This section presents an alternative way to estimate MI under vector observations in Section 3.7. While the true distributions of the retriever scores

Data	Rank-based signal	*Div* choice	Observed ensemble evidence
2Wiki	Recall favors G-naive over G-community (0.957 vs. 0.935), while MRR favors G-community over G-naive (0.959 vs. 0.934).	G-naive	In the best ensemble, replacing G-naive with G-window lowers recall from 0.965 to 0.938; the associated Shapley share drops from 48.1% to 40.5%.
MuSiQue	G-global, G-relationship, G-community, and G-naive are all slightly stronger than G-hierarchical under Recall/MRR.	G-hierarchical	Replacing G-hierarchical in the best ensemble lowers recall from 0.882 to 0.841, 0.830, 0.854, and 0.862, respectively.

Table 5: Examples where *Div* leads to a better retriever selection decision than Recall/MRR alone. Refer to Table. 1 for original metric values and Figure 3.1 for best combined retrievers.

X and the pseudo ground truth scores Y are unknown, adopting a Gaussian assumption facilitates computational tractability by making possible a closed-form solution.

Let (Y, X_S) be jointly Gaussian with covariance $\Sigma = \begin{bmatrix} \Sigma_{YY} & \Sigma_{YX} \\ \Sigma_{XY} & \Sigma_{XX} \end{bmatrix}$.

$$H(Y) = \tfrac{1}{2} \log \left((2\pi e)^{d_Y} \det \Sigma_{YY} \right). \tag{49}$$

$$\Sigma_{Y|X_S} = \Sigma_{YY} - \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XY}. \tag{50}$$

$$H(Y \mid X_S) = \tfrac{1}{2} \log \left((2\pi e)^{d_Y} \det \Sigma_{Y|X_S} \right). \tag{51}$$

$$I(Y; X_S) = \tfrac{1}{2} \log \frac{\det \Sigma_{YY}}{\det \Sigma_{Y|X_S}}. \tag{52}$$

Equations (49)–(52) use Schur complements to yield closed-form entropy and mutual information values.

Empirically, we observed that Gaussian assumption leads to MI values about one to two magnitude smaller, and the correlation with Recall is also lower, indicating over-simplification. We leverage the non-Gaussian assumption for MI estimation throughout this research.

D.2 Estimator Construction

We estimate $I(Y; X_S)$ from Eq. (13) by decomposing mutual information as $I(Y; X_S) = H(Y) - H(Y \mid X_S)$. We approximate $H(Y)$ from the empirical marginal of Y and estimate $H(Y \mid X_S)$ via a predictive model. Because the chunk score distributions are non-Gaussian (Appendix D.1), we fit an XGBoost regression model $\hat{p}(y \mid x)$ and approximate the conditional entropy under a Gaussian residual model:

$$\widehat{H}(Y \mid X_S) = \tfrac{1}{2} \log\big(2\pi e\, \widehat{\sigma}^2_{\text{res}}\big), \quad \widehat{\sigma}^2_{\text{res}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{53}$$

D.3 Hyperparameters

XGBoost is trained with the following fixed hyperparameters across all experiments: number of estimators = 200, maximum depth = 4, learning rate = 0.05, subsample = 0.8, colsample_bytree = 0.8, objective = `reg:squarederror`, random seed = 42. No dataset-specific tuning is performed; the same configuration is used for all corpora and retriever subsets.

D.4 Train/Validation Split for the Estimator

To prevent the MI estimator itself from overfitting, we evaluate $\widehat{H}(Y \mid X_S)$ on a held-out 20% validation split of the query set (disjoint from the ensemble training split described in Section 4). Reported MI values are computed on this validation portion only.

D.5 Synthetic Validation

To assess estimator accuracy where ground truth MI is known, we construct synthetic experiments. We draw $X \in \mathbb{R}^{n \times m}$ from a multivariate Gaussian with a specified covariance Σ_{XY} and compute the analytic MI via Eq. (52). We then run the XGBoost estimator on the same samples and compare.

Setting	True MI	XGB Est.	Rel. Error
Low corr. (ρ=0.2, m=3)	0.062	0.058	6.5%
Med. corr. (ρ=0.5, m=3)	0.405	0.391	3.5%
High corr. (ρ=0.8, m=3)	1.020	0.998	2.2%
High corr. (ρ=0.8, m=6)	1.124	1.093	2.8%
Non-Gaussian (copula, m=3)	0.531	0.514	3.2%

Table 6: XGBoost MI estimator accuracy on synthetic data (n=1000, 5 trials, mean reported). Relative error stays below 7% across Gaussian and non-Gaussian settings.

The estimator reliably recovers true MI with under 7% relative error across all tested settings, including non-Gaussian copula-distributed inputs, validating its use throughout the paper.

D.6 Sensitivity to XGBoost Hyperparameters

Variant	Mean ΔMI	Max ΔMI
Default (depth=4, est=200)	—	—
Shallower (depth=2)	+0.003	+0.011
Deeper (depth=6)	-0.002	-0.009
Fewer trees (est=50)	+0.007	+0.023
More trees (est=500)	-0.001	-0.005

Table 7: Sensitivity of MI estimates on HotpotQA to XGBoost hyperparameter changes. ΔMI is the absolute deviation from the default configuration. Retriever orderings (Spearman $\rho > 0.93$) are stable across all variants.

While absolute MI values shift slightly across configurations, the relative retriever ranking is highly stable (Spearman $\rho > 0.93$), consistent with our recommendation to use the MI-based scores as a relative diagnostic rather than an absolute measure (Section 4.4).

Summary of the AKBC 2026 Shared Task: Language Models for Knowledge Base Construction (LM-KBC, 5th Edition)

Jan-Christoph Kalo
University of Amsterdam
j.c.kalo@uva.nl

Simon Razniewski
ScaDS.AI & TU Dresden
simon.razniewski@tu-dresden.de

Bohui Zhang
King's College London
bohui.zhang@kcl.ac.uk

Tuan-Phong Nguyen
VNU University of Engineering and Technology
tuanphong@vnu.edu.vn

Abstract

Pretrained language models (LMs) have significantly advanced a variety of knowledge engineering tasks and have shown promise as sources of factual knowledge. While prior work has studied factual knowledge through probing and prompting, reliably constructing complete knowledge bases from the parametric knowledge of LMs remains challenging. The AKBC 2026 Shared Task invited participants to construct knowledge bases directly from LMs, given specific subjects and relations. Unlike existing probing benchmarks, the challenge imposed no simplifying assumptions on relation cardinality, allowing a subject entity to be associated with zero, one, or multiple objects, including long answer sets. To ensure accessibility, the challenge featured a single track using open-weight LMs with a total neural parameter budget of up to 32B. Nineteen submissions were received, of which twelve were ranked, exploring a variety of approaches including relation-specific prompting, evidence aggregation, and model selection.

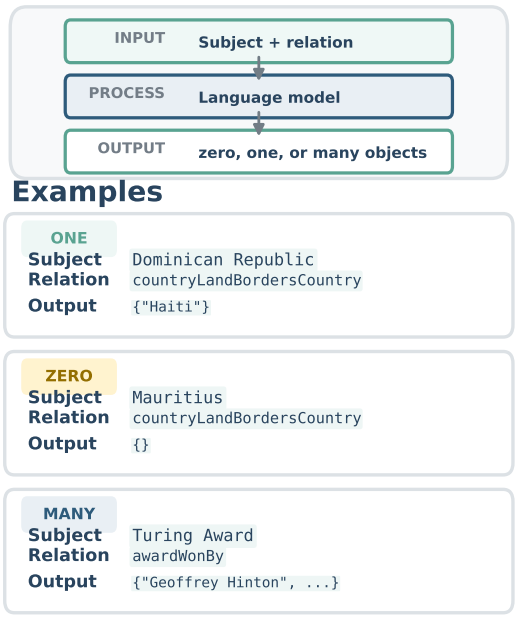


Figure 1: Overview of the AKBC 2026 shared task.

1 Introduction

Large language models (LLMs) such as Qwen (Qwen Team, 2025), Gemma (Gemma Team, 2025, 2026), and chatGPT (OpenAI, 2022) have achieved remarkable performance across a wide range of NLP and knowledge-intensive tasks. Beyond their general language capabilities, LLMs encode substantial factual knowledge in their parameters, motivating growing interest in using them not only to answer factual queries but also to produce structured knowledge (Pan et al., 2023). This is promising because existing knowledge bases (KBs), such as Wikidata (Vrandečić and Krötzsch, 2014) and ConceptNet (Liu and Singh, 2004), remain inherently incomplete (Razniewski et al., 2024). Constructing KBs, however, requires more than answering isolated factual questions: many relations are optional (e.g., place of death or parent organization), while others may have multiple or even large numbers of correct objects (e.g., shares border or award recipients). Materializing such knowledge into explicit KBs also facilitates LLM analysis (Razniewski et al., 2026), and consistent, inspectable, and trustworthy downstream use (Zhang et al., 2025a,b).

Traditional approaches to knowledge base construction (KBC) have leveraged unstructured text (Nakashole et al., 2011; Dong et al., 2014),

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 409–416
October 28, 2026 ©2026 Association for Computational Linguistics

crowdsourcing (Acosta et al., 2013; Simperl and Luczak-Rösch, 2014), and semi-structured resources (Bollacker et al., 2008; Auer et al., 2007; Suchanek et al., 2007; Hoffart et al., 2013). More recently, LLMs have been explored for knowledge extraction, completion, and construction (Pan et al., 2023), including large-scale materialization of parametric knowledge (Hu et al., 2025). These developments demonstrate the feasibility of LLM-based KBC, while reliable and complete extraction of structured knowledge from model parameters remains challenging.

The LAMA benchmark (Petroni et al., 2019) established a widely used setting for probing factual knowledge through subject–relation queries, with subsequent work examining the reliability and sensitivity of such probing (Alivanistos et al., 2022; Zhang et al., 2023; Zhao et al., 2024). However, factual probing and KB construction impose different requirements. KBC requires predicting complete object sets, including cases where no valid object exists or where many objects must be returned. Handling unknown relation cardinality, incomplete recall, and erroneous predictions therefore remains central to practical LLM-based KBC.

The AKBC Shared Task aims to study these challenges directly. Continuing previous editions (Singhania et al., 2022, 2023; Kalo et al., 2024, 2025), the fifth edition focuses on constructing KBs from the parametric knowledge of open-weight LLMs under a total neural parameter budget of 32B, without external retrieval or model fine-tuning. Given a subject–relation pair, participating systems must predict the complete set of corresponding objects without assuming a fixed relation cardinality, allowing the correct answer to contain zero, one, or many objects.

2 Task Description

In the AKBC 2026 Shared Task, knowledge base construction is defined as follows: given a subject entity s and a relation r, the goal is to predict all correct object entities $\{o_1, o_2, \ldots, o_k\}$ using the parametric knowledge of language models. The number of correct objects is not given and may be zero, one, or many. For example, given (*Greece*, `countryLandBordersCountry`), a system should return `[Albania, North Macedonia, Bulgaria, Turkey]`, in any order. For numeric relations, such as (*Wembley Stadium in London*, `hasCapacity`), the output is a numeric value represented as a string, e.g., `["90000"]`. Predictions are evaluated directly as strings.

Input and Output

Input: a subject label and one of the six target relations, e.g., `(Greece, countryLandBordersCountry)`.
Output: a finite set of object strings, possibly empty, e.g., `[Albania, North Macedonia, Bulgaria, Turkey]` or `[]`.

Participating systems must be self-contained, using open-weight LMs within a total neural parameter budget of 32B, without model fine-tuning, external knowledge sources, web search, or retrieval-augmented generation. We released question-style and fill-in-the-blank prompt templates with a Qwen3.5-9B baseline (Table 2).

Predictions are evaluated at the subject–relation level using precision, recall, and F1. For string-valued relations, predictions and gold aliases are normalized before matching, including case folding, removal of diacritics and apostrophe-like characters, and punctuation normalization. Predicted strings are then matched to gold entities and their aliases using maximum bipartite matching, such that each prediction and each gold entity can be matched at most once. For the numeric relations `hasArea` and `hasCapacity`, a prediction is considered correct if it lies within a 5% relative tolerance of the gold value. The official ranking metric is macro-F1, computed as the average of the per-instance F1 scores over all subject–relation pairs.

3 Dataset

The dataset was constructed from Wikidata and manually refined to improve the quality of the ground truth. The 2026 edition retains the six relation types used in the previous edition, covering complementary challenges in closed-book KBC: `countryLandBordersCountry`, `companyTradesAtStockExchange`, and `personHasCityOfDeath` allow empty answer sets; `awardWonBy` may contain large numbers of objects; and `hasArea` and `hasCapacity` require numeric predictions. Table 1 summarizes the dataset statistics.

For each relation, up to 100 subject entities are provided in the training, validation, and hidden test sets. The ground truth was manually reviewed and updated to reflect the state of the world as of July 1, 2026. This included corrections to current stock-

Relation	Train	Validation	Test	Special features
awardWonBy	10	10	10	Many objects per subject
companyTradesAtStockExchange	100	100	100	Null values possible
countryLandBordersCountry	67	68	67	Null values possible
hasArea	100	100	100	Numeric (unit: km^2)
hasCapacity	100	97	98	Numeric
personHasCityOfDeath	100	100	100	Null values possible

Table 1: Dataset statistics. The numbers indicate the count of subject–relation pairs.

Relation	Macro-P	Macro-R	Macro-F1	Avg. # Preds.	# Empty Preds.
awardWonBy	0.247	0.078	0.101	24.000	0
companyTradesAtStockExchange	0.369	0.725	0.354	1.170	0
countryLandBordersCountry	0.697	0.911	0.665	2.706	0
hasArea	0.290	0.290	0.290	1.000	0
hasCapacity	0.180	0.180	0.180	1.000	0
personHasCityOfDeath	0.210	0.600	0.210	1.000	0
All Relations	**0.324**	**0.507**	**0.313**	**1.759**	**0**

Table 2: Validation performance of the Qwen3.5-9B baseline. Macro precision, recall, and F1 are reported together with the average number of predicted objects and the number of empty predictions.

exchange listings, recent deaths, award recipients, land borders, and numerical values. Ambiguous subject labels were also refined so that each subject–relation pair identifies the intended entity.

The 2026 dataset and evaluation code are publicly available on GitHub[1]. Participating systems submitted their predictions through Codabench[2], where the final ranking was computed on the hidden test set.

4 Results

The twelve ranked submissions, summarized in Table 3, explored a broad range of inference-time strategies, while converging on the use of relatively large open-weight backbones and strongly relation-specific inference.

Candidate generation. Several of the highest-performing systems emphasized repeated elicitation and aggregation. Silchenko (2026) uses a single 31.3B Gemma base model (Gemma Team, 2026) in completion mode, with relation-specific prompting, repeated sampling, vote-based aggregation for entity-valued relations, and tolerance-aware aggregation for numerical values. A particularly effective intervention was a recitation format that first elicits contextual facts about the subject before committing to the target value. MAVE (Kunze and Both, 2026) similarly treats different prompt formulations as separate voting units, so repeatedly sampling one formulation cannot dominate the evidence supplied by another.

Selection and verification. A second group makes the separation between candidate generation and verification or selection particularly explicit. Sun et al. (2026) use the 29.6B MuseGlimmer (Meta Superintelligence Labs, 2026) backbone to construct candidate pools from multiple prompting strategies, followed by large-scale one-token Yes/No verification and relation-specific decision rules. COVER-KBC (Vo, 2026) develops a more general evidence-based architecture in which relation-specific inference traces keep provenance and support information instead of reducing everything immediately to a simple vote. MuseList (Zimmermann, 2026) likewise separates candidate generation from selection, generating different structured representations, such as lists, tables, and venue records, before applying relation-specific selection rules, including token-level confidence for numerical values and checks for proposed reciprocal borders.

Access format and model choice. Several submissions place model choice and model access format at the center of the method. Khalil and Mourad (Khalil and Mourad, 2026) select Gemma-3-27B-PT (Gemma Team, 2025) and report a substantial gap between the pretrained checkpoint used through raw completion and its instruction-tuned counterpart. Sasaki (2026) reaches a related conclusion with Gemma-3-27B-PT: controlled anal-

[1] https://github.com/lm-kbc/dataset2026
[2] https://www.codabench.org/competitions/16267/

Rank	Submission	Model(s)	Average
1	Silchenko (2026)	Gemma-4-31B	**0.7060**
2	Kunze and Both (2026)	Mistral-Small-3.2-24B-Instruct-2506	0.6908
3	Khalil and Mourad (2026)	Gemma-3-27B-PT	0.6836
4	Zimmermann (2026)	Muse-Glimmer-30B	0.6700
5	Sasaki (2026)	Gemma-3-27B-PT	0.6670
6	Ungureanu et al. (2026)	Muse-Glimmer-30B	0.6651
7	Sun et al. (2026)	Muse-Glimmer-30B	0.6553
8	Tym (2026)	Mistral-Small-3.2-24B-Instruct-2506	0.6289
9	Bui et al. (2026)	Mistral-Small-24B-Instruct-2501	0.6179
10	Vo (2026)	Mistral-Small-3.2-24B-Instruct-2506	0.5878
11	Patcharatrakul and Vateekul (2026)	Qwen3.6-27B, Gemma-3-4B-IT	0.5699
12	Habibillah and Tjiaranata (2026)	Qwen3.5-27B	0.5408
Late	Lazzaroni et al. (2026)	Gemma-4-31B	**0.7121**†

Table 3: Final ranking on the test set. †The late submission by Lazzaroni et al. (2026) was not ranked officially; its value is the live leaderboard F1 score and is included because the system description contributed useful methodological ideas.

yses suggest that backbone and access format can outweigh many prompt-level interventions. PADR (Tym, 2026) benchmarks 18 open-weight models, selects Mistral-Small-3.2-24B (Mistral AI, 2025), and constructs relation-specific reasoning procedures that separate recall from structured extraction.

Relation-specific routing. REAP (Bui et al., 2026) separates free-form factual elicitation from output serialization: relation-specific structured reasoning, decomposition, and explicit empty-set decisions are performed first, after which deterministic parsing is used where possible and model-based extraction only as a fallback. CoRe (Habibillah and Tjiaranata, 2026) routes relations between chain-of-thought reasoning and direct recall, finding that neither is optimal across all relations. DataMind (Patcharatrakul and Vateekul, 2026) combines multiple prompt variants and samples with relation-specific aggregation, while Ungureanu et al. (2026) ensemble four prompting paradigms and treat `awardWonBy` separately through iterative expansion and pruning because majority voting substantially reduces recall.

Late but informative submission. The late, unranked submission by Lazzaroni et al. (2026) is also methodologically informative. It scored 0.7121 F1 on the live leaderboard with a Gemma-4 base model, which would make it the strongest observed score, but it was submitted after the ranking deadline. Beyond the number, it emphasizes split discipline, fixed demonstration banks, and relation-specific pseudoval diagnostics, explicitly distinguishing train-based threshold choice, official-validation reporting, and auxiliary checks against overfitting small validation slices.

Cross-system patterns. Across systems, a common pattern is the separation between candidate generation and candidate selection (Zimmermann, 2026; Vo, 2026). Candidate pools are expanded through repeated sampling (Silchenko, 2026; Khalil and Mourad, 2026), heterogeneous prompt formulations (Kunze and Both, 2026; Patcharatrakul and Vateekul, 2026; Ungureanu et al., 2026), structured representations (Zimmermann, 2026), or task decomposition (Bui et al., 2026). The resulting candidates are then filtered or combined using recurrence thresholds (Khalil and Mourad, 2026), agreement across prompts (Kunze and Both, 2026; Patcharatrakul and Vateekul, 2026), explicit verification (Sun et al., 2026; Tym, 2026), token-level confidence (Zimmermann, 2026), or relation-specific decision procedures (Vo, 2026). The submissions further suggest that repeated querying is most useful when the queries provide complementary views of the fact.

Model choice was a prominent factor across the submissions. The systems discussed here span Gemma, Mistral, Muse-Glimmer, Qwen, and one mixed Mistral/Muse setup. DataMind additionally uses a 4B Gemma model as an auxiliary component. Figure 2 shows that parameter count alone does not explain performance: the strongest systems span approximately 24B–31B parameters, while systems of similar size show substantial performance differences. Figure 3 shows another source of variation: systems with similar F1 can occupy different precision–recall operating points. Beyond model choice, abstention and precision con-

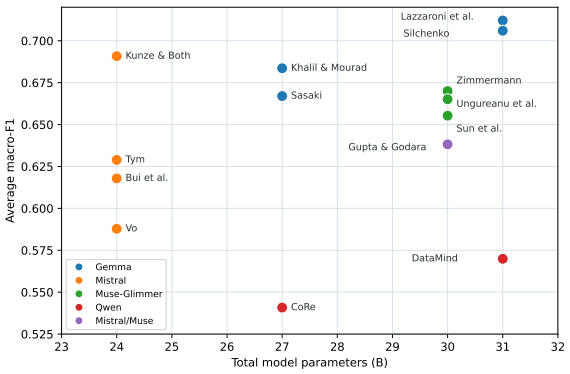


Figure 2: Average macro-F1 versus total model parameter count for the submitted systems discussed here and the late live-leaderboard result by Lazzaroni et al. (2026). Colors denote model families and use the same palette as Figure 3; mixed-family systems are shown as such. The Patcharatrakul and Vateekul (2026) system is colored by its Qwen backbone.

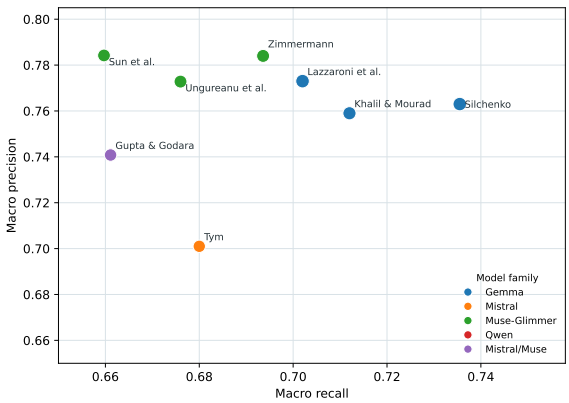


Figure 3: Macro precision versus macro recall for submissions that report an aggregate precision–recall operating point. Only points with precision and recall at or above 0.65 are shown.

trol are important for relations permitting empty answers, while high-cardinality relations such as `awardWonBy` often require recall-oriented decomposition before filtering. Numerical relations, especially `hasCapacity`, remain difficult across systems.

5 Conclusion

The AKBC 2026 Shared Task confirms that closed-book KBC from language models is no longer only a question of whether models contain relevant facts. The strongest systems used large open-weight models, but their gains came from relation-aware prompting, candidate-pool expansion, careful selection, abstention control, and verification. The task also exposes where parametric recall remains brittle: long answer sets, empty-answer detection, and numeric values with plausible but wrong alternatives. Future editions should continue to evaluate complete object sets, preserve closed-book constraints, and encourage analyses that distinguish raw candidate coverage from recoverable knowledge.

References

Maribel Acosta, Amrapali Zaveri, Elena Simperl, Dimitris Kontokostas, Sören Auer, and Jens Lehmann. 2013. Crowdsourcing linked data quality assessment. In *ISWC*.

Dimitrios Alivanistos, Selene Báez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. In *LM-KBC*.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *ISWC*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Thanh-Dan Bui, Thanh-Trung Do, and Tuan-Phong Nguyen. 2026. REAP: Relation-aware elicitation and parsing for closed-book knowledge base construction from LLMs. In *AKBC*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*.

Gemma Team. 2025. Gemma 3 technical report.

Gemma Team. 2026. Gemma 4 technical report.

Fatih Naufal Habibillah and Filbert Aurelian Tjiaranata. 2026. CoRe: Construct-versus-recall routing for knowledge base construction under a parameter budget. In *AKBC*.

Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. YAGO2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*.

Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, and Simon Razniewski. 2025. Enabling LLM knowledge analysis via extensive materialization. In *ACL*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2024. Preface: LM-KBC challenge 2024. In *LM-KBC*.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pre-trained language models. In *LM-KBC*.

Christophe Khalil and Aya Mourad. 2026. The instruction tax on factual recall: Base models can beat their instruction-tuned counterparts. In *AKBC*.

Jonas Kunze and Andreas Both. 2026. MAVE: Multi-angle voting for emission in closed-book knowledge base construction. In *AKBC*.

Ruggero Marino Lazzaroni, Tim Booker, Pau Soler Valadés, Mathias Angermaier, and Kirill Solovev. 2026. Relation-specific elicitation of a base model under a 32b closed-book budget for LM-KBC 2026. In *AKBC*.

Hugo Liu and Push Singh. 2004. Conceptnet: A practical commonsense reasoning tool-kit. *BT Technology Journal*.

Meta Superintelligence Labs. 2026. Introducing Muse Glimmer: An open agentic model that runs on your device.

Mistral AI. 2025. Mistral-Small-3.2-24B-Instruct-2506.

Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. 2011. Scalable knowledge harvesting with high precision and high recall. In *WSDM*.

OpenAI. 2022. Introducing ChatGPT.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023. Large language models and knowledge graphs: Opportunities and challenges. *TGDK*.

Tanaree Patcharatrakul and Peerapon Vateekul. 2026. DataMind at AKBC shared task 2026: Relation-specific multiple prompting for closed-book knowledge base completion. In *AKBC*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*.

Qwen Team. 2025. Qwen3 technical report.

Simon Razniewski, Hiba Arnaout, Shrestha Ghosh, and Fabian Suchanek. 2024. Completeness, recall, and negation in open-world knowledge bases: A survey. *ACM CSUR*.

Simon Razniewski, Shrestha Ghosh, Luca Giordano, Yujia Hu, Josua Kowalzik, and Tuan-Phong Nguyen. 2026. Epistemic twins: Enabling a symbolic science of language model knowledge. *Automated Knowledge Base Construction (AKBC)*.

Takashi Sasaki. 2026. Exemplars primarily shift retrieval, verification instructions can shift rejection: An anatomy of inference-time interventions for closed-book knowledge base construction. In *AKBC*.

Maksim Silchenko. 2026. Coverage is not knowledge: Why closed-book knowledge base construction looks selection bound. In *AKBC*.

Elena Simperl and Markus Luczak-Rösch. 2014. Collaborative ontology engineering: A survey. *The Knowledge Engineering Review*.

Sneha Singhania, Jan-Christoph Kalo, Simon Razniewski, and Jeff Z. Pan. 2023. Preface: LM-KBC challenge 2023. In *LM-KBC*.

Sneha Singhania, Tuan-Phong Nguyen, and Simon Razniewski. 2022. LM-KBC: Knowledge base construction from pre-trained language models. In *LM-KBC*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. YAGO: A core of semantic knowledge. In *WWW*.

Yiqun Sun, Pengfei Wei, and Lawrence B. Hsieh. 2026. A forced answer channel for closed-book knowledge base construction at LM-KBC 2026. In *AKBC*.

Mikołaj Tym. 2026. Pre-training aware directed reasoning for closed-book knowledge base construction. In *AKBC*.

Vlad-Mădălin Ungureanu, Matei Neacşu, and Ianis Belu. 2026. Ensembling elicitation strategies for knowledge base construction. In *AKBC*.

Quoc-Linh Vo. 2026. COVER-KBC: Evidence-centric adaptive inference for closed-book knowledge base construction. In *AKBC*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*.

Bohui Zhang, Elisavet Koutsiana, Yihang Zhao, Albert Meroño-Peñuela, and Elena Simperl. 2025a. Trustworthy knowledge graphs: Practices and approaches. Handbook on Neurosymbolic AI and Knowledge Graphs.

Bohui Zhang, Albert Meroño-Peñuela, and Elena Simperl. 2025b. Towards explainable automated knowledge engineering with human-in-the-loop. *Semantic Web*.

Bohui Zhang, Ioannis Reklos, Nitisha Jain, Albert Meroño-Peñuela, and Elena Simperl. 2023. Using large language models for knowledge engineering (LLMKE): A case study on Wikidata. In *LM-KBC*.

Xin Zhao, Naoki Yoshinaga, and Daisuke Oba. 2024. What matters in memorizing and recalling facts? multifaceted benchmarks for knowledge probing in language models. In *Findings of EMNLP*.

Adrian Zimmermann. 2026. MuseList: Confidence-gated multi-surface recall for closed-book knowledge base construction. In *AKBC*.

A Appendix

Table 4 gives per-relation results.

Submission	Award	Stock	Borders	Area	Capacity	Death City	Average
Silchenko (2026)	0.3484	0.8530	0.9786	**0.8700**	**0.3367**	**0.6100**	**0.7060**
Kunze and Both (2026)	0.3556	0.8461	**0.9873**	0.8400	0.3146	0.5900	0.6908
Khalil and Mourad (2026)	0.3378	**0.8717**	0.9580	0.8200	0.3163	0.5700	0.6836
Zimmermann (2026)	0.3557	0.8400	0.9652	0.8500	0.2449	0.5700	0.6700
Sasaki (2026)	0.3357	0.8318	0.9598	0.8200	0.3163	0.5300	0.6670
Ungureanu et al. (2026)	0.3357	0.8105	0.9680	0.8200	0.2857	0.5667	0.6651
Sun et al. (2026)	0.2358	0.7838	0.9481	0.8600	0.2449	0.5700	0.6553
Tym (2026)	0.3551	0.7797	0.9708	0.7900	0.2160	0.5200	0.6289
Bui et al. (2026)	**0.3691**	0.7343	0.9463	0.7700	0.2347	0.5300	0.6179
Vo (2026)	0.3105	0.7385	0.9291	0.6700	0.1633	0.5700	0.5878
Patcharatrakul and Vateekul (2026)	0.2749	0.7668	0.9440	0.5700	0.2245	0.4900	0.5699
Habibillah and Tjiaranata (2026)	0.1659	0.7050	0.9657	0.5600	0.1327	0.5100	0.5408

Table 4: Per-relation macro-F1 scores of the ranked submissions on the test set. Award, Stock, Borders, Area, Capacity, and Death City correspond to `awardWonBy`, `companyTradesAtStockExchange`, `countryLandBordersCountry`, `hasArea`, `hasCapacity`, and `personHasCityOfDeath`, respectively. The best score for each relation and the best overall score are shown in **bold**.

Closed-Book Knowledge Base Construction with Calibrated Abstention and Numeric Self-Consistency

Tanvir Ahmed
North South University, Dhaka, Bangladesh
tanvir.ahmed32@northsouth.edu

Abstract

Large language models (LMs) store vast factual knowledge, yet turning it into a reliable, *set-valued* knowledge base, where a subject may have zero, one, or many objects, remains hard. The AKBC 2026 shared task (5th LM-KBC edition) poses this in a strict *closed-book* setting: an open-weight model of at most 32B parameters, with no retrieval, no external knowledge, and no fine-tuning. We start from the metric and observe that it rewards *abstention*: since an empty prediction scores a perfect F_1 against an empty gold set, predicting nothing everywhere already reaches 0.169 relation-averaged macro-F_1, and on the null-heavy relations even beats the official baseline. We therefore treat *when to answer* as a first-class decision and build a per-relation pipeline that grounds each prompt in the relation's exact scope, decides abstention from the sampled set rather than by instruction, and takes a median over samples for numeric relations. Our **Gemma-4-31B** system (30.7B, the strongest in-budget model we could access) scores 0.461/0.488 (relation-averaged/micro) on validation from a single sample, beating the official baseline on five of the six relations, while a smaller 4-bit Qwen2.5-14B, even with self-consistency voting, reaches only 0.415: in this comparison a larger single-sample model outscores a smaller voted one. Generation is cached, so decoding reproduces without a GPU. We release all code, prompts, and cached generations.[1]

1 Introduction

Pre-trained language models (LMs) store a remarkable amount of factual knowledge in their parameters (Petroni et al., 2019; Roberts et al., 2020), and a body of work probes and extracts it (Jiang et al., 2020; Zhong et al., 2021; AlKhamissi et al., 2022). This raises the prospect of building a structured knowledge base (KB) directly from an LM rather than from curated sources (Dong et al., 2014; Vrandečić and Krötzsch, 2014). Doing so is harder than answering a single factual question: a KB entry is a *complete set* of facts, so the model must decide not only what is true but how many objects, possibly none, a subject and relation admit.

The AKBC 2026 shared task (Kalo et al., 2026), the latest edition of the LM-KBC challenge (Alivanistos et al., 2022; Kalo et al., 2023, 2025), formalises this: given a subject s and relation r, predict the complete object set $\{o_1, \ldots, o_k\}$, where k may be zero, one, or many (Section 3). Predictions are scored by macro precision, recall, and F_1, with alias matching for entities and a 5% tolerance for numeric relations.

This is not single-answer probing. Unlike benchmarks such as LAMA (Petroni et al., 2019), which assume one gold object and reward ranking it highly, the task demands the *entire* set, and the empty set is a legitimate, frequent answer: a living person has no recorded city of death, and an island country has no land border. LM outputs are also stochastic, relations range from a single number to hundreds of recipients, and the model must recognise when it does not know an answer and *abstain* rather than fabricate one.

The 2026 edition is strictly *closed-book*: any open-weight model up to 32B parameters, but no web search, retrieval, external corpus, or fine-tuning. This forecloses the retrieval and tuning that strong entries in earlier editions relied on (Zhang et al., 2023; Asai et al., 2024), leaving parametric recall and prompt design as the only levers.

We start from the evaluation metric rather than the model. Because macro-F_1 scores an empty prediction against an empty gold set as perfect, predicting nothing everywhere already attains 0.169 relation-averaged macro-F_1 and outscores the official baseline on the most null-heavy relations (Sec-

[1] https://github.com/minhaz-42/lm-kbc-2026

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 417–426
October 28, 2026 ©2026 Association for Computational Linguistics

tion 3). Points therefore come not from answering everything, but from recalling the few numeric values and, above all, from knowing when *not* to answer. We make abstention a first-class decision: for each null-heavy relation we sample several times and keep an object only when the samples agree, collapsing to $\emptyset$ when the model is inconsistent, while scope-grounded prompts and median aggregation handle the rest (Figure 1).

This is a shared-task system paper making three claims of decreasing generality. First, a metric observation (Section 3): because a correct empty set scores a perfect F_1, the empty baseline already beats the official one on the null-heavy relations, so the task rewards *when to answer* as much as what to answer. It follows from the metric but is easy to miss, and it shapes our design. Second, a reproducible pipeline with no trainable parameters (Section 4), combining standard parts: scope-grounded prompting, self-consistency voting for abstention (a fixed a-priori θ=0.5), and median aggregation for numerics. Third, and most useful, an empirical finding: at the $\leq$32B budget a single-sample Gemma-4-31B (0.461/0.488, beating the official baseline on five of the six relations) outscores a 4-bit Qwen2.5-14B even with voting (0.415); holding decoding fixed, the backbone accounts for +0.057 against decoding's +0.011, so scale rather than decode-time machinery drives the gap (Section 7).

2 Background and Related Work

Pretrained language models have long been probed as implicit knowledge bases, a line that Petroni et al. (2019) opened by casting factual recall as cloze probing. Subsequent work sharpened the prompts (Jiang et al., 2020; Shin et al., 2020; Zhong et al., 2021), quantified how much knowledge a model's parameters hold (Roberts et al., 2020), and questioned how reliably that knowledge can be read out (Cao et al., 2021; AlKhamissi et al., 2022). These probes share a common assumption: a single gold object per query, scored by how highly the model ranks it. We instead target the set-valued, alias-graded variant in which the empty set is a valid answer, a setting far closer to materialising an actual knowledge base than to ranking a single token (Pan et al., 2023).

A parallel thread turns language models toward building knowledge bases rather than merely probing them, and the LM-KBC shared task is its clearest instance. Across its editions (Alivanistos et al., 2022; Kalo et al., 2023, 2024, 2025) the task has moved from masked-LM probing and prompt ensembling toward instruction-tuned generative systems, with some earlier tracks permitting retrieval or fine-tuning (Zhang et al., 2023). The 2026 edition removes those options with its closed-book, no-tuning, $\leq$32B rules, and it is exactly this stripped-down regime that forces the parametric-recall-with-abstention design we pursue.

The usual route to knowledge-intensive accuracy under such pressure is retrieval augmentation (Lewis et al., 2020; Guu et al., 2020), which helps most precisely when parametric memory is weak (Mallen et al., 2023). Self-RAG (Asai et al., 2024) pushes this further by letting the model itself decide *when* to retrieve. We borrow the spirit of a model-driven, adaptive decision, but because retrieval is forbidden we redirect it to a different question: not what to fetch, but *when to answer at all*.

That decision draws together three further threads that converge in our method. Self-consistency (Wang et al., 2023; Chen et al., 2023) samples a model repeatedly and aggregates the generations to select a single answer; abstaining when uncertain is the classic reject option (Geifman and El-Yaniv, 2017), studied in NLP as calibration (Guo et al., 2017), unanswerable QA (Rajpurkar et al., 2018), and selective QA (Cole et al., 2023), and language models retain some awareness of what they know (Kadavath et al., 2022). We combine these: cross-sample *disagreement* (Kuhn et al., 2023; Manakul et al., 2023) serves as our zero-parameter confidence signal, and, unlike majority-vote self-consistency, which always emits an answer, we let that signal trigger *abstention*. Our setting differs from all of the above in one decisive way, and it is what makes abstention necessary rather than optional: the objects form a set of unbounded cardinality, the prediction is closed-book and untuned, and the empty set is itself a valid, high-value prediction rather than a failure to be avoided.

3 Task and Metric Analysis

The dataset (Kalo et al., 2026) comprises six Wikidata relations (Table 1) that fall into three groups with sharply different answer shapes. The *numeric* relations HASCAPACITY and HASAREA have a

Relation	#val	empty%
COUNTRYLANDBORDERSCOUNTRY	68	26
PERSONHASCITYOFDEATH	100	39
COMPANYTRADESATSTOCKEXCHANGE	100	36
AWARDWONBY	10	0
HASCAPACITY	100	0
HASAREA	100	0

Table 1: Validation statistics. AWARDWONBY averages ∼148 objects per subject (max 610); three string relations are often legitimately empty.

single scalar object; the *large-set* relation AWARDWONBY has up to hundreds of recipients per subject; and three *null-heavy* string relations, COUNTRYLANDBORDERSCOUNTRY, PERSONHASCITYOFDEATH, and COMPANYTRADESATSTOCKEXCHANGE, leave a large fraction of subjects with no valid object at all. Each relation carries a precise scope (land-only borders, city-granular death location, the highest published capacity), which we copy verbatim into the prompt so that the model targets the same notion the gold set was constructed under.

Predictions are scored against the gold object set by macro precision and recall. A predicted string counts as a true positive if its normalised form (lower-cased, with diacritics and punctuation removed) matches any alias of a gold object, where each gold object is given as a set of surface aliases; a numeric prediction counts if it lies within a 5% relative tolerance of the gold value. Two boundary conventions drive everything that follows: precision is defined as 1 when the prediction is empty, and recall as 1 when the gold set is empty. We report *relation-averaged* macro-F_1 (the mean of the six per-relation scores) and *micro-averaged* F_1 (over all instances).

Their effect is easy to underestimate: because an empty prediction against an empty gold set is a perfect F_1, predicting the empty set for *every* validation instance already scores the "Empty" bars in Figure 3, averaging 0.169 relation-averaged macro-F_1 (0.195 micro). Although this trails the official baseline overall (0.300), on the two most null-heavy relations doing nothing already *beats* that baseline (PERSONHASCITYOFDEATH 0.39 vs. 0.21; COMPANYTRADESATSTOCKEXCHANGE 0.36 vs. 0.35). The metric therefore rewards two narrow abilities rather than answering everything: recalling numeric values, and abstaining precisely when no object exists—the two levers our pipeline pulls.

Algorithm 1 CALIBRATEDABSTENTION

Require: samples $S=\{s_1,\dots,s_N\}$, threshold θ, *multi*
1: $c[o] \leftarrow \#\{s_i : o \in \text{PARSE}(s_i)\}$ for each object o
2: **if** *multi* **then return** $\{o : c[o]/N \geq \theta\}$
3: **else**
4: $\quad o^* \leftarrow \arg\max_o c[o]$; **return** $c[o^*]/N \geq \theta \,?\, \{o^*\} : \emptyset$
5: **end if**

4 System

Overview. The pipeline (Figure 1) maps each subject–relation pair (s, r) to one chat prompt, decodes a cached set of samples (N for voting, n for the numeric median), and aggregates them per relation kind into an object set. It adds no trainable parameters of its own and targets the two analysis levers: abstention and numeric recall.

Scope-grounded prompts. Each relation's system message states the task, the *exact* ground-truth scope (copied verbatim from the official definitions), and the required answer form (Table 2); the model emits one JSON object `{"answers":[...]}` (Figure 2), which makes the empty case unambiguous. Few-shot exemplars come from the training split; for null-heavy relations we interleave empty and non-empty exemplars so abstention is demonstrated, not merely instructed. Even a 0.5B model emits valid JSON on 100% of instances, so format is never a bottleneck.

Calibrated abstention. Scope grounding tells the model *what* counts as an answer, but not *when* to stay silent. A bare instruction to "answer empty when unsure" is unreliable: left unconstrained, a small model names a death-city for living people and invents stock exchanges, dropping precision *below* the empty baseline (Section 7). Instead, for the three null-heavy relations we sample $N{=}3$ completions at temperature $\tau{=}0.7$ and keep an object only if it appears in at least a fraction θ of samples; single-object relations abstain entirely if even the top answer is below θ (Algorithm 1). When the model guesses, samples disagree and the prediction collapses to $\emptyset$, exactly where the metric rewards silence. We disable voting for AWARDWONBY (recall-bound) and the numeric relations.

Numeric aggregation. The numeric relations need a different aggregator: HASCAPACITY and HASAREA take a single number graded within 5%. We sample $n{=}5$ completions and take the *median* of the parsed numbers (Wang et al., 2023), which resists outlier hallucinations.

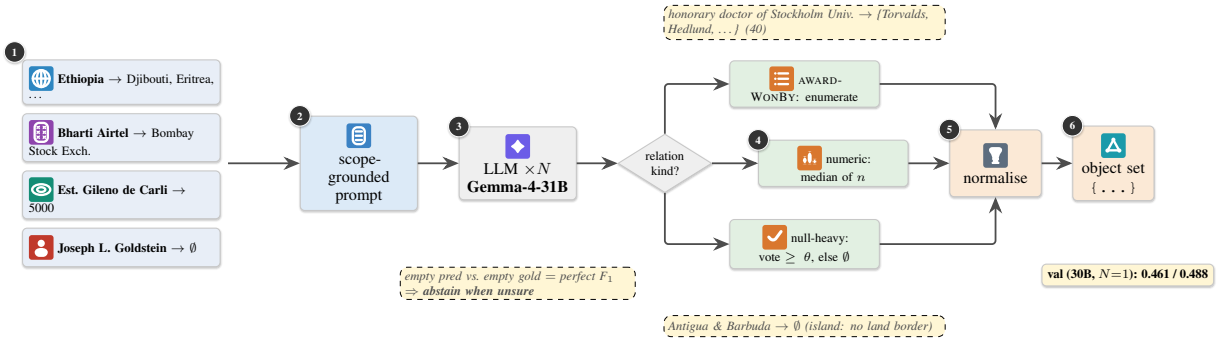


Figure 1: System pipeline (no trainable parameters). Real dataset instances (left, stage 1), shown with entity-type icons and their gold objects, are each rendered into a scope-grounded chat prompt (stage 2); the model is sampled N times (stage 3, cached), and the samples are aggregated by relation kind (stage 4): *median* for numeric relations, consistency *voting with abstention* (keep an object only if it appears in $\geq \theta$ of samples, else $\emptyset$) for the null-heavy string relations, and *enumeration* for AWARDWONBY; a final normalisation matches the evaluator exactly. Dashed callouts give worked examples of the abstention cases. The reported Gemma-4-31B score uses a single sample (N=1, voting off); the N-sample voting path is exercised by the 14B ablation (Section 7).

Relation	Scope stated in the prompt	Answer form
COUNTRYLANDBORDERSCOUNTRY	countries sharing a *land* border; maritime borders excluded; an island has none	countries, or []
PERSONHASCITYOFDEATH	the *city* where the subject died; empty if alive or the city is unknown	one city, or []
COMPANYTRADESATSTOCKEXCHANGE	stock exchange(s) on which the company is publicly listed; empty if unlisted	exchanges, or []
AWARDWONBY	every recipient of the award (often tens to hundreds)	names (long)
HASCAPACITY	maximum spectator capacity in people; highest published value	one integer
HASAREA	total surface area in km^2 (land + inland water)	one number

Table 2: Per-relation prompt specification. Each system message states the relation's exact scope (copied from the official definitions) and the required answer form; the model replies with a single JSON object `{"answers":[...]}`.

```
SYS: precise KB engine (closed book).
TASK: city where the subject died.
SCOPE: city granularity; [] if
alive/unknown.
Reply only {"answers":[...]}.
U: Egbert Mulder
A: {"answers":["Groningen"]}
U: Dave Keon
A: {"answers":[]}
U: <query>  A:
```

Figure 2: Abbreviated prompt for a null-heavy relation. Mixed exemplars teach abstention; output is forced JSON.

Robust parsing. Whatever the relation kind, every sample passes through a common final stage: a relation-aware parser recovers the answer list from noisy output (code fences, prose), strips explicit empty markers, extracts one number for numeric relations (handling commas, units, scale words), and de-duplicates. Normalisation (lower-casing, diacritic/punctuation removal) is byte-identical to the official evaluator; pushing gold answers back through the parser scores 1.000, confirming it is lossless.

5 Experimental Setup

Models. Our primary system uses **Gemma-4-31B-it** (Gemma Team, 2026) (**Gemma-4-31B** for brevity), a 30.7B-parameter open-weight model within the 32B cap. It exceeds the memory of our single 16 GB GPU, so we serve it through a hosted OpenAI-compatible API. The weights are open and we apply no fine-tuning; the endpoint serves the unmodified weights as a plain text-completion service (no retrieval, tools, or augmentation), so it is compute-only and the closed-book rules hold. For the self-consistency ablations (Section 7) we additionally run **Qwen2.5-14B-Instruct** (Qwen Team, 2024) (14.7B) locally in 4-bit. Several nominally "32B" models in fact exceed the cap (Qwen2.5-32B at 32.5B, Qwen3-32B at 32.8B) and are ineligible.

Decoding. At decode time, the pipeline selects an aggregator per relation kind (Section 4): the median over $n{=}5$ samples for numeric relations, consistency voting over $N{=}3$ samples at threshold $\theta{=}0.5$ for null-heavy relations (Algorithm 1), and a single greedy enumeration for AWARDWONBY. We evaluate these self-consistency configurations (temperature $\tau{=}0.7$) on Qwen2.5-14B, which we can sample freely. API rate limits preclude multi-sampling Gemma-4-31B at scale, so we decode it with a *single* sample per query and confine the self-consistency ablations to the 14B (Section 6). Prompts are 8-shot for every relation except AWARDWONBY (2-shot), with exemplars from the training split (Brown et al., 2020).

Hardware. Qwen2.5-14B runs on a single NVIDIA T4 (16 GB) via HuggingFace Transformers in 4-bit (`bitsandbytes nf4`, `float16` compute) (Dettmers et al., 2023); quantisation only fits the model to the hardware and does not change the counted parameter budget. Gemma-4-31B is queried over its API from the same machine. Each model processes the full 478-instance validation set.

Baselines. We compare against two references on the same validation split: the *empty* baseline, which predicts $\emptyset$ for every instance (Section 3), and the *official* organiser baseline (a few-shot Qwen3.5-9B system) whose per-relation scores we take from the task repository.

Reproducibility. The pipeline caches every raw sample, so all post-processing (parsing, consistency voting, median aggregation, and the threshold sweep) is a deterministic, GPU-free function of that cache. Every ablation in Section 7 therefore comes from a single generation pass. We release all code and prompts.

6 Results

Our primary system, **Gemma-4-31B**, scores **0.461** relation-averaged macro-F_1 (**0.488** micro) on validation, versus 0.169/0.195 for the empty baseline, 0.300/0.313 for the official baseline, and 0.415/0.442 for the smaller Qwen2.5-14B. Figure 3 and Table 3 give the per-relation picture. Gemma-4-31B beats the official baseline on *five of the six* relations, the exception being HASCAPACITY (within decoding noise). Over the 14B, its gains concentrate where the model already has usable knowledge: the biggest 14B → 30B jumps are

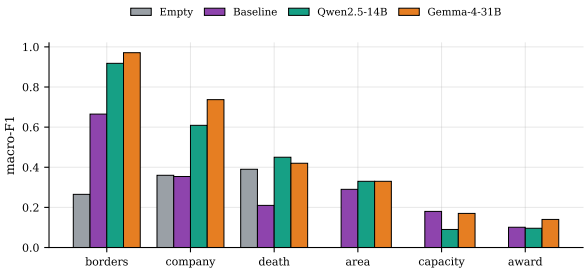


Figure 3: Per-relation validation macro-F_1: empty baseline, official baseline, Qwen2.5-14B, and Gemma-4-31B. Gemma-4-31B beats the official baseline on five of the six relations.

Relation	Empty	Base	14B	**30B**
countryLandBorders	0.265	0.665	0.918	**0.971**
companyTrades	0.360	0.354	0.609	**0.737**
personCityOfDeath	0.390	0.210	**0.450**	0.420
hasArea	0.000	0.290	**0.330**	**0.330**
hasCapacity	0.000	**0.180**	0.090	0.170
awardWonBy	0.000	0.101	0.096	**0.140**
Avg. of relations	0.169	0.300	0.415	**0.461**
Micro (all inst.)	0.195	0.313	0.442	**0.488**

Table 3: Validation macro-F_1 by model. "Empty" is the measured empty baseline (predict nothing everywhere), "Base" the official baseline (Qwen3.5-9B); **30B** is our primary Gemma-4-31B system, 14B is Qwen2.5-14B. Best per row in bold. Gemma-4-31B beats the official baseline on five of the six relations (all but HASCAPACITY).

on COMPANYTRADESATSTOCKEXCHANGE and HASCAPACITY, while the near-saturated COUNTRYLANDBORDERSCOUNTRY reaches 0.971. It ties the 14B on HASAREA and trails it on PERSONHASCITYOFDEATH, the relation where the single-sample 30B forgoes the abstention voting the 14B uses. Strikingly, the 30B attains this overall lead *without* self-consistency or abstention, from a single sample. We could not run voting on the 30B, so Section 7 isolates the decode-time levers on the 14B and confirms the ordering is not an artefact of mismatched decoding.

7 Analysis

Because the headline 30B result is single-sample, we isolate the decode-time levers on the 14B, using the same cached samples throughout.

Run-to-run variance. Decoding is stochastic ($\tau{=}0.7$), so no margin should be read before knowing how far a score moves between runs. Scoring the pipeline on each independent cached draw *separately* yields repeated single-sample runs at

no extra cost. Across them the relation-averaged macro-F_1 is 0.404 ± 0.006 (micro 0.428 ± 0.008; sd over five runs), but the per-relation spread is very uneven: COUNTRYLANDBORDERSCOUNTRY is nearly deterministic (sd 0.002), whereas HASCAPACITY ranges over 0.04–0.11 (sd 0.030) and PERSONHASCITYOFDEATH over 0.41–0.44 (sd 0.017). Small per-relation gaps in Table 3 are therefore not interpretable—the 0.03 on PERSONHASCITYOFDEATH and the 0.01 on HASCAPACITY both sit inside their own run-to-run range—while the aggregate ordering is stable: the 0.057 by which the 30B leads is roughly nine times the aggregate sd.

Backbone or decoding? Our headline comparison could conflate two factors: the 30B is decoded with a single sample, the 14B with full aggregation. Holding decoding fixed separates them. At *matched* single-sample decoding (N=1, no voting, no median) the 30B scores 0.461 and the 14B 0.404 ± 0.006, a +0.057 backbone effect; switching the 14B's full decode-time machinery on moves it from 0.404 to 0.415, a +0.011 decoding effect. The backbone gap is about five times the largest decoding gain available on the 14B, so the model-scale conclusion does not rest on the two systems having been decoded differently.

Abstention. Abstention matters most exactly where the metric rewards silence. On the three null-heavy relations, even predicting nothing beats the official 9B baseline (Section 3), so the question is whether a real model does better by abstaining *selectively* rather than never. It does: decoding the 14B with consistency voting lifts its null-heavy average from 0.642 (single sample) to 0.659 at θ=0.5. The gain is modest but consistent, and comes from precision: raising θ collapses inconsistent (usually wrong) guesses to $\emptyset$, trading recall for fewer false positives.

Setting the threshold. How high should θ be? Figure 4 sweeps it for the 14B on the null-heavy relations. Our deployed systems fix θ=0.5; the sweep is diagnostic only, run on the split we report on, so its optima are optimistic. Even so, the pattern argues for per-relation thresholds: PERSONHASCITYOFDEATH (39% empty) peaks at a low θ and falls off as voting grows stricter, while COUNTRYLANDBORDERSCOUNTRY and COMPANYTRADESATSTOCKEXCHANGE tolerate aggressive voting. Numerics differ: median aggregation is roughly neutral, so their gap is knowledge- rather than variance-bound, with area and capacity guesses often the right order of magnitude but outside the 5% band (Appendix A).

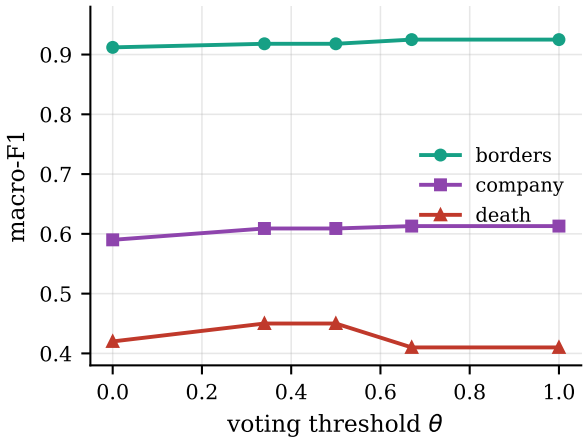


Figure 4: Per-relation validation F_1 vs. the voting threshold θ (Qwen2.5-14B) on the three null-heavy relations. The most null-heavy relation (DEATH, 39% empty) peaks at a low θ and declines as voting grows stricter, while borders and company tolerate aggressive voting, arguing for per-relation thresholds.

The AWARDWONBY recall cap. AWARDWONBY, which we beat only narrowly, is recall-capped for both models. Gold sets average ∼146 recipients (max 610) but the 14B enumerates only ∼11: its outputs are *not* truncated—it closes the JSON early, capping recall near $11/146 \approx 0.07$—and the names it lists are often plausible but wrong (P=0.39). Within a closed-book single model this relation is largely unsolved; a larger in-budget model or multi-step decomposition would help most.

8 Conclusion

We treated *when to answer* as the central decision in closed-book, set-valued KB construction: because macro-F_1 rewards a correct empty set, abstention pays off, numeric recall is scarce, and scope grounding curbs over- and under-prediction. With Gemma-4-31B it reaches 0.461/0.488 (relation-averaged/micro), beating the official baseline on five of six relations, reproducibly from a cached pass. Comparatively, a single-sample 30B outscores a voted 14B, so pairing the two is the clearest next step.

Limitations

No held-out evaluation. Every number we report is measured on validation. We did not obtain an official test score: predictions for the 477 test instances were generated and are released with our code, but they were not scored on the shared task's evaluation server, and the test gold labels are not public, so we cannot score them ourselves. Our results are therefore not independently verifiable, and since the validation split informed our design decisions, they should be read as an optimistic estimate of held-out performance rather than a held-out measurement. This bears most on the comparisons we draw between models: the ranking we report may not survive on test. The caveat applies with more force to the threshold sweep (Figure 4), which is run on the very split it is scored on, so its per-relation optima are diagnostic only; our deployed systems avoid tuning on it by fixing θ=0.5 a priori.

Decoding variance. We measure run-to-run variance for the 14B only (Section 7), by scoring each cached draw separately; this covers five of the six relations, since AWARDWONBY was decoded once. The estimate uses three to five runs, so the per-relation standard deviations are themselves noisy, and AWARDWONBY has only 10 subjects. Because the 30B is single-sample we cannot measure its variance directly and assume it is comparable. Small per-relation deltas should accordingly be read as indicative only.

Model coverage. Being closed-book, the system inherits the base model's coverage and recency limits and cannot recover facts the model never stored. This caps AWARDWONBY recall and rare-venue numeric accuracy, and it means our gains reflect parametric recall rather than any external grounding.

Single-sample 30B. Because API rate limits precluded multi-sample decoding, our Gemma-4-31B scores are single-sample. We therefore cannot report how the 30B behaves under the self-consistency and abstention voting that help the smaller models; pairing them is left to future work and may further lift the null-heavy and numeric relations, so our 30B numbers are likely a floor rather than a ceiling.

Metric specificity. The abstention advantage stems from this task's macro metric and its empty-as-valid-answer setting. It may not transfer to single-answer or micro-only regimes, and practitioners porting the method to other benchmarks should re-verify that abstention is still a positive-value action before adopting the voting machinery.

Ethics and Generative-AI Declaration

Closed-book KB construction surfaces facts from model parameters that may be stale, biased, or hallucinated. Predictions about people (e.g. place of death) can be wrong, so they should not be treated as authoritative without verification. Our abstention mechanism partly mitigates this: when the model is uncertain it emits an empty answer rather than a confident fabrication. The only generative-AI component of the *system* is the open-weight LM that produces the predictions, as the task requires: no proprietary assistant was used to generate, aggregate, or score any result. An AI coding assistant was used while drafting and editing this manuscript and its accompanying code, as the public commit history records; all experiments, numbers, and claims were designed, run, and verified by the author, who takes full responsibility for the content.

References

Dimitrios Alivanistos, Selene Báez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. In *Semantic Web Challenge on KB Construction from Pre-trained Language Models (LM-KBC)*.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *NeurIPS*.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021.

Knowledgeable or educated guess? revisiting language models as knowledge bases. In *ACL*.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Jeremy R. Cole, Michael J.Q. Zhang, Daniel Gillick, Julian Martin Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. 2023. Selectively answering ambiguous questions. In *EMNLP*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *NeurIPS*.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*.

Yonatan Geifman and Ran El-Yaniv. 2017. Selective classification for deep neural networks. In *NeurIPS*.

Gemma Team. 2026. Gemma 4. Google DeepMind. Open-weight models under the Apache 2.0 license; we use the 31B dense variant. Model card: https://ai.google.dev/gemma/docs/core/model_card_4.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *ICML*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *ICML*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the ACL*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2024. Preface: Knowledge base construction from pre-trained language models (lm-kbc) 2024. In *LM-KBC Challenge at ISWC, CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pre-trained language models. In *LM-KBC Challenge, CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Sneha Singhania, Simon Razniewski, and Jeff Z. Pan. 2023. LM-KBC 2023: The 2nd challenge on knowledge base construction from pre-trained language models. In *Knowledge Base Construction from Pre-trained Language Models (LM-KBC), CEUR Workshop Proceedings*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *ICLR*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating the effectiveness of parametric and non-parametric memories. In *ACL*.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *EMNLP*.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023. Large language models and knowledge graphs: Opportunities and challenges. *Transactions on Graph Data and Knowledge (TGDK)*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*.

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *ACL*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *EMNLP*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10).

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Bohui Zhang, Ioannis Reklos, Nitisha Jain, Albert Meroño-Peñuela, and Elena Simperl. 2023. Using large language models for knowledge engineering (LLMKE): A case study on wikidata. *arXiv preprint arXiv:2309.08491*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *NAACL*.

A Qualitative Examples

Qualitative behaviour. Table 4 shows representative Gemma-4-31B outputs: exact on well-known facts and near-complete on borders, with the two error modes above visible, a numeric near-miss on HASAREA and plausible-but-wrong famous names on AWARDWONBY.

Relation	Subject	Gold (abbrev.)	Gemma-4-31B output	
borders	Ethiopia	Djibouti, Eritrea, Kenya, Somalia, S. Sudan, Sudan	Djibouti, Eritrea, Kenya, Somalia, S. Sudan	✓ 5/6
death	Vito Acconci	New York City	New York City	✓
company	Bharti Airtel	Bombay Stock Exch., NSE of India	Bombay Stock Exchange, NSE of India	✓
capacity	Gileno de Carli	5000	5000	✓
area	Lošinj	74.36	123.3	×
award	Stockholm hon. dr	Torvalds, Hedlund, Trebst, . . . (40)	Mandela, Tutu, Annan, Yunus, . . .	×

Table 4: Qualitative examples: Gemma-4-31B vs. gold. Exact hits on PERSONHASCITYOFDEATH/HASCAPACITY, near-complete borders; the numeric near-miss (HASAREA) and plausible-but-wrong enumeration (AWARDWONBY) are the knowledge-bound errors discussed above.

A Forced Answer Channel for Closed-Book Knowledge Base Construction at LM-KBC 2026

Yiqun Sun
Magellan Technology Research Institute (MTRI)
duke.sun@mtri.co.jp

Pengfei Wei
Magellan Technology Research Institute (MTRI)
pengfei.wei@mtri.co.jp

Lawrence B. Hsieh
Magellan Technology Research Institute (MTRI)
lawrence.hsieh@mtri.co.jp

Abstract

The AKBC 2026 shared task (Kalo et al., 2026) asks a system to complete the object set of a subject and a relation out of a language model's parameters, with no retrieval, no training, and no more than 32B parameters. Our system drives one 29.6B backbone through three stages. Several elicitation channels per relation build a candidate pool, exhaustive one-token Yes/No probes verify whatever part of the answer space can be enumerated, and a per-relation decision layer emits the object set. The backbone writes a reasoning channel before its user-facing answer, so a probe prompt has to append the answer-channel header before its first generated token is a verdict. Forcing that channel also makes auxiliary generations cheap, which is what puts roughly 26k border probes inside the budget. Most decision constants are fitted on validation gold under a rule that displaces an incumbent only on a strict improvement of more than one row, and the fitting procedure itself is cross-validated, which cut one city gain from $+0.070$ to $+0.020$ before it was adopted. The system scores 0.6553 official macro-F1 against 0.6415 for a prompting baseline on the same backbone, and the gap is uneven: the machinery is worth $+0.1662$ on awards and costs 0.0204 on `hasCapacity`, where plain prompting wins. We also report the ceiling we could not move. Twenty-one method families have been fitted against `hasCapacity` and none of them beats a mass vote over the pool. Harvesting the model's own reasoning traces raises pool coverage from 0.582 to 0.724 and converts none of it.

1 Introduction

The LM-KBC challenge series (Kalo et al., 2023, 2024, 2025), whose 2026 edition is the AKBC shared task (Kalo et al., 2026), asks a language model to serve as the knowledge base itself (Petroni et al., 2019; Roberts et al., 2020). A system receives a subject entity and a relation and returns the complete object set, which may be empty, a single value, or several hundred names. The 2026 edition forbids retrieval at inference and training of any kind, and it caps the total neural budget at 32B parameters. A score therefore measures two things at once, what a backbone holds and what the machinery around it can pull out.

The system uses one model for every stage. Muse-Glimmer-30B fits the budget on its own at 29.6B parameters, so the pipeline contains no second network and no cross-model agreement term, and none of the heterogeneous multi-model orchestration that knowledge-intensive pipelines usually reach for (Du et al., 2023; Feng et al., 2026b). Per subject the backbone runs several elicitation channels into a candidate pool, one-token Yes/No probes verify candidates wherever the answer space can be enumerated, and a per-relation decision layer turns pool and probes into an object set (§3). The system scores 0.6553 official macro-F1 over the 475 test rows, against 0.6415 for self-consistent prompting of the same backbone with none of the machinery attached.

The mechanism that shapes the rest of the design is a property of the backbone. It answers in two channels, a reasoning channel followed by a user-facing answer channel, and a generation prompt lands the model in the first of them. A one-token Yes/No probe therefore reads the opening token of deliberation. Appending the answer-channel header to the prompt makes the first generated token the verdict, which is what turns verification into a single forward pass and puts roughly 26k border probes inside the compute budget.

The protocol behind the decision constants is not specific to this task (§4). Constants are fitted on validation gold, a challenger displaces an incumbent only on a strict improvement of more than one row, and the fitting procedure itself is cross-validated. That last step matters, because an in-sample sweep of the city rule preferred a setting worth $+0.070$

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 427–433
October 28, 2026 ©2026 Association for Computational Linguistics

on validation and cross-validating the procedure deflated it to $+0.020$. Neither is the negative program (§6). Twenty-one method families have now been fitted against `hasCapacity`, spanning selection statistics, probe reformulations, representation read-outs and fresh elicitation surfaces, and the diagnostics say where the wall is. Code, prompts and the elicitation checkpoints behind the scored predictions are released at `https://github.com/dukesun99/elicitation-beats-selection`.

2 Task and metric

Six relations are scored over 475 test rows. Three are nullable string relations, `countryLandBordersCountry` (67 rows, *borders* below), `personHasCityOfDeath` (100 rows, *city*) and `companyTradesAtStockExchange` (100 rows, *company*). `awardWonBy` (10 rows, *award*) asks for recipient sets that run to hundreds of names. `hasCapacity` (98 rows, *capacity*) and `hasArea` (100 rows, *area*) are numeric and are scored within a 5% relative tolerance. The metric is per-row F1 macro-averaged over rows with alias-aware normalization, and an empty prediction against an empty gold set scores 1.0. Rows and not relations are averaged, so a relation's weight is its row count. Decision constants are fitted on the train and validation splits. No retrieval runs at inference, no external resource enters any prompt, and no parameter of the backbone is trained, adapted or otherwise updated at any point. Every test number below is official.

3 System

Backbone. Muse-Glimmer-30B carries 29.6B parameters including a 1.8B vision encoder and is released under Apache-2.0. It fits the 32B budget alone, which removes the usual option of pairing a large generator with a small verifier. Sampling channels keep the reasoning channel, because that is where the model's factual advantage lives (Wei et al., 2022), and probes discard it by forcing the answer channel open, which is the same trade a shortcut decoder makes for cost (Liu et al., 2026). The chat template defaults to high reasoning effort, which silently truncates long elicitation prompts unless the effort is lowered. Two failures follow from ignoring either property. Unforced probing returns the first token of deliberation and ranks a living person as more likely deceased than a long-dead one, with no error raised. Leaving the effort high drops the parse rate far enough to look like a capability result.

Elicitation. Each subject draws $k{=}12$ samples from a few-shot prompt that carries the relation definition verbatim, a JSON output schema and training exemplars, including empty-gold exemplars wherever the relation permits an empty answer (Wang et al., 2022; Jiang et al., 2020). Auxiliary channels approach the same fact from another angle at $k{=}8$, all of them forced into the answer channel, which is what makes them affordable. Varying the frame, and not sampling one frame harder, is what widens the pool, the same reason diversity-aware retrieval covers an event that relevance ranking alone leaves half-told (Tang et al., 2025a). The numerics add a recitation channel, a commonly-cited-figure frame and a wiki-table completion frame (Elazar et al., 2021). City adds an obituary recitation, a local-language frame (Kassner et al., 2021; Zhao et al., 2026) and a presupposition frame that asserts the death and asks only for the locality. Award uses a list enumeration. Borders runs two list passes at different sampling depths. Greedy channels run at $k{=}1$ and temperature 0.

Verification. Where the answer space can be enumerated, exhaustive one-token probing (Kadavath et al., 2022; Sun et al., 2026b) replaces open-ended generation. The model writes the universe first, the world's sovereign states for borders and its stock exchanges for company, keeping a name that recurs across samples. Borders then probes every subject-country pair in both directions, about 26k probes in total. Company probes every subject-exchange pair and debiases each score in log-odds against a placebo built from training companies with empty gold, which removes the model's standing tendency to answer Yes (Zhao et al., 2021). City gets an alive-deceased gate from a single forced token. Capacity scores each cluster representative by its pointwise mutual information against a generic-venue prefix (Holtzman et al., 2021).

Decision layer. String relations fuse sampling frequency with probe score under a per-relation threshold, reconciling what the model generates with what it will affirm, in the manner of a second-pass rescorer over an n-best list (Kadavath et al., 2022; Tang and Tung, 2024; Tian et al., 2025). Borders admits a country at $0.7\,\mathrm{freq} + 0.3\,\mathrm{ver} \geq 0.40$ with the deeper of the two list passes weighted 3, and company keeps its released fusion. City pools the direct channel with its recitation channel

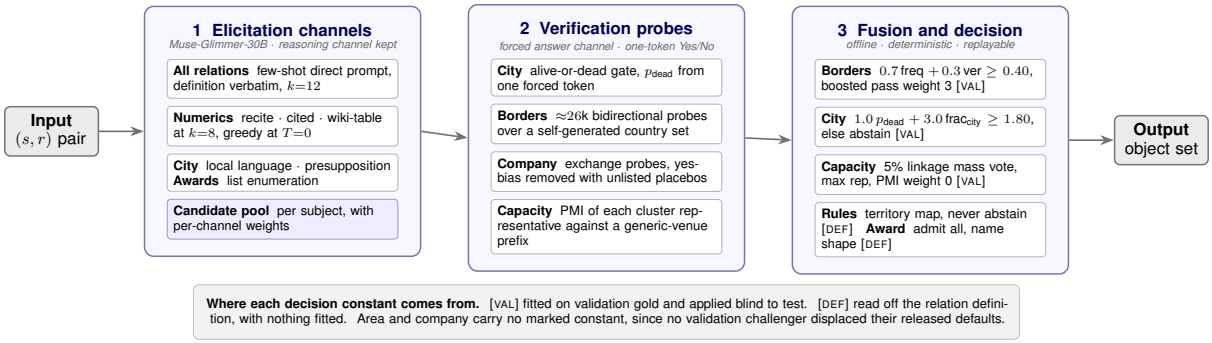


Figure 1: The pipeline, and where each group of decision constants came from (§4).

at double weight, answers with the most frequent candidate when the deceased probability plus three times the fraction of samples naming a city clears 1.80, and abstains otherwise. Numerics cluster the pooled values by single linkage and take the largest cluster's maximum, at a 5% linkage width for capacity. The capacity PMI term is computed and then weighted at zero, which is the setting validation preferred. Definition-derived rules close the pipeline, mapping integral territories to their state so that France borders Suriname through French Guiana, and filtering award candidates by name shape. Numerics and award never abstain. Everything downstream of the elicitation checkpoints is deterministic and replays offline.

4 Fitting the decision constants

Most decision constants are fitted on validation gold and applied to test without adjustment, and the exceptions are named at the end of this section. The adoption rule is what makes that fitting trustworthy. A challenger displaces the incumbent only when it improves validation by more than one row, since a margin of one row is inside the resolution of a 100-row relation. Four candidates were declined under that rule, two of them on area (the clustering tolerance together with the representative choice, and the channel weights), one on company fusion, and one capacity setting that merely tied. A tie does not displace.

City needed more than a threshold on the sweep. The in-sample sweep preferred a rule worth $+0.070$ on validation, which is large enough for the winner's curse to be the leading explanation (Blum and Hardt, 2015; Dwork et al., 2015). Cross-validating the fitting procedure, so that each fold reran the entire sweep and the fold's own winner was scored on the held-out half, deflated the gain to $+0.020$, and the cross-validated setting is the one the system uses. Both folds independently dropped the presupposition channel, which is why that part of the choice is trusted. Cross-validating the procedure and not the fitted rule is what separated the real effect from the noise around it.

The largest gain came from a knob nobody had turned. Channel weights had never been swept anywhere in the system. They were frozen once and carried forward while decision rules were reworked around them. Refitting them produced a larger gain than any decision-rule change we tried. Capacity rose 0.031 on test from two validation-derived choices, switching the PMI term off and doubling the weight of the cited, greedy and wiki-table channels. Both choices remove fitted structure instead of adding more of it.

Not every constant comes from validation gold, and the figure marks the difference. Area and company keep their released defaults, because no validation challenger strictly improved on them. Award carries no fitted constant at all. Its rule admits every distinct candidate the enumeration channel names, subject to a name-shape filter and the never-abstain behaviour, and both of those are read off the relation definition, which asks for recipient entities and allows large object sets. That is the whole rule, and it needs no fitting ground, which is fortunate: validation and test awards differ in shape, with validation dominated by fellowship-style gold sets and test by annual prizes, so the validation split would have been a poor one.

5 Results

The system scores 0.6553 official macro-F1, with macro precision 0.7842 and macro recall 0.6597 (Table 1).

The per-relation profile separates cleanly into three regimes. Borders and area sit above 0.85, and both are relations where the answer space is

Relation	rows	prompting	system
borders	67	0.9346	0.9481
area	100	0.8300	0.8600
company	100	0.7838	0.7838
city	100	0.5400	0.5700
capacity	98	0.2653	0.2449
award	10	0.0696	0.2358
all rows	**475**	**0.6415**	**0.6553**

Table 1: Official per-relation test F1 for the prompting baseline and for the full system. The system's macro precision over all rows is 0.7842 and its macro recall is 0.6597.

small enough to be swept exhaustively or resolved by agreement across channels. Company and city sit in the middle, where the binding constraint is a decision about what to admit and what to leave out. Capacity and award sit at the bottom, and §6 gives the measurements behind the capacity floor, which no rule in twenty-one method families has moved. Precision runs well ahead of recall over the whole test set, which is the operating point per-row F1 rewards, because filling a gold-empty row with a plausible object costs the full row while one recovered object gains a fraction of it.

Prompting baseline. The reference point is the same backbone with none of the machinery around it. It reads the k=12 direct samples and aggregates them by self-consistency under rules taken from the task definition instead of fitted. Numerics cluster the parsed values at the metric's own 5% tolerance and answer with the largest cluster's median, city takes a majority over parsed cities including the null answer, and the set-valued relations admit an object named in more than half of the samples that parsed. No auxiliary channel, no probe, no gate and no fitted constant enters it. That baseline scores 0.6415, so three stages of elicitation, verification and fitted decision are worth $+0.0138$ over prompting the same model.

The gain is concentrated, not broad. Award moves $+0.1662$, though on 10 rows, city and area each move $+0.0300$, and borders moves $+0.0135$. Two rows go the other way, and both are worth stating plainly.

Where the machinery does not pay. Capacity is 0.0204 *worse* under the full system than under plain prompting. The auxiliary channels and the max-representative rule are net harmful on the relation that most of this paper's negative program is about, and a system built to elicit and select loses there to a median over direct samples. The choice was not made with test labels in view. On validation the system's capacity configuration scores 0.3505 against the baseline's 0.3196, so the validation signal pointed the other way and the protocol of §4 kept the fitted setting. We report the relation as the system produced it, and record that validation did not predict the test outcome here. Company is the second row: it gains exactly 0.0000. The exchange probe universe and the placebo correction recover to four decimal places what the direct channel already says, so that relation carries verification machinery it does not need.

6 What bounds the score

Capacity has defeated 21 method families. On the selection side these include self-verification, pairwise duels, multiple choice, PMI ranking, cross-model and cross-language agreement, activation read-outs (Burns et al., 2022; Li et al., 2023; Wang et al., 2026; Wu et al., 2025), task-vector steering (Hendel et al., 2023; Todd et al., 2024; Sun et al., 2026a), multi-agent debate (Du et al., 2023; Feng et al., 2026a), complete-linkage clustering, nine hand-designed selection statistics, channel-subset search, a highest-corroborated-value rule, ordinal threshold inversion, channel-breadth scoring, within-cluster representative choice and the per-relation sampling budget. On the elicitation side they include volume boosts, higher reasoning effort, multilingual prompts, explicit maximum-configuration prompts, prompt surfaces that imitate Wikidata property syntax, which carry no external data and only reframe the question, and reasoning-trace harvesting.

Four measurements say the most about why. Harvesting the model's own reasoning traces (Lu et al., 2025; Turpin et al., 2023) raises the fraction of capacity subjects whose pool holds a value within tolerance from 0.582 to 0.724 and converts none of it, in any confidence stratum, at any validation-fitted weight. Treating the task as ordinal inversion and probing thresholds directly fails on calibration (Zhao et al., 2021; Tang et al., 2021), since 23% of adjacent thresholds violate monotonicity and the model gives a randomly chosen venue a 28% chance of holding at least 150,000 people, with direct readout landing within tolerance on 4 of 98 rows. Hedging with a second value pays only when hit@2 exceeds 1.5 times hit@1, and the measured ratio is 1.20. Raising the sampling budget

does nothing, with the validation capacity curve flat from k=6 at 0.2809 to k=12 at 0.2784.

The error structure explains the pattern. Gold is present in the pool for about two thirds of capacity rows and holds a median 12% of the cluster mass when it is present, and the errors are bimodal at roughly a factor of two, the signature of confusing a seated figure with a total or a current configuration with a historical maximum. No statistic computed from the generation distribution finds a signal that thin, which is why raising coverage without changing the shape of the mass buys nothing (Kandpal et al., 2023; Xu et al., 2024).

The other two relations are bounded differently, and both bounds are measured. City's gold-empty rows are living people, so abstention on that relation is alive-dead classification. The deceased-gate probe separates the two classes at AUC 0.745 and no alternative detector beat it, which caps how far the abstention rule can move before it starts trading correct empties for wrong cities. Company recall is bounded by secondary listings the sampler never names, an omission a confident-looking answer set does not reveal (Tang et al., 2025b; Kandpal et al., 2023). A directed secondary-listing channel does surface them on validation and still loses under cross-validation, because the same channel names plausible exchanges for companies that hold no secondary listing at all.

7 Conclusion

One 29.6B backbone, multi-channel elicitation, exhaustive one-token verification made affordable by forcing the model's answer channel open, and a small per-relation decision layer reach 0.6553 on the LM-KBC 2026 test set, 0.0138 above self-consistent prompting of the same backbone. The constants that were fitted came from validation gold under a strict-improvement rule, with the fitting procedure itself cross-validated, and the single largest gain came from channel weights that had never been swept. What the system cannot do is convert pool coverage into score on `hasCapacity`, where the correct value is present in about two thirds of pools, holds a median 12% of the cluster mass, and has resisted 21 families of method. On that relation the machinery is not merely stuck, it is behind the baseline it was built to beat.

Limitations

The negative results were measured on one backbone inside one size band, and a model with a different pretraining mixture may put the capacity mass somewhere a selection statistic can reach. Individual per-relation deltas are small against the resolution of a 100-row relation, so a single component is rarely settled on its own by this metric (Reimers and Gurevych, 2017; Dodge et al., 2019). The comparison against the prompting baseline inherits that resolution, so the two relations where the machinery fails to pay, capacity at -0.0204 and company at 0.0000, are better read as evidence that those components earn nothing than as a calibrated size for the loss. Award carries 10 of 475 rows, and its gold sets run to hundreds of names, so conclusions about that relation are the weakest in the paper, and its rule is the least constrained by data of any in the system.

References

Avrim Blum and Moritz Hardt. 2015. The ladder: A reliable leaderboard for machine learning competitions. In *Proceedings of ICML*.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of EMNLP-IJCNLP*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. 2015. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Yang Feng, Ziwei Xu, Xia Hu, and Fengxiang He. 2026a. PEAR: Permutation-equivariant adaptive routing multi-agent debate. *arXiv preprint arXiv:2606.20621*.

Yichao Feng, Haoran Luo, Zhenghong Lin, Yiqun Sun, Pengfei Wei, Lawrence B. Hsieh, and Anh Tuan Luu. 2026b. OrchMAS: Orchestrated reasoning with multi collaborative heterogeneous scientific expert structured agents. *arXiv preprint arXiv:2603.03005*.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of EMNLP*.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of EMNLP*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2024. Preface: LM-KBC challenge 2024. In *Joint Proceedings of the KBC-LM Workshop and the LM-KBC Challenge*, volume CEUR Vol-3853.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pre-trained language models. In *Joint proceedings of KBC-LM and LM-KBC @ ISWC*, volume CEUR Vol-4041.

Jan-Christoph Kalo, Sneha Singhania, Simon Razniewski, and Jeff Z. Pan. 2023. LM-KBC 2023: 2nd challenge on knowledge base construction from pre-trained language models. In *Joint proceedings of KBC-LM and LM-KBC @ ISWC*, volume CEUR Vol-3577.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of ICML*.

Nora Kassner, Philipp Dufter, and Hinrich Schütze. 2021. Multilingual LAMA: Investigating knowledge in multilingual pretrained language models. In *Proceedings of EACL*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*.

Fan Liu, Yanhao Wang, Min Zhang, Zhikang Chen, Zeyuan Li, Lewei He, and Jiahui Pan. 2026. Think faster than words: Efficient LLM chain-of-thought reasoning via dynamic shortcut decoding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28825–28836. Association for Computational Linguistics.

Jiahao Lu, Ziwei Xu, and Mohan Kankanhalli. 2025. Reasoning LLMs are wandering solution explorers. *arXiv preprint arXiv:2505.20296*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of EMNLP*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of EMNLP*.

Yiqun Sun, Junyu Chen, Pengfei Wei, and Lawrence B. Hsieh. 2026a. GGSS: Geodesic-gated spherical steering for inference-time debiasing of generative vision-language models. *arXiv preprint arXiv:2608.25375*.

Yiqun Sun, Pengfei Wei, and Lawrence B. Hsieh. 2026b. Very efficient listwise multimodal reranking for long documents. *arXiv preprint arXiv:2605.11864*.

Yixuan Tang, Hwee Tou Ng, and Anthony K. H. Tung. 2021. Do multi-hop question answering systems know how to answer the single-hop sub-questions? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3244–3249. Association for Computational Linguistics.

Yixuan Tang, Yuanyuan Shi, Yiqun Sun, and Anthony K. H. Tung. 2025a. Uncovering the bigger picture: Comprehensive event understanding via diverse news retrieval. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33939–33957. Association for Computational Linguistics.

Yixuan Tang and Anthony K. H. Tung. 2024. Contextualized speech recognition: Rethinking second-pass rescoring with generative large language models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6478–6485.

Yixuan Tang, Jincheng Wang, and Anthony K. H. Tung. 2025b. The missing parts: Augmenting fact verification with half truth detection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 33979–33996. Association for Computational Linguistics.

Yang Tian, Fan Liu, Jingyuan Zhang, Victoria W., Yupeng Hu, and Liqiang Nie. 2025. CoRe-MMRAG: Cross-source knowledge reconciliation for multimodal RAG. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32967–32982. Association for Computational Linguistics.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. 2024. Function vectors in large language models. In *Proceedings of ICLR*.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36.

Hao Wang, Yiqun Sun, Pengfei Wei, Lawrence B. Hsieh, and Daisuke Kawahara. 2026. Sparse autoencoders as plug-and-play firewalls for adversarial attack detection in VLMs. *arXiv preprint arXiv:2605.07447*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2025. AxBench: Steering LLMs? even simple baselines outperform sparse autoencoders. *arXiv preprint arXiv:2501.17148*.

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817*.

Jiaqi Zhao, Qiang Huang, Haodong Chen, Xiaoxing You, and Jun Yu. 2026. When abundance conceals weakness: Knowledge conflict in multilingual models. *arXiv preprint arXiv:2601.07041*.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of ICML*.

REAP: Relation-Aware Elicitation and Parsing for Closed-Book Knowledge Base Construction from LLMs

Thanh-Dan Bui **Thanh-Trung Do** **Tuan-Phong Nguyen***
VNU University of Engineering and Technology, Hanoi, Vietnam
{23020342, 24022472, tuanphong}@vnu.edu.vn

Abstract

We present the REAP system for the AKBC Shared Task 2026 on constructing knowledge bases from language models in a closed-book setting, subject to a budget of at most 32B parameters and no model fine-tuning. Our system combines structured chain-of-thought reasoning, relation-specific query strategies, and a reasoning-based empty-set gate to elicit parametric knowledge, followed by direct extraction into valid JSON arrays. On the official test set, the system, built on Mistral-Small-24B-Instruct-2501, achieves a macro-F1 score of 0.62, with particularly strong results on `countryLandBordersCountry` ($F_1 = 0.95$), `companyTradesAtStockExchange` ($F_1 = 0.73$), and `hasArea` ($F_1 = 0.77$). Our code is publicly available.[1]

1 Introduction

Large language models (LLMs) encode substantial factual knowledge in their parameters (Petroni et al., 2019). However, eliciting this knowledge for knowledge base construction (KBC) is considerably more challenging than answering isolated factual questions: a subject–relation pair (s, r) may correspond to an empty ($\emptyset$), single-valued (1), or multi-valued (N) object set. The AKBC Shared Task 2026 (Kalo et al., 2026) formalizes this problem under a constrained setting: systems must operate in a closed-book fashion, without retrieval-augmented generation or external knowledge; use at most 32B parameters; and perform no model fine-tuning.

In this task, each input record specifies a subject entity and one of the six given relations (see Table 1), and the system must return the complete set of corresponding objects as a valid JSON array. The relations cover geographic adjacency and area, a person's city of death, venue capacity, award recipients, and the stock exchanges on which a company is traded. Consequently, the required output may be empty, contain a single object, or contain a long list of objects, while quantitative answers must also be sufficiently precise. Table 1 summarizes the six target relations and the scope of their ground-truth labels.

Earlier factual probing methods use cloze-style prompts or prompt ensembles to rank candidate objects (Petroni et al., 2019; Jiang et al., 2020), while the LM-KBC challenge series additionally requires systems to materialize variable-cardinality answer sets (Singhania et al., 2022; Kalo et al., 2023, 2024, 2025). A direct generative solution that asks the model to recall all objects and produce correctly formatted JSON in a single pass can therefore hallucinate objects, omit valid answers, or violate the output schema.

To address this issue, we propose REAP (**R**elation-aware **E**licitation **A**nd **P**arsing), a two-stage pipeline that decouples factual elicitation from answer serialization. Stage 1 prompts the LLM using strategies tailored to each relation, whereas Stage 2 uses deterministic JSON parsing when possible and LLM-based extraction only as a fallback. This separation allows the model to focus on recalling the answer set before the system enforces the required JSON format.

Our main contributions are as follows:

1. **Relation-Aware Elicitation**: We design specialized strategies for different relation types, including CoT reasoning with an empty-set gate, a four-direction geographical scan for land borders, and chronological multi-pass querying for awards.

2. **Efficient Hybrid Parsing**: Deterministic parsing handles approximately 80% of the records, while LLM-based extraction is re-

* Corresponding author.

[1] https://github.com/yammdd/AKBC-Shared-Task-2026

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 434–440
October 28, 2026 ©2026 Association for Computational Linguistics

Relation	Ground-truth scope and definition
`countryLandBordersCountry`	Countries that share a land border with the subject (maritime borders excluded; island countries $\rightarrow \emptyset$).
`personHasCityOfDeath`	The city in which the person died (living people or cases with an unknown city $\rightarrow \emptyset$).
`hasCapacity`	The maximum spectator capacity of the venue, represented as an integer.
`awardWonBy`	Entities, including individuals and organizations, that have received the corresponding award.
`companyTradesAtStockExchange`	Stock exchanges on which the company's shares are publicly traded (private companies $\rightarrow \emptyset$).
`hasArea`	The total surface area (land + inland water) of the geographical entity, measured in km^2.

Table 1: Definitions and ground-truth scope of the six target relations.

served for complex outputs, substantially reducing the overall computational cost.

3. **Scalable Inference and Evaluation**: Batched inference with vLLM on TPU hardware enables fast end-to-end processing. The complete system achieves macro-F1 0.62 on the test set and highlights the limits of parametric knowledge for rare entities.

2 Related Work

Probing knowledge in language models. LAMA (Petroni et al., 2019) established cloze-style probing but assumes exactly one object per relation, and what can be extracted depends heavily on the prompt (Jiang et al., 2020); see AlKhamissi et al. (2022) for a survey. The LM-KBC challenge series (Singhania et al., 2022; Kalo et al., 2023, 2024, 2025) recast this as knowledge base construction with arbitrary-cardinality object sets (0, 1, or N), the setting continued by the AKBC Shared Task 2026 (Kalo et al., 2026). Prior systems pair LLM probing with entity mapping (Zhang et al., 2023); most recently, ReWiSe (Albert-Roulhac and Zouaq, 2025), a top-performing system at LM-KBC 2025, combines chain-of-thought (CoT) reasoning with relation-wise self-consistency. In REAP, we adopt the idea of relation-specific CoT and explicit empty-set handling, but drop self-consistency voting, which is costly and can be dominated by confident-but-incorrect reasoning chains.

Parametric knowledge and its limits. Closed-book QA shows that language models can answer factual questions without retrieval (Roberts et al., 2020), motivating the closed-book setting we adopt. This parametric knowledge is uneven, however: models recall popular facts well but struggle with long-tail entities (Kandpal et al., 2023; Mallen et al., 2023), which bounds any closed-book system and matches our error analysis on rare subjects.

Step-by-step reasoning and empty-set gate. Chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022) improves multi-step reasoning and builds on the in-context learning ability of instruction-tuned LLMs (Brown et al., 2020). Since a correct answer may itself be the empty set, in REAP, we add a *reasoning-based empty-set gate* inspired by selective prediction (Kamath et al., 2020). Unlike calibrated methods that estimate whether a model knows an answer (Kadavath et al., 2022), our gate is a prompt-driven decision that returns an empty set as the final answer when the property does not exist or the evidence of it is insufficient.

3 Method

3.1 Architecture Overview

The REAP system consists of two sequential stages (Figure 1). In Stage 1, we use an LLM to *generate evidence* as free-form text for multi-valued relations or use CoT reasoning for single-valued relations. If the generation phase produces unparsable output, the system automatically retries or switches to a fallback prompt. Stage 2 *extracts* the evidence into JSON arrays using regular expressions; for multi-valued relations that cannot be handled reliably by simple rules, the LLM performs the extraction itself.

3.2 Relation-Specific Prompt Design

Stage 1 prompts are specialized per relation following three principles: (i) **query decomposition** to maximize recall on multi-valued relations (e.g., six decade-scoped passes for `awardWonBy` and a four-direction geographi-

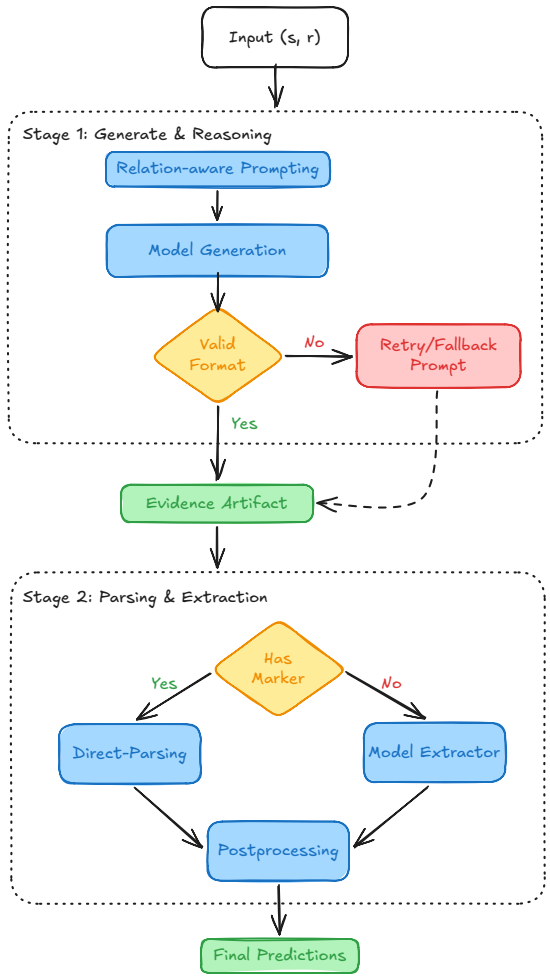


Figure 1: The architecture of REAP.

cal scan for `countryLandBordersCountry`); (ii) **output-format constraints** via `BORDERS:` or `FINAL_ANSWER: [...]` markers; and (iii) **step-by-step reasoning with an empty-set gate** that outputs `FINAL_ANSWER: []` for unanswerable entities (e.g., living individuals or private companies). Together, these relation-tailored, multi-step CoT procedures constitute the *structured reasoning* that Stage 1 performs before extraction.

Few-shot demonstrations are static exemplar pairs (one positive, one empty-set case) fixed per relation to illustrate the reasoning chain and empty-set triggering. Crucially, all exemplars are drawn exclusively from `train.jsonl` or hand-written triples, ensuring zero data leakage against the evaluation splits.

`awardWonBy`: Chronological Multi-Pass. For major awards with dozens or hundreds of recipients, a single query typically yields low recall. We therefore use a multi-pass strategy comprising one general prompt and five auxiliary prompts decomposed chronologically: from the award's inception through the 1970s, the 1980s–1990s, the 2000s–2010s, 2020 onward, and a final pass targeting less prominent or non-Western recipients. All six generations use temperature $T = 0.7$ to diversify recall before the evidence is merged.

`countryLandBordersCountry`: Four-Direction Geographical Scan. The prompt instructs the model to act as a geography expert and inspect the subject's eastern, western, southern, and northern boundaries. This scan reduces omissions of small neighboring countries and adjacent detached territories. The prompt includes land borders only, explicitly excludes maritime boundaries, and returns `BORDERS: NONE` for island countries. If the response lacks the `BORDERS:` marker, the system automatically retries with a shorter fallback prompt.

`companyTradesAtStockExchange`: Four-Step Listing Check. The model follows a four-step CoT procedure: (1) identify the company; (2) apply a **Strict Public Check (empty-set gate)** to determine whether the entity is a private company, a nonprofit organization, or a subsidiary that is not separately listed, returning `FINAL_ANSWER: []` if any condition holds; (3) retrieve all listing exchanges using their full official names, such as *New York Stock Exchange* or *SIX Swiss Exchange*; and (4) conclude with `FINAL_ANSWER: ["Exchange Name"]`.

`hasArea`: Four-Step Entity Disambiguation and Unit Normalization. The model follows four CoT steps: (1) distinguish the target entity (e.g., an island, lake, or region) from the country containing it; (2) retrieve its area from parametric knowledge while preserving decimal precision; (3) convert the value to km^2; and (4) output an integer or decimal value as `FINAL_ANSWER: ["Value"]`.

`hasCapacity`: Four-Step Reasoning with Capacity-Range Verification. The model follows four CoT steps: (1) determine the venue type and location; (2) distinguish a university or small local stadium from a larger national stadium with a similar name or in the same city; (3) check the estimate against a plausible capacity range (1,000–35,000 seats for small venues and 35,000–100,000 for large national stadiums); and (4) return the capacity as an integer in `FINAL_ANSWER: ["Value"]`.

personHasCityOfDeath: Five-Step Reasoning with Empty-Set Gate. The model performs five steps: (1) check the subject's biographical status and record "ALIVE TODAY" if appropriate; (2) resolve name ambiguity; (3) retrieve the city or town of death; (4) isolate the place name; and (5) return `FINAL_ANSWER: []` if the person is alive or the information is unavailable, or `FINAL_ANSWER: ["CityName"]` if the place of death is confirmed.

3.3 Post-processing

To ensure that extracted outputs adhere to the ground-truth format, we apply the following automated post-processing procedure after the generation phase:

1. **Direct Parsing and Robust Array Extraction**: The system directly parses the content of the `FINAL_ANSWER:` line or recovers a truncated JSON array using balanced-bracket scanning.

2. **Numeric Extraction and Normalization**: For quantitative relations (`hasArea` and `hasCapacity`), regular expressions extract integer or floating-point values while removing thousands separators and measurement units.

3. **Title and Noise Filtering**: For `awardWonBy`, the system removes year prefixes (e.g., *"1938: Albert Einstein"*), HTML tags, work-title suffixes, and honorifics such as *Dr.*, *Prof.*, *Sir*, *Dame*, *Lord*, *Saint*, *St.*, and *Mr.* to obtain normalized entity names.

4. **Parenthetical Filtering and Deduplication**: The system removes trailing parenthetical qualifiers (e.g., *"Guinea (West Africa)"* $\rightarrow$ *"Guinea"*) and performs case-insensitive deduplication while preserving the capitalization of the first occurrence.

4 Experiments

4.1 Experimental Setup

Data. The organizers provide training, validation, and test splits for six relations. The validation and test splits each contain 475 records: 68 for `countryLandBordersCountry` in validation (67 in test), 97 and 98 for `hasCapacity` respectively, 10 for `awardWonBy`, and 100 for each of the other three relations. Each record consists of a subject, a relation, and a set of objects, each accompanied by a list of aliases. The data cover all three cardinality cases: empty sets (e.g., *New Zealand* for land borders), single-valued relations, and multi-valued relations (e.g., Nobel Prize in Physics, 229 recipients in `train.jsonl`, or AAAI Fellow, 350 people in `val.jsonl`).

Models. We evaluate three instruction-tuned models—Gemma-2-9B-it (Gemma Team, 2024), Llama-3.1-8B-Instruct (Llama Team, Meta AI, 2024), and Mistral-Small-24B-Instruct-2501 (Mistral AI, 2025)—and compare against the organizer's Qwen3.5-9B baseline (Qwen Team, 2026). Gemma-2-9B-it and Llama-3.1-8B-Instruct serve as comparison systems, while Mistral-Small-24B-Instruct-2501 (hereafter Mistral-24B) is the primary model in our final system.

Hardware and Runtime. Experiments are conducted on Kaggle using a TPU v5e-8 with eight TPU v5e cores. We serve the models with vLLM TPU Server and use bfloat16 precision. With direct parsing and batched inference, one complete run over all six relations in the validation set (475 records) takes approximately *2–5 minutes* with `batch_size=32`.

Metrics. We report macro/micro precision, recall, and F1 using the organizer-provided `evaluate.py` script. For string-valued relations, the evaluator normalizes predictions and matches aliases through maximum bipartite matching. For quantitative relations, it applies a relative tolerance of 5%. Overall macro-F1 is the mean of the *per-record* F1 scores.

4.2 Validation Results

We first evaluate all models in a *zero-shot* setting, using direct queries without evidence generation or CoT reasoning. As shown in Figure 2, Llama-3.1-8B, Gemma-2-9B, and Mistral-24B achieve macro-F1 scores of 0.38, 0.42, and 0.43, respectively. These relatively low and similar scores suggest that all three models struggle when required to recall facts and satisfy the output format simultaneously. The limitation is particularly pronounced for multi-valued relations that require long object lists, such as `awardWonBy`, and for quantitative relations requiring precise values.

Effectiveness of the Two-Stage Pipeline. Figure 2 shows that the two-stage pipeline improves macro-F1 for all three models. Llama-3.1-8B improves from 0.38 to 0.48, Gemma-2-9B

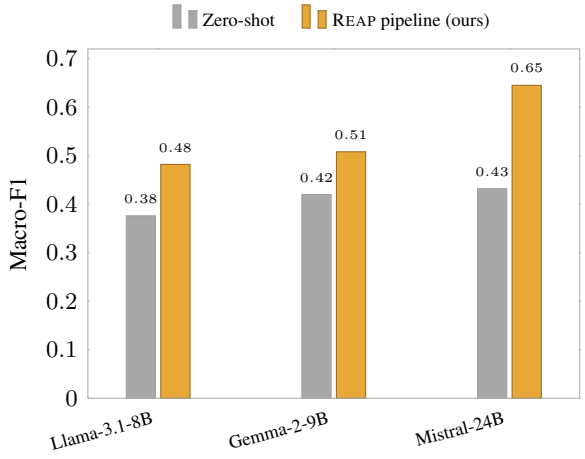


Figure 2: Overall macro-F1 of zero-shot prompting and the two-stage pipeline on the validation set.

Table 2: Macro-F1 of Mistral-24B on the test set.

Relation	Macro-F1
awardWonBy	0.37
companyTradesAtStockExchange	0.73
countryLandBordersCountry	0.95
hasArea	0.77
hasCapacity	0.23
personHasCityOfDeath	0.53
Overall	**0.62**

from 0.42 to 0.51, and Mistral-24B from 0.43 to 0.65. Mistral-24B yields the largest absolute gain (+0.22) and is therefore selected as the primary model in our final system submitted to the AKBC Shared Task 2026.

4.3 Test Results

Table 2 reports the relation-level and overall macro-F1 scores of the primary system on the official test set.

The relation-level results in Figure 3 show that the Mistral-24B pipeline consistently outperforms all other configurations across all six relations on the F1 metric. It achieves its highest scores on countryLandBordersCountry (0.95) and hasArea (0.77). Overall, the Mistral pipeline leads with a macro-F1 score of 0.62, significantly outperforming the Llama (0.48) and Gemma (0.48) pipelines, as well as more than doubling the organizer's baseline that is based on Qwen3.5-9B (0.30).

Table 3: Ablation results on the validation set.

Configuration	Macro-F1	Δ
Full Pipeline (Ours)	**0.65**	–
w/o Task Decomposition	0.59	−0.06
w/o Empty-Set Gate	0.57	−0.08
w/o CoT (Zero-Shot)	0.43	−0.22

5 Ablation Study

To quantify each component's contribution, we perform a subtractive ablation study on the validation set using Mistral-24B (Table 3).

Removing *Task Decomposition* drops macro-F1 from 0.65 to 0.59, as single-pass queries suffer severe recall loss on multi-valued relations: awardWonBy collapses from 0.25 to 0.00 and countryLandBordersCountry falls from 0.98 to 0.63. Removing the *Empty-Set Gate* lowers macro-F1 to 0.57 due to hallucinated answers for unanswerable entities, severely hurting precision on personHasCityOfDeath (0.50 → 0.36) and companyTradesAtStockExchange (0.76 → 0.57). Finally, removing *CoT Reasoning* entirely causes the largest drop to 0.43, heavily degrading hasArea (0.82 → 0.55) and countryLandBordersCountry (0.98 → 0.27), confirming that our gains stem primarily from structured parametric reasoning rather than output parsing alone.

6 Discussion

Relation-Aware Prompting versus Zero-Shot Prompting. In the zero-shot setting, all models exhibit limited and relatively similar performance. Requiring a model to recall facts and produce valid JSON in a single pass is especially difficult for relations with large object sets, such as awardWonBy, and for quantitative relations that demand precise values, such as hasArea and hasCapacity. Applying the complete *two-stage pipeline* of REAP with relation-specific prompts, CoT reasoning, a reasoning-based empty-set gate, and direct parsing improves all three models. These results indicate that structured reasoning is important for reliably eliciting knowledge encoded in model parameters.

Limits of Parametric Knowledge. Our results reveal a clear gap between the 9B model (Gemma-2-9B) and the 24B model (Mistral-24B). In the zero-shot setting, the models perform similarly, partly because both struggle with output format-

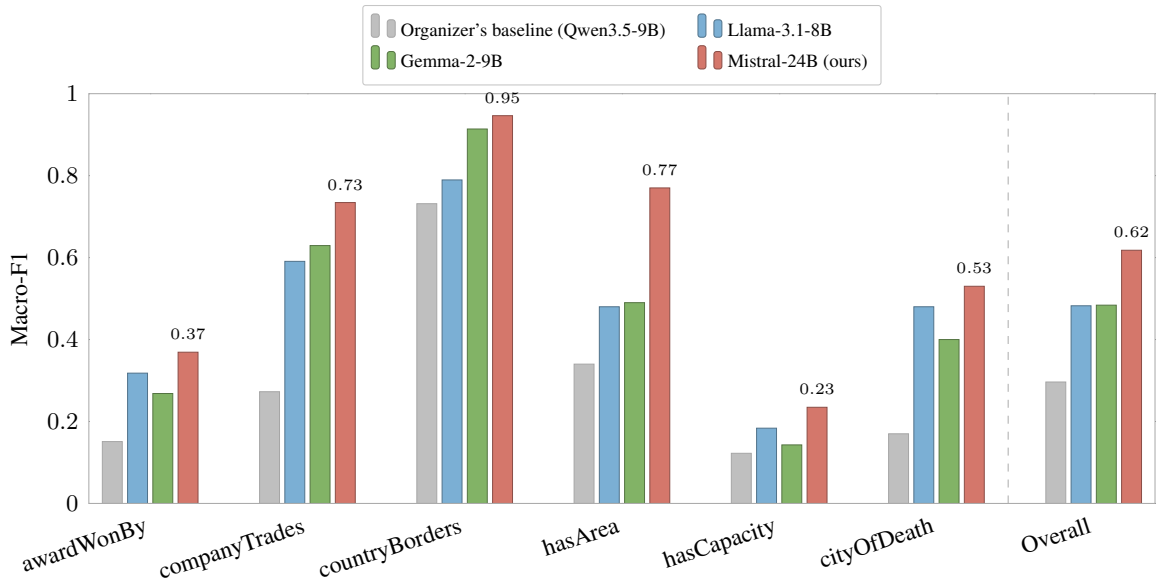


Figure 3: Relation-level macro-F1 of the organizer's Qwen3.5-9B baseline and three models using the two-stage REAP pipeline on the official test set of the AKBC Shared Task 2026.

ting. Under the full structured-reasoning pipeline, however, Mistral-24B reaches macro-F1 0.65 on the validation set, compared with 0.51 for Gemma-2-9B. This result suggests that reasoning prompts can reduce formatting noise, while successful factual retrieval still depends on model capacity. Mistral-24B performs better on long-tail entities such as small islands, club-level stadiums, and less prominent awards. Thus, in a closed-book setting, a larger model may benefit more from such a pipeline.

Run-to-run variance. Two minor sources of variation affect our runs. First, `awardWonBy` uses sampling ($T = 0.7$, six decade-scoped passes) to diversify recall over long recipient lists, so its score fluctuates slightly given only 10 validation awards. Second, even under greedy decoding ($T = 0.0$) the TPU v5e-8 is not bit-exact, because the accumulation order of bfloat16 values differs across its eight cores. Both effects are small—overall macro-F1 varies by about ± 0.005—and do not change our conclusions.

7 Conclusion

We presented the REAP system for the AKBC Shared Task 2026 under a strict closed-book setting with at most 32B parameters, no external knowledge, and no model fine-tuning. The system implements a unified two-stage pipeline with Mistral-Small-24B-Instruct-2501, combining relation-aware prompt engineering, a reasoning-based empty-set gate, and direct parsing. It achieves a macro-F1 of 0.65 on the validation set and a macro-F1 of 0.62 on the official test set. The results demonstrate that structured reasoning can effectively elicit factual knowledge encoded in model parameters for knowledge base construction.

Limitations

Despite its promising performance, our system has several limitations: (i) its parametric knowledge remains incomplete for very rare, long-tail entities; (ii) predictions for quantitative relations sometimes fall outside the 5% relative-tolerance threshold because of retrieval or rounding errors; and (iii) stochastic sampling for `awardWonBy`, together with non-deterministic distributed computation on TPU v5e-8, causes slight variation across runs. Among the six relations, `hasCapacity` is the most challenging (test macro-F1 0.23): venues are frequently confused with larger namesakes in the same city, and even small numeric errors exceed the 5% tolerance.

References

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-Wise Self-consistency for LLM Probing. In *LM-KBC @ ISWC 2025*.

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. https://arxiv.org/abs/2204.06031.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *NeurIPS*.

Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. https://arxiv.org/abs/2408.00118.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *TACL*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. https://arxiv.org/abs/2207.05221.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2024. Preface: LM-KBC challenge 2024. In *ISWC*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pre-trained language models. In *ISWC*.

Jan-Christoph Kalo, Sneha Singhania, Simon Razniewski, and Jeff Z. Pan. 2023. LM-KBC 2023: 2nd challenge on knowledge base construction from pre-trained language models. In *ISWC*.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *ACL*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *ICML*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.

Llama Team, Meta AI. 2024. The llama 3 herd of models. https://arxiv.org/abs/2407.21783.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*.

Mistral AI. 2025. Mistral-small-24b-instruct-2501. https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *EMNLP*.

Sneha Singhania, Tuan-Phong Nguyen, and Simon Razniewski. 2022. LM-KBC: Knowledge base construction from pre-trained language models. In *ISWC*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Bohui Zhang, Ioannis Reklos, Nitisha Jain, Albert Meroño-Peñuela, and Elena Simperl. 2023. Using large language models for knowledge engineering (LLMKE): A case study on Wikidata. In *LM-KBC @ ISWC 2023*.

SetProbe: Description-First Relation-Wise Self-Consistency for Closed-Book Knowledge Base Construction☆

Jannic Alexander Cutura
ECB† & DSTI‡
jannic.cutura@dsti.institute

Abstract

The AKBC 2026 shared task asks a language model to recover complete object sets closed-book. SetProbe anchors each subject, samples several candidate sets, and aggregates them per relation, reaching 0.513 macro F1 on validation. Sample count, aggregation rule, abstention thresholds, and the choice between two eligible models each move the overall score by less than the evaluation set's own 0.045 confidence interval. Only the model moves individual relations, and it moves them in opposite directions that cancel. The binding constraint is elementary: closed-book prediction cannot recover an object that appears in no sample.

1 Introduction

How can knowledge stored in a language model be turned into a complete database entry? A model may readily answer that the Dominican Republic borders Haiti, yet the same question becomes considerably harder when the expected answer contains nine neighboring countries, 80 award recipients, or no object at all. The AKBC 2026 shared task isolates this problem: given a subject s and a relation r, a system must predict the complete set of object strings $\{o_1, \ldots, o_k\}$ (Kalo et al., 2026).

Consider a country with three gold neighbors. Returning two correct ones yields perfect precision and $2/3$ recall, while adding three plausible but wrong neighbors leaves recall unchanged and cuts precision to $2/5$; inventing a death city for a living person turns an otherwise perfect row into a complete miss. The design question is therefore not whether the model knows a fact, but when a further candidate should be admitted.

We answer it with description-first relation-wise self-consistency, which identifies the exact subject, generates candidate sets through prompts exposing different recall paths, and admits empty answers into aggregation rather than gating generation beforehand. It reaches 0.513 macro F1 on validation.

The instructive result however is what did not move that number. Sample count is flat from three to twenty passes, no aggregation rule beats a plain median, calibrating abstention thresholds on training traces reproduced the defaults exactly, and swapping one eligible model for another shifts the overall score by less than its confidence interval. Only the model moves individual relations, for an elementary reason: a correct area value appears in the samples for 0.69 of rows under one model and 0.43 under another, and no rule selects a value never generated.

2 Task

The dataset contains six relations and 477 training, 478 validation, and 475 test rows; we reserve validation for evaluation, and fit prompts and thresholds on training data only. Cardinality varies sharply across relations. Area and capacity carry exactly one numeric object, accepted within 5% relative tolerance. Death cities and stock exchanges are gold-empty on 39 and 36 of their 100 validation rows, so returning nothing is often the correct answer. Borders reach 14 neighbors and one award row has 610 recipients. String predictions are matched after normalization and alias expansion, and the official score is macro-averaged precision, recall, and F1 over subject–relation pairs.

The setting is closed-book: inference may use open-weight models totalling at most 32 billion published parameters, but may not query the web, a retrieval index, an external corpus, or a knowledge

☆ The views expressed in this paper are solely those of the author and do not necessarily reflect the views of the European Central Bank or the European System of Central Banks. All code and data for full replication are available at https://github.com/JannicCutura/emnlp-kbc-2026.

†European Central Bank, Sonnemannstraße 20, 60314 Frankfurt am Main, Germany.

‡DSTI School of Engineering, 59 Rue de l'Abbé Groult, 75015 Paris, France.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 441–446
October 28, 2026 ©2026 Association for Computational Linguistics

base, and fine-tuning is prohibited. In-context examples, multi-step inference, and non-neural post-processing are allowed (Kalo et al., 2026). Our system therefore uses the training split only for few-shot demonstrations and no external factual source.

3 System

3.1 Common inference procedure

Figure 1 shows one row of inference. All calls use one open-weight model through an OpenAI-compatible chat endpoint. The model first produces a short identity anchor describing entity type, scope, and likely namesake confusions, then receives that anchor as explicitly uncertain context and generates n_r concise rationales and candidate sets. Each pass uses a different seed, recall focus, and rotating leave-one-subject-out sample of training demonstrations. Passes are independent and issued concurrently, with trace records following request order rather than completion order, so a concurrent run reproduces a serial one exactly.

For relations that permit empty answers, the demonstration mix mirrors the relation's own training empty rate—two of five for death cities, one of five for borders—rather than being fixed in advance. An earlier version alternated empty and non-empty examples, showing a 60% empty rate for every relation and biasing the model toward abstention on precisely the relations where abstention is scored.

Candidate strings are parsed from the first valid JSON structure and normalized with the official evaluator's rules. We additionally build relation-specific alias maps from the nested aliases in the training split, rewriting an alias only when all its training occurrences identify one canonical entity, before support is counted so that equivalent surfaces reinforce rather than split a vote. A response the parser cannot read is dropped from its row's aggregate rather than counted as empty, which would convert a parser failure into a vote for "no objects"; this never occurred in 5,392 validation calls.

3.2 Relation-specific policies

Table 1 gives the pass count, aggregation rule, and post-processing for each relation; the prompts themselves are in the repository. Three of these choices are not self-evident.

Seven numeric samples, and a median. Both follow a training-only sweep rather than convention. Over the 100 training area rows accuracy is flat between 0.53 and 0.58 for every count from 3 to 20 and for every estimator we tried, including mode-seeking, trimmed, and order-of-magnitude-filtered variants. Twenty samples cost three times as much as seven and bought nothing measurable; the median survives by default rather than by merit.

Relation	n_r	Aggregation	After
Area	7	median	—
Capacity	7	median	—
Death city	20	plurality w/ empty	—
Exchange	10	support + empty	—
Borders	3	union	compl. + verif.
Awards	20	support union	compl.

Table 1: Per-relation policy. Post-processing runs only on a non-empty aggregate. Support thresholds are 0.30 for exchanges and 0.05 for awards, and the empty-vote threshold is 0.50 for both abstention relations.

Post-processing runs only on a non-empty set. This gate was the largest correction we made and we did not anticipate it. Asking "which neighbors are missing?" is a leading question, and a model asked for omissions supplies them whether or not any exist: in 12 of the 18 gold-empty validation rows every sample correctly returned no neighbor, after which the completion pass invented Italy for Malta, Cuba and the United States for the Bahamas, and three maritime neighbors for Australia. The verifier accepted all of them. Across the split completion introduced 36 neighbors while verification removed 22. An omission pass presupposes a set to have omissions from; gating it raised border F1 from 0.796 to 0.859.

Awards are partitioned by era. A single answer may contain 610 recipients, so one undifferentiated prompt invites both truncation and omission. Twenty passes cycle through five time windows and a sixth focus organized by profession and recipient type. We apply no binary verifier: an earlier one cut correct recipients from 96 to 56.

4 Experimental Setup

Models. We evaluate Gemma 4 31B-IT (30.7B published total parameters) and Gemma 3 27B-IT (27.4B) on complete splits, and screen Qwen3.5 27B on area alone. Every relation in a run uses one model, so the inference-time total never exceeds 32B; quantization does not reduce it and mixture-of-experts models count by published total.

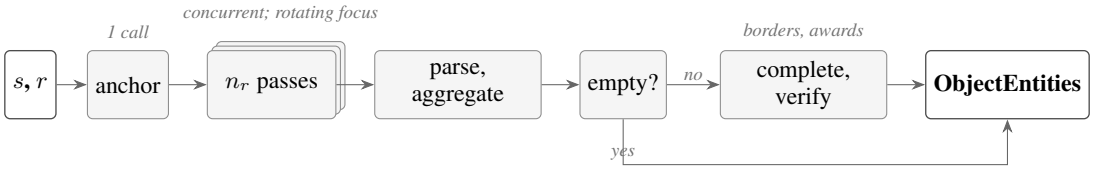


Figure 1: One row of inference. Completion and verification run only on a non-empty aggregate: asking an omission pass what is missing from an empty set invents members.

Inference configuration. Inference runs through a hosted OpenAI-compatible gateway. Such gateways route one model name across backends at precisions from fp4 to bf16, so we pin the provider and bf16 precision and record both on every call; otherwise a comparison confounds model identity with numeric precision. Runs use temperature 0.6, top-p 0.95, a 4,096-token cap, and five in-context examples except for awards. Hidden reasoning is disabled explicitly: it is not bounded by the generation cap and is billed at the completion rate, and one 100-row screen on a reasoning-tuned model emitted 3.87M such tokens. A complete validation run costs 5,392 calls and about 80 minutes.

Selection and controlled comparisons. All thresholds are selected from complete leave-one-out training traces with the official evaluator, and the locked configuration is then evaluated on the validation rows. We do not tune thresholds, sample counts, aggregation rules, or entity-specific rules against validation answers, and because the validation gold is distributed with the task we additionally report training-split scores as an independent estimate. Four factors are varied in controlled comparisons: sample count, aggregation rule, abstention threshold, and model. The identity anchor and the demonstration mix are held fixed in every run and are not ablated, so their contribution to the score is not measured. The historical baseline is an earlier GPT-OSS-20B prototype scoring 0.460, whose hard empty-answer gates and award verifier the present system removes. Comparisons use a paired bootstrap over rows with 20,000 resamples.

Test split. The test gold is not distributed, so we report no test score. Test predictions were generated once, on 8 August 2026, under the configuration locked above and before any test gold was available; the 475-row submission archive is frozen with its SHA-256 hash (`2ef01673...`) and released with the code, so the organizers' score can be tied to this exact configuration.

	Validation			Train
Relation	G4 31B	G3 27B	Δ	G4 31B
Land borders	**.989**	.859	$+.130^*$	.953
Stock exchange	**.713**	.648	$+.064$	.663
Death city	**.460**	.340	$+.120$	.500
Capacity	.220	**.230**	$-.010$	.110
Awards	.148	**.205**	$-.057$	.395
Area	.370	**.520**	$-.150^*$	.360
All relations	**.513**	.490	$+.022$	.484

Table 2: Per-relation F1. * marks validation differences whose 95% paired-bootstrap interval excludes zero; the overall difference does not ($[-0.022, +0.067]$, $p = 0.33$). The last column is the locked Gemma 4 system on the training split, an independent estimate on rows whose gold was used only for demonstrations and threshold selection.

5 Results

On all 478 validation rows the system reaches macro precision 0.542, recall 0.578, and F1 0.513, against 0.460 for an earlier gated pipeline of ours on a different model. The 95% bootstrap interval is $[0.468, 0.557]$, and an independent run on the training split scores 0.484.

Table 2 shows the two models are complementary rather than ordered. Gemma 4 is better on every relation rewarding adherence to a precise definition, reaching 0.989 on borders where one row separates it from a perfect score, while Gemma 3 is better on the two relations rewarding raw numeric recall. Only borders and area survive a paired bootstrap (death city's interval, $[0.000, +0.240]$, touches zero) and they nearly cancel: the overall $+0.022$ has interval $[-0.022, +0.067]$ and $p = 0.33$, so on the whole split the two models are statistically indistinguishable. The best system we can describe is one we may not submit, since pairing Gemma 3's area with Gemma 4's other relations would score 0.544 but sum to 58.1B parameters.

Tuning does not move the score; the model redistributes it. None of our four controlled comparisons moves the overall score beyond its interval.

Sample count is flat from three to twenty passes; no aggregation rule we tried improves on a plain median; sweeping the abstention thresholds over complete training traces simply rediscovered the values we had guessed, leaving death and exchange where they already stood; and exchanging one eligible model for another changes the total by 0.022. The model is the only factor that changes any single relation, and it changes them in opposite directions rather than lifting the whole.

The reason is visible in the candidate pools. For `hasArea` a correct value appears in at least one sample for 0.69 of rows under Gemma 3 and 0.43 under Gemma 4, while the aggregates return 0.52 and 0.37; for `hasCapacity` the ceiling is 0.19. Where the correct value is absent from every sample no selection rule can recover it, and where it is present but in the minority, agreement-based aggregation discards it because an incorrect value dominates the distribution.

6 Error Analysis

Abstention is too conservative, not too aggressive. On stock exchanges the system predicts empty on all 36 gold-empty rows with no spurious answer at all, yet returns nothing on 26 rows that do have a listing; death cities are correct on 33 of 39 gold-empty rows against 32 rows where a recoverable city was suppressed. This is a negative result about our own design: the prototype's hard pre-generation gate suppressed 28 of 61 non-empty death rows, and the generation-first voting that replaced it suppresses 32. The mechanism changed from a classifier to a vote but the operating point did not, because the samples agree on silence—when the model does not know, twenty passes independently return nothing, and no rule over unanimous abstentions manufactures a city.

Parsing is not the limiting factor. Because our central claim is that correct objects are absent rather than mishandled, we audited it: across 5,392 calls the parser never dropped a gold alias present in an answer array. Of the incorrect rows, 58 of 63 for area, 71 of 78 for capacity and 38 of the 48 death rows with a gold city contain no correct value anywhere in any response. Aggregation discards a correct value that at least one sample did return in 14 further rows (2 area, 7 capacity, 5 death), worth about 0.03 overall; in the remaining 8 the value appears only in a rationale, never in an answer array.

Stock exchanges are the exception. In 9 of the 27 rows scoring zero the model names the correct exchange in its rationale and then returns an empty set because it believes the listing has ended—Leclanché is "previously listed on the SIX Swiss Exchange, but delisted following its acquisition". Given that judgement it correctly obeys our instruction to return nothing for a company that is not independently public, so reading the answer out of the rationale would mean admitting its own negations. Here the ceiling is set by temporal belief rather than by missing knowledge.

Remaining errors. Among incorrect area rows the median relative error is 41% under Gemma 4 and 63% under Gemma 3, while over correct rows it is below 0.5%, so failures are misidentification rather than noise. The identity anchor was introduced for exactly this, and the traces show it can transmit the error as readily as prevent it: for *Ayon Island*, 2,000 km^2 in the East Siberian Sea, the anchor placed the subject in the Philippines and all seven samples followed, with a median of 1.5 km^2; for *Mainland*, a 540 km^2 Scottish island, the anchor declared the name ambiguous and the system returned nothing, which the evaluator scores no better. Whether the anchor helps on net we cannot say, because no run omits it. Under Gemma 4, border recall is 0.984 and precision 0.996 once the omission pass is gated. Awards average 68.0 predicted entities against a gold mean of 145.8, up from 15.2, but ten rows cannot be measured to better than ± 0.11.

7 Related Work

LAMA established that pretrained models hold retrievable factual associations (Petroni et al., 2019) and closed-book question answering that they can be queried without document access (Roberts et al., 2020). Both ask whether a known object is recovered, not how many exist, which is what makes this task hard: a probe scored on its top answer never has to decide that a country has nine neighbors or that a living person has no city of death. Phrasing alone moves probing results substantially (Elazar et al., 2021), so we vary phrasing across passes rather than search for one best prompt. Retrieval would supply exactly the long-tail facts parametric memory covers worst (Lewis et al., 2020; Mallen et al., 2023); forbidding it leaves prompt diversity and repeated sampling as the only sources of variation.

The task has run annually with shifting resource rules (Kalo et al., 2024, 2025). The 2025 edition is closest to ours: it fixed a single small model, so methods rather than models distinguished the entries. The winner combined chain-of-thought prompting with relation-wise self-consistency and train-tuned per-relation aggregation (Albert-Roulhac and Zouaq, 2025); the runner-up generated an entity description before extraction and decomposed high-cardinality relations (He and Razniewski, 2025). We adopt both. Our contribution is not their invention but a measurement of how much the sampling and aggregation choices around them are worth once the model may vary, which a fixed-model setting cannot reveal.

Self-consistency aggregates repeated samples by majority vote (Wang et al., 2023), but assumes one final answer; set prediction instead votes over normalized candidates, where a value generated once in twenty passes may still be correct. Chain-of-verification reduces hallucination through separate checking questions (Dhuliawala et al., 2024), though a closed-book verifier consults the same parameters as the generator, so we verify only borders. Finally, a third of death and exchange rows are gold-empty, making abstention a first-class prediction as in unanswerable-question benchmarks (Rajpurkar et al., 2018) and calibrated selective prediction (Kamath et al., 2020).

8 Conclusion

We presented a description-first self-consistency pipeline for closed-book knowledge base construction, sharing one orchestration pattern across relations while assigning aggregation by cardinality. It scores 0.513 on validation.

The more useful result is where the remaining error lives. Varying sample count, aggregation rule, abstention threshold, and model, none changes the overall score beyond its confidence interval; only the model changes individual relations, and its gains are relation-specific and mutually exclusive under a parameter budget. An incorrect value that dominates the sample distribution defeats agreement-based selection precisely when agreement is what one would like to trust. Effort is therefore better spent on elicitation diversity than on aggregation, and reporting a candidate-pool ceiling alongside the score would make systems easier to compare.

Limitations

The system cannot recover facts absent from the model's pretraining data, which matters most for deaths, listings, and awards close to the July 2026 reference date. The identity anchor is not ablated: it is present in every run, so its net contribution is unmeasured, and Section 6 shows it can propagate a misidentification as well as prevent one. The two models are compared on one validation split and are not statistically distinguishable overall; the per-relation differences we report are the finding, not a ranking. Reusing one model for border generation and verification may reinforce rather than correct a misconception, and train-calibrated thresholds may shift under distribution change. No test score is reported because the test gold is not distributed; the frozen test predictions described in Section 4 await the organizers' evaluation, and every prompt, response, and configuration is recorded.

References

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-wise self-consistency for LLM probing. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction*, volume 4041 of *CEUR Workshop Proceedings*, Nara, Japan.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Jingbo He and Simon Razniewski. 2025. Combining self-retrieval-augmented generation with divide-and-conquer for language model-based knowledge base construction. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction*, volume 4041 of *CEUR Workshop Proceedings*, Nara, Japan.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2024. Preface: LM-KBC challenge 2024. In *Joint Proceedings of the KBC-LM Workshop and the LM-KBC Challenge*

2024, volume 3853 of *CEUR Workshop Proceedings*, Baltimore, USA.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pretrained language models. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction*, volume 4041 of *CEUR Workshop Proceedings*, Nara, Japan.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2463–2473. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Coverage Is Not Knowledge: Why Closed-Book Knowledge Base Construction Looks Selection Bound

Maksim Silchenko
National University of Singapore
mthylinao@gmail.com

Abstract

This paper describes the first place system of the AKBC 2026 shared task on predicting complete knowledge base entries from language models, built on a single 31.3B parameter base model in completion mode with no training and no retrieval. It scores 0.7060 on the official test set; the runner up scores 0.6908, and a composite taking the best per relation score of every other ranked team would reach 0.7011.

The largest single improvement I found is not an algorithm. Asking the model to recall facts about an entity before committing to an answer, rather than asking for the answer directly, moved `hasArea` from 0.8100 to 0.8700. Three earlier configurations, across two frames and two aggregation methods, had all returned exactly 0.8100, which I had read as evidence that the answers were absent from the candidate pool. That reading was incorrect.

The main analytical result is a correction to how this field measures headroom, and it corrects my own earlier conclusion as much as anyone else's. Oracle coverage analyses report that `hasCapacity` is selection bound: the recitation frame puts an acceptable answer in the pool for 84.5 percent of validation subjects while the aggregator realises 36.1 percent. That inference does not survive a null control. A pool of one hundred guesses spanning one decade contains about twenty one mutually exclusive 5 percent tolerance windows, so it contains something acceptably close to almost any plausible target whether or not the model knows the answer. Permuting gold values across subjects measures that coincidence rate directly: 0.355 for `hasCapacity` against 0.031 for `hasArea`. Chance corrected, the findable headroom beyond the rank one answer falls from 0.443 to 0.203. Three instruments that are not functions of vote frequency then agree that the relation is knowledge bound.

1 Introduction

The task (Kalo et al., 2026) gives a subject and a relation and asks for the complete set of object entities as of a fixed world state, using only the parametric memory of an open weight model of at most 32B parameters, continuing the LM-KBC challenge series (Singhania et al., 2022; Kalo et al., 2025). No retrieval, no external knowledge base, no training of any kind.

Two features of the scoring decide what a good system looks like, and both work against the intuition that this is a recall problem.

First, empty answers are first class. A row whose gold set is empty scores 1.0 if and only if the system also predicts nothing, and 0 otherwise. On the test split, 102 of 475 rows have empty gold. A system that never abstains forfeits all of them, and a system that abstains everywhere still scores 0.2147. The abstention decision is therefore worth more than additional recall on the rows that do have answers.

Second, the two numeric relations are graded as a single value within five percent of the gold. There is no partial credit and no set overlap available. Together the numeric relations are 198 of 475 rows, so a system's numeric behaviour accounts for close to half its score.

I began with the reading that the interesting difficulty is not whether the model knows the fact but whether the system can identify which of the things the model says is the fact. Every coverage statistic I and the published systems computed supported it. Section 6.4 retracts it for the relation where it mattered most: that statistic does not control for the width of the guess distribution, and once it does, most of the apparent selection headroom on `hasCapacity` is no longer present. I report the retraction in full, including the effort spent on the wrong side of it, because the same uncontrolled statistic is what the rest of this literature

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 447–455
October 28, 2026 ©2026 Association for Computational Linguistics

uses to direct its effort.

Code, prompts, seeds, threshold grids, and the parameter audit are available at https://github.com/thylinao1/lm-kbc-2026.

2 Task, data, and metric

Six relations, 477 train rows, 475 validation rows, 475 test rows. Objects are alias lists, and matching any alias counts as correct.

The official scorer computes, per (subject, relation) row, precision as $tp/|preds|$ with precision defined as 1.0 when the system predicts nothing, and recall as $tp/|golds|$ with recall defined as 1.0 when the gold set is empty. The reported figure is the unweighted mean of per row F1 across all rows. Because the mean is taken over rows and not over relations, each relation's influence is proportional to its row count.

The test split holds `personHasCityOfDeath`, `hasArea` and `companyTradesAtStockExchange` at 100 rows each, `hasCapacity` at 98, `countryLandBordersCountry` at 67 and `awardWonBy` at 10. Empty gold rates are 48%, 0%, 44%, 0%, 14.9% and 0% respectively.

Those rates are measured rather than assumed. I submitted a single all empty predictions file, which returns per relation macro F1 equal to that relation's empty gold fraction exactly, and read the six fractions off the returned scores. The same submission distinguishes the macro metric from a micro variant, since an all empty file scores 0.2147 under the former and 0.0 under the latter.

Two consequences are used throughout. `awardWonBy` is 2.1% of the score across ten rows, so its relation level F1 carries a confidence interval of roughly ± 0.23 and work on it cannot be measured reliably. And the validation split has materially different empty rates from test, by $+9$ points on `personHasCityOfDeath` and -11.6 points on `countryLandBordersCountry`, so abstention thresholds tuned on validation priors are mis-centred on test in both directions.

3 Related systems

This system builds on published work, and the parts that are replication are identified as such.

The strongest public system available during development (ruggsea, 2026) uses a 31B base model in pure completion mode with per relation elicitation recipes, sampling tens of draws per subject and deciding by vote share. Two of its findings shaped the design here: base completion performs better than instruction tuned on most relations, and a standalone abstention gate was measurably inert while removing vote share carried a large cost. I reimplement the recipes from the published description rather than copying code, because that repository carries no licence file. A second public system (dukesun99, 2026) reaches a lower overall score using one token logit probes with placebo correction, multi channel numeric elicitation, and cross model agreement. It is CC BY 4.0 and I credit it for the multi channel numeric idea and the overshoot calibration.

From the 2025 edition (Kalo et al., 2025; Razniewski et al., 2024) I take entity level self consistency (Wang et al., 2023) with per relation thresholds, the finding that chain of thought reduces factual recall on several relations, and a decomposition and union strategy for the high cardinality award relation, together with the negative result that decomposition reduces performance on medium cardinality relations.

4 System

4.1 Substrate

One model, `gemma-4-31B` (Google, 2026), in base completion mode, served with vLLM. I measured the checkpoint at 31,273,088,876 BF16 parameters from the safetensors index, which is under the 32B cap. For auditors, the model's hosting page displays 32.68B; the difference is the tied input embedding counted a second time as an output projection, and the served model holds one copy.

The budget is a hard constraint on architecture rather than a preference. At 31.273B the remaining headroom is 0.727B, which excludes any second neural component: no agreement sidecar, no reranker, no draft model for speculative decoding. Every component of the system after generation is non neural post processing. I also require that any submitted predictions file is produced end to end by this single substrate, since mixing per relation outputs from two models would sum their parameter counts and breach the cap without this being visible in the output file. Appendix A describes the four checks that enforce closed book compliance and the content hash that keys every generated pool.

4.2 Elicitation and aggregation

Each relation is served by one or more channels. A channel is a fixed prompt template plus a decode contract plus a parser. Demonstrations are drawn from train only and are fixed per channel, never resampled per draw: per draw resampling defeats prefix caching and costs roughly three orders of magnitude more prefill for a gain that sits inside the measurement noise here.

For set valued relations I take the vote share of each normalised candidate across draws and keep those at or above a threshold. Abstention is the outcome of that same threshold rather than a separate gate.

For numeric relations I score each distinct drawn value by how many draws fall within the grader's own tolerance of it and take the argmax, emitting a single bare integer. Averaging across clusters is not appropriate here: a draw set of 4900, 5000, 5050, 5100 and 50000 has a mean near 14000, which fails under any tolerance, and a largest cluster median of 5025, which is correct.

5 Experimental setup

5.1 Serving

One H100 94GB card, tensor parallel size 1, `gpu_memory_utilization` 0.90, `enable_prefix_caching` on. The 62.5 GB of bf16 weights leave roughly 30 GB for the KV cache. Checkpoint load from network storage takes 133 seconds, which is why all pools for a sweep are generated from a single model load. Sampling is temperature 0.7, top-p 0.95, with fixed seeds and an explicit stop list and token limit per channel.

5.2 Threshold selection, and two effects that inflate it

Thresholds are chosen on train and reported on validation. Two effects had to be corrected before the numbers here were reliable, and both inflated results in the direction of appearing better.

Demonstration leakage. Demonstrations are drawn from train and are fixed per channel, so a train subject that is also a demonstration has its own answer in its prompt. For borders that was 32 of 67 subjects, and it inflated the train figure from 0.9854 to 0.9924. I exclude demonstration subjects from the train curve. The cost is statistical power: 35 scorable subjects remain, so train side threshold choice on small relations is weak, and I report this rather than relying on it.

relation	rows	this	runner up	best other
`countryLandBordersC.`	67	0.9786	0.9873	0.9873
`hasArea`	100	**0.8700**	0.8400	0.8600
`companyTradesAtSE`	100	0.8530	0.8461	0.8717
`hasCapacity`	98	**0.3367**	0.3146	0.3163
`personHasCityOfDeath`	100	**0.6100**	0.5900	0.5900
`awardWonBy`	10	0.3484	0.3556	0.3691
total	475	**0.7060**	0.6908	0.7011

Table 1: Test macro F1 by relation on the official final leaderboard. Best other is the highest score of any other ranked team on that relation, so its total is a per relation composite rather than one system. Bold marks the best official figure.

Plateau edges. After that correction the train curve is frequently flat across a run of thresholds, so a plain argmax returns an arbitrary tied edge. I take the centre of the widest tied run instead. On borders that moves the choice from 0.15 to 0.30 and validation from 0.9846 to 0.9896.

6 Results

All leaderboard figures below are measured on the test set by the official grader rather than estimated on validation; the coverage diagnostics, which require retained draw pools, are computed on validation and labelled with their split and pool depth. On the official final leaderboard the system placed first of twelve ranked teams at 0.7060, with the runner up at 0.6908.

Three relations have some other ranked team ahead: borders by 0.0087, the stock exchange relation by 0.0187, and `awardWonBy` by 0.0207. The award relation has ten rows and a relation level confidence interval of roughly ± 0.23, so it is not measurable at this size and I did not spend effort on it.

6.1 Elicitation register rather than aggregation

The single change that produced the largest improvement in this work is not an algorithm. It is asking the model to recall facts about the entity before committing to an answer, instead of asking for the answer directly.

Concretely, for `hasArea`, replacing `{entity} has an area of ___` with a short recitation frame that first states what kind of thing the en-

tity is moved the relation from 0.8100 to 0.8700 on test, above the previous best posted figure of 0.8500. The same register is the best frame here for `hasCapacity`, where it reaches a pool coverage of 0.804 against the 0.77 reported by the strongest published system.

This is worth reporting because of how difficult it was to detect. Three configurations of `hasArea`, spanning two prompt frames and two aggregation methods, returned exactly 0.8100 on test. I had concluded from that stability, and from the three frames agreeing on 90 of 100 subjects, that the incorrect rows were incorrect because the pool did not contain the answer. That conclusion was wrong. Pool composition depends on the register, and better selection from a poorly elicited pool does not recover what the register never surfaced. Validation did not detect this: it rated the two frames a tie, 0.8500 each, with a paired bootstrap interval of $[-0.050, +0.050]$.

6.2 Validation alone could not have produced this system

Five submissions were scored, and the per-relation results contradict validation three separate times in a single submission. The `hasArea` recitation register read as a tie at 0.8500 either way and was worth $+0.0600$; `personHasCityOfDeath` threshold 0.35 to 0.45 read flat and was worth $+0.0300$; `stockExchange` ratio emission read $+0.0077$ and was worth $+0.0166$; the borders threshold 0.45 to 0.15 read worse at -0.0126 and was worth $+0.0123$; a `hasCapacity` anti-round-attractor rule read $+0.0103$ and cost -0.0102. Validation did not detect the two largest improvements, understated a third by half, and inverted the sign of the remaining two.

The reason is arithmetic rather than chance: each relation has 67 to 100 rows, so its validation standard error is 0.03 to 0.05, while the differences that decide the ranking are 0.01 to 0.03. A validation split of this size cannot resolve them, and a system tuned only on it is tuned on noise.

What can resolve them is the evaluation itself. Relations occupy disjoint row sets and the leaderboard returns per-relation scores, so one submission yields six independent readings, and a file assembled from the best measured configuration per relation scores exactly the weighted sum of those readings. I verified this: the assembled file was predicted to score 0.6916 and returned 0.6916. I report this as a methodological finding rather than a technique. Calibrating scalar decision parameters on returned aggregate per-relation scores is disclosed here as part of the method. I did not, and would not, use per-row feedback or any procedure that reconstructs which specific rows are correct.

6.3 A negative result with high confidence

Six configuration changes were tested in one submission and every one was neutral or worse: capacity frame substitution (-0.0408), award threshold (-0.0247), `stockExchange` threshold and ratio (-0.0100), `cityOfDeath` threshold pushed further (-0.0100), area selector and borders threshold (both 0.0000). Every scalar axis under my control is therefore at a measured local optimum, and `cityOfDeath` is bracketed on both sides (0.5800 at 0.35, 0.6100 at 0.45, 0.6000 at 0.55).

Summing the best figure of any other ranked team for each relation separately gives 0.7011, which is below this system's 0.7060: even a per relation composite of the entire rest of the official field does not reach the submitted single system. During development the same envelope computed over posted figures sat within noise of the then leading system, so a system built by assembling published recipes could not have exceeded that leader by more than noise. Exceeding such an envelope requires improving on a per relation best somewhere, which is why the work concentrated on one relation.

6.4 `hasCapacity` is not selection bound, although the coverage metric indicates that it is

For each subject I separate two questions. Coverage asks whether any draw in the pool contains an acceptable answer, which bounds what any selector reading that pool could achieve. Realized asks whether the aggregator actually emitted one.

Coverage against realized, by frame, on validation pools of thirty draws per subject: recitation 0.804 / 0.361, current-capacity 0.763 / 0.320, pipetable listing 0.722 / 0.237, location-disambiguated 0.701 / 0.299, infobox completion 0.680 / 0.247. At the full pool depth of one hundred draws the recitation frame's coverage rises to 0.845, the figure used in the permutation analysis below. Read at face value, every frame is selection bound by a wide margin, with 0.443 of the relation apparently sitting in the gap on the best of them.

That reading is incorrect, and the rest of this section is the correction. I report it in full because I

relation	cov.	chance	corr.	pool width
`hasCapacity`	0.845	**0.355**	0.760	0.91 dec., 21 win.
`hasArea`	0.940	**0.031**	0.938	0.01 dec., 0 win.

Table 2: Permutation control over 400 shuffles.

accepted it for most of the work and spent most of the available effort on it.

6.5 Coverage is not knowledge

A coverage number asks whether any draw lands within the grader's 5 percent tolerance of the gold. That question has a confound. A pool of one hundred guesses that spans one decade contains roughly twenty one mutually exclusive 5 percent windows, so it will contain something acceptably close to almost any plausible target whether or not the model knows the answer. Coverage therefore measures the width of the guess distribution as well as its correctness, and the two are not separated anywhere in this literature that I am aware of.

The control is a permutation test. Give every subject a different subject's gold value, drawn from the same relation so the null preserves the marginal distribution of plausible answers, and recompute coverage. Whatever survives is coincidence. Four hundred shuffles, validation split:

Both pools carry real knowledge, at z of 12.4 and 53.1 against the null. What differs is how much of the headline number reflects that knowledge. Area draws are close to unanimous, so area coverage is almost entirely knowledge, and rank one selection already realises 0.850 of the available 0.940. Capacity draws are dispersed, and 35.5 of the 84.5 coverage points reflect the pool being wide enough to contain many plausible values.

Decomposing by where the gold sits in the frequency ranking indicates where a selector could find something. Excess is the real rate minus the coincidence rate at that rank: at rank 1 it is $0.330 - 0.049 = +0.281$; at ranks 2 to 3, $0.216 - 0.077 = +0.140$; at ranks 4 to 10, $0.206 - 0.160 = +0.046$; at rank 11 or worse, $0.082 - 0.066 = +0.017$.

Rank one is what the shipped selector already emits. The strongly marked knowledge is therefore already realised, and what remains findable is the excess at rank two and beyond: 0.203 of the relation. The raw coverage gap implies 0.443. The recoverable figure is roughly two and a half times smaller than the number I, and the published system I was working from, had been quoting.

This also accounts for an observation that previously had no explanation. Ranks four and worse are almost entirely coincidence, and no feature can mark a value as correct when its presence in the pool is an accident of spread. That is the mechanical reason why more than twenty published aggregation mechanisms and ten of my own reach the same limit. They were competing for headroom that is largely not present. Any oracle style ceiling in closed book knowledge base construction needs this control before it is used to direct effort.

6.6 What did not work, and three faults that cost more

Three selection methods that might have been expected to help did not, and the reasoning behind each is the more useful contribution. Cross-frame consensus scored below the best single frame (0.2887 against 0.3196) because the frames share one base model and one prior toward round numbers, so their errors are correlated and they agree on the same incorrect attractors. Routing by draw concentration also scored lower (0.2784 raw, 0.3093 calibrated) because only about a quarter of frame-subject pairs reach support above 0.6, so the cross frame choice is made precisely where no signal exists. And the overshoot scaling reported by a published system did not transfer: all five numeric frames selected a scale of exactly 1.0.

Separately, three implementation faults each cost more than any modelling decision made here, and all three left every self-check passing: integer rounding that removed sub-unit areas, single-linkage clustering that chained across genuinely different answers, and a leakage guard that excluded only a channel's own demonstrations and so produced an apparent $+0.0788$ gain that was entirely an artifact. Appendix B gives all six in full, with the diagnostics that identified them.

7 Analysis

7.1 Measured priors and assumed priors

The test priors measured in Section 2 are not the validation priors. `personHasCityOfDeath` moves from 39.0% to 48.0%, `companyTradesAtStockExchange` from 38.0% to 44.0%, and `countryLandBordersCountry` from 26.5% to 14.9%. All three exceed a five point shift, and they move in opposite directions. I re-centre each threshold by importance weighting

validation rows to the measured test rate, worth 0.0064 overall. The assumption is explicit: difficulty within the empty and non-empty groups is unchanged and only their proportion differs, which is a weaker assumption than treating the splits as interchangeable.

7.2 Where the remaining headroom is located

Of the six relations, borders is at its ceiling (0.9942 with a standard error of 0.0033, and a perfect solver would add 0.0008 overall). `companyTradesAtStockExchange` and `hasArea` are within noise of the best official figures of any other ranked team, ahead by 0.0100 on `hasArea` and behind by 0.0187 on the stock exchange relation. `awardWonBy` cannot be evaluated at ten rows.

That leaves `hasCapacity`, which holds 20.6 percent of the rows and where the entire field scores below 0.33. I tested this conclusion rather than relying on it. Forced choice duels are the one instrument available here that is not a transformation of the sampled frequency table. I generated every pairwise comparison among the top six tolerance separated candidates, in both presentation orders, with eight demonstrations balanced so that the correct answer is at position A four times and is the larger value four times. Aggregating by Borda count, with the rule fixed and committed to version control before the pools were generated, the probe scores 0.3196 on validation against the shipped selector's 0.3608, so it does not improve on the shipped selector.

The diagnostics are more informative than the verdict, because they distinguish a broken instrument from a working instrument reporting an absence. The probe is not degenerate. Position bias is corrected to a mean $P(A)$ of 0.5188, a third of pairs are decided at a margin above 0.25, and on the 580 validation pairs where exactly one side is acceptable the model chooses correctly 62.4 percent of the time, at z of 6.0. The relevant question is what that accuracy consists of. On the same pairs, always choosing the larger of the two numbers scores 60.3 percent, and the probability the model assigns to the larger option rises monotonically with the ratio between them, from 0.567 below $10^{0.15}$ to 0.727 above $10^{0.75}$. That pattern is consistent with a magnitude prior rather than with recall. The comparative knowledge above that prior is 2.1 points at z of 0.7.

I then ran the other non-frequency framing, since a duel and a verification have opposite failure modes: a duel must split its probability mass and so inherits any position or magnitude prior, while a verifier can accept everything. The verification probe presents one claim at a time, with balanced yes and no demonstrations whose false claims are drawn from that demonstration subject's own pool, and scores a candidate by the log odds of yes against no at the next token. It is not a yes-bias readout: it accepts only 28.9 percent of candidates at a positive margin, with a median within row margin spread of 0.250. It scores 0.2680 against 0.3608, a difference of 0.0928 whose 90 percent interval excludes zero from below, and it is worse inside both strata of a split-half stability gate. The probe therefore discriminates between candidates, but in the direction opposite to the one required.

instrument	val., vs 0.3608
cross-frame agreement	0.3308 vs 0.3308, 0.0000
forced choice duel	0.3196, −0.0412
yes or no verification	0.2680, −0.0928

Table 3: Three instruments, each asking the model a different question about the same candidate set, agree.

A fourth line of evidence supports this. Sampled frequency is a Monte Carlo estimate of one functional, so the value of estimating it exactly is the infinite draw limit of the accuracy curve. Fitting accuracy as $a - b/n$ across pool depths from 5 to 100 gives a limit of 0.368 to 0.383, a headroom of 0.008 to 0.022 on the relation. Two disjoint fifty draw halves of the same pool disagree on 38.1 percent of validation rows at almost no cost in accuracy, because the values they disagree between are equally likely to be incorrect.

The available improvement is therefore not a better selector, and not a better aggregator. On this relation the model is estimating within a plausible range, and the range contains the answer often enough for coverage to appear to represent an opportunity.

Limitations

Most of the numbers in this paper are validation figures with a standard error of about 0.021 on the overall metric and 0.03 to 0.05 per relation. Differences smaller than that, including the apparent margins on `hasCapacity` and `personHasCityOfDeath`, are point estimates rather than demonstrated leads, and are reported as such.

Thresholds are selected using validation, so validation is a selection set rather than a clean holdout: every keep or cut decision reads it. With roughly twenty looks and a paired delta standard deviation near 0.008, about one spurious keep carrying 0.015 to 0.019 of validation gain is expected, so the reported validation figure is optimistic by about that amount. That inflates the validation number only: the submitted entry is chosen by observed test score, so the cost is submissions spent discovering it rather than points lost.

The award relation is not meaningfully evaluable at ten rows and I claim no result on it. The validation to test transfer estimates rest on a single edition and one published system. The coverage analysis measures whether an acceptable string appears anywhere in a pool, which upper bounds what a selector could achieve; it is a diagnostic rather than a method. And the per relation choice of numeric aggregator is supported by a confidence interval excluding zero only for `hasArea`; for `hasCapacity` the interval includes zero and the choice ships as provisional.

References

dukesun99. 2026. elicitation-beats-selection: A multi-channel numeric elicitation system for the lm-kbc 2026 challenge. Public code repository, CC BY 4.0. https://github.com/dukesun99/elicitation-beats-selection.

Google. 2026. gemma-4-31B base model. Model weights, revision 5bbc2fb1c1b2c611d06e3d9f23c170ba21659d89. Served in completion mode; 31,273,088,876 parameters measured from the safetensors index.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: The 4th large language models for knowledge base construction challenge, overview and results. In *Proceedings of the 4th Challenge on Language Models for Knowledge Base Construction (LM-KBC) at ISWC*, volume 4041 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jeff Z. Pan, Tuan-Phong Nguyen, and Bohui Zhang, editors. 2024. *Joint Proceedings of the 2nd Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the 3rd Challenge on Language Models for Knowledge Base Construction (LM-KBC) at ISWC*, volume 3853 of *CEUR Workshop Proceedings*. CEUR-WS.org.

ruggsea. 2026. lmkbc2026-submission: A closed-book system for the lm-kbc 2026 challenge. Public code repository. https://github.com/ruggsea/lmkbc2026-submission; no licence file, recipes reimplemented from the method description only.

Sneha Singhania, Tuan-Phong Nguyen, and Simon Razniewski. 2022. LM-KBC: Knowledge base construction from pre-trained language models. In *Proceedings of the Semantic Web Challenge on Knowledge Base Construction from Pre-trained Language Models (LM-KBC) at ISWC*, volume 3274 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain-of-thought reasoning in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*.

A Reproducibility and closed book enforcement

Code, prompts, seeds, threshold grids, and the parameter audit are available at https://github.com/thylinao1/lm-kbc-2026. Pools are regenerable from the committed configuration, and the manifest hash for every pool consumed by a submitted predictions file is recorded alongside that file. The full experimental record, including the runs that failed, is in `docs/engineering-notes.md` in that repository. All figures reported here are produced by the official `evaluate.py` from `lm-kbc/dataset2026`, imported directly rather than parsed from its printed output.

Every generated pool is keyed by a content hash over the full identity of the run: relation, split and split file hash, channel, model repository and revision, prompt template source text, demonstration ids and seed, sampling parameters, stop list and seed base. The hash is the filename and a manifest sits beside it, so a stale pool cannot be reused without this being detected. Draw count is deliberately excluded from the key, and draws are append only with per draw seeds, so raising the number of draws extends a pool instead of invalidating it.

Closed book compliance is enforced by four checks rather than an import grep alone: an extended grep over network libraries, an input al-

lowlist that permits only the three official split files plus generated artifacts and is enforced in code by a single loader, a fork hygiene rule that keeps other participants' repositories outside the published tree and copies no file from them, and offline environment variables on every inference job, because the compute nodes used here have working internet.

B Negative results and implementation faults in full

B.1 Three selection methods that did not work

Cross-frame consensus scores below the best single frame. The hypothesis was that a value surviving a change of register is better evidence than one that is merely modal within a single register. On identical cached draws, consensus scored 0.2887 against 0.3196. The premise was incorrect in a specific way: the frames share one base model and one prior toward round numbers, so their errors are correlated. For a venue whose true capacity is 5000, three frames independently proposed 10000, 10500 and 15000. Agreement among correlated predictors provides little evidence.

Routing by confidence also scores lower. Concentration of draws is strongly predictive of correctness: accuracy rises from 0.000 at support below 0.2 to 0.886 above 0.8. But routing each subject to its most concentrated frame scored 0.2784, and per frame calibrated concentration scored 0.3093, both below the best single frame. The reason is structural: only about a quarter of frame-subject pairs reach support above 0.6, and below that every frame sits near 0.15, so the choice is made precisely where no signal exists. An oracle selecting the correct frame per subject would score 0.3814. The information is present and three methods failed to extract it, which I record as an open problem rather than a result.

Overshoot scaling does not transfer. A published system reports multiplying capacity predictions by 0.95. On this substrate all five numeric frames selected a scale of exactly 1.0 on both train and validation.

B.2 Three implementation faults

All three were found by reading errors rather than by adding compute, and all three are the kind of fault that leaves every self-check passing.

Integer rounding removed small areas. The numeric writer ended by rounding to an integer, which is correct for stadium capacities and destructive for areas: golds of 0.43, 0.176, 0.063 and 0.008482 square kilometres all became "0", which can never match. Five validation subjects were lost this way, every one with the correct value already in its pool. Correcting the formatter moved `hasArea` from 0.7600 to 0.8500, a gain of 0.019 on the overall metric, larger than any modelling change made here.

Single-linkage clustering chains. The published aggregation recipe clusters draws in log space and takes the median of the largest cluster. With enough draws, intermediate values bridge distant ones: 10000, 10500, 12000 and 15000 merge into one cluster spanning fifty percent, whose median is a value no draw proposed and no gold at either end accepts. I replaced it with a fixed radius count: score each distinct drawn value by how many draws fall within the grader's own tolerance of it, and take the argmax. A fixed radius cannot chain, and it optimises the quantity the metric rewards.

A leakage guard correct for one channel and incorrect for two. Threshold tuning excludes a channel's own few shot demonstrations from its train curve, since a demonstration subject has its gold answer in its own prompt. That guard is sufficient for any single channel measurement and insufficient, without this being visible, for any measurement reading two channels at once. Channels draw different demonstration sets from the same 100 train rows, so a subject clean for one frame is frequently a demonstration for another, and a cross frame feature computed on it encodes gold rather than agreement. The union of demonstration subjects is 64 of 100 train rows for `hasCapacity` and 74 of 100 for `hasArea`, reaching all 100 once the 100 shot frame is included, which leaves that relation with no leak free train row at all.

I found this by building a cross frame reranker that keeps the shipped frame as the only proposer of candidates and uses the other frames purely as evidence to reorder them. Pooled over train and validation it read $+0.0788$ on `hasCapacity` and $+0.0476$ on `hasArea`, with a shuffled label control far below and every leave one frame out variant positive. Split by split it was $+0.2206$ on train and exactly $+0.0000$ on validation. On leak free rows only, 133 for `hasCapacity` and 126 for `hasArea`,

the effect is 0.0000 with 8 rows up and 8 down, and 2 up and 2 down. Subject grouped cross validation does not detect this and neither does a label permutation control, because both test whether the model is learning something real while the leak is in the feature for that subject. The only check that detected it was asking which split the gain came from.

Knowledge Base Construction from Language Models: Mechanisms and Calibrations for Multi-Object Relational Extraction under Parameter Budgets

Chengcheng Wei, Ze Chen, Jiewen Zheng, Jiarong He, Max Gao
Interactive Entertainment Group of Netease Inc., Guangzhou, China
{weichengcheng, jackchen, zhengjiewen, gzhejiarong, jgao}@corp.netease.com

Abstract

We describe our system for the AKBC Shared Task 2026, which asks participants to build knowledge base entries from open-weight language models under a strict 32B total-parameter budget, closed-book setting, and no fine-tuning. Our system combines three complementary families of techniques: (i) *diversity-based generation*, which uses multi-question paraphrase prompting and temperature sampling with self-consistency voting to maximize recall; (ii) *gold-caliber post-processing*, a set of generic rules that align model outputs to the gold annotation caliber (e.g., collapsing regional stock exchanges, filtering contested territories, and calibrating capacity estimates by rounding patterns); and (iii) *confidence-based gating*, which uses log-probability signals to reject low-confidence answers and to verify candidates through independent knowledge pathways. On the validation set our best single-model configuration (Mistral-24B, 24B parameters) reaches a macro-F1 of 0.6562. We additionally analyze which strategies transfer across base models and which fail, providing a practical recipe for closed-book KB construction within tight parameter budgets.[1]

1 Introduction

The AKBC Shared Task 2026 asks systems to extract knowledge base entries—sets of object entities for a given subject-entity/relation pair—from open-weight language models. Unlike conventional QA benchmarks, the task emphasizes three difficulties: *completeness* (gold sets may contain dozens of objects), *precision* (hallucinated objects are penalized), and *empty-set discrimination* (correctly answering "none" when the gold set is empty). Systems are constrained to at most 32B total neural parameters, a closed-book setting with no retrieval or external KB access, and no fine-tuning.

[1]Code is available at https://github.com/LexieeWei/akbc2026.

This setting is fundamentally different from fine-tuned or retrieval-augmented pipelines: the parametric memory of a single model is the only source of knowledge. Consequently, our design principle is that *knowledge cannot be created by inference-time techniques; it can only be recalled, verified, and aligned.* We therefore treat each of the six relations as a distinct challenge with a targeted strategy, summarized in Figure 1.

Our main contributions are:

- A multi-question paraphrase voting scheme combined with temperature-based self-consistency, which provides the only form of *effective* diversity we found (Section 3).
- A set of generic *gold-caliber* post-processing rules—including regional-exchange collapse, alias deduplication, and numeric rounding-pattern calibration—that are fully transferable across base models (Section 3.3).
- A systematic analysis of which confidence signals survive under thinking-mode models, where answer-token log-probabilities saturate after reasoning-chain stripping (Section 5).

2 Background

2.1 Task Definition

Given a subject entity s and a relation r, a system outputs a set of object entities $\{o_1, \ldots, o_k\}$. The gold object set may be empty, singleton, or large (up to hundreds of entities, e.g., recipients of a long-running award). Macro precision/recall/F1 are computed per sample and macro-averaged. String relations use normalized exact matching with alias sets and bipartite matching; numeric relations (area, capacity) use a $\pm 5\%$ tolerance.

The six relations are: country person death city, venue capacity, award recipients, company stock exchanges, and geographic area. The validation set contains 475 samples; the test set is hidden.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 456–461
October 28, 2026 ©2026 Association for Computational Linguistics

2.2 Related Work

Prompting language models for knowledge tasks dates to few-shot learning (Brown et al., 2020), and chain-of-thought reasoning (Wei et al., 2022) improves arithmetic and symbolic reasoning. Self-consistency (Wang et al., 2023) improves accuracy by sampling diverse reasoning paths and majority-voting. The annual LM-KBC challenges (Kalo et al., 2025) have established closed-book KB construction as a recurring evaluation setting. Reasoning-capable open-weight models such as DeepSeek-R1 (Guo et al., 2025) and Llama (Touvron et al., 2023) and Mistral (Jiang et al., 2023) provide the base knowledge; we treat the base model as fixed and focus entirely on inference-time mechanisms.

3 Method

Our overall pipeline is shown in Figure 1. Each relation uses one or more of four building blocks: (1) generation with diverse prompts, (2) self-consistency voting, (3) gold-caliber rules, and (4) confidence gating. We describe each block, then give the per-relation configuration.

3.1 Diversity-Based Generation

We found that the only reliable form of diversity is *prompt-view diversity*: asking the same question in several paraphrases (e.g., "On which stock exchanges are shares of X publicly traded?" vs. "Where is X listed?") produces genuinely different recall sets, whereas sampling the same prompt at different temperatures produces correlated answers that majority-voting cannot correct. For each string relation we generate 6–8 paraphrase questions and vote over the union.

Few-shot design. Two aspects of few-shot construction matter more than example selection: magnitude balance and abstention balance. For numeric relations, we sort training examples by answer magnitude and sample five evenly spaced examples spanning the full range; this prevents the model from anchoring on a single magnitude regime and improved hasCapacity F1 by about 0.02. For the city-of-death relation, we observed that the first five training examples happen to contain four empty-gold instances; the model copies the "none" pattern and abstains on 80% of test samples. Replacing them with non-empty examples reduced abstention to 76%, showing that thinking models are particularly sensitive to the abstention pattern in examples.

For numeric relations, temperature sampling (Wang et al., 2023) is useful because numeric answers vary continuously; we combine it with a *magnitude-balanced few-shot* scheme: instead of taking the first k training examples, we sort training examples by answer magnitude and sample uniformly across the range. This prevents anchoring on a single magnitude regime. We found magnitude-balanced few-shot improves hasCapacity validation F1 by about 0.02.

3.2 Self-Consistency Voting

For company and country, majority voting over paraphrase answers with a threshold (≥ 2 of 6–8 questions) robustly filters single-question hallucinations. We found that lowering the threshold to 1 and then attempting to "verify" the extra candidates with yes/no questions does *not* help: the yes/no verification is drawn from the same knowledge source, so the model confidently confirms its own hallucinations. This is consistent with our central negative result that *within-source verification is ineffective*.

3.3 Gold-Caliber Post-Processing

A key observation is that a large fraction of model errors are not knowledge errors but *caliber errors*: the model's output format disagrees with the gold annotation caliber. We define a set of generic, transferable rules:

- **Regional-exchange collapse**: when the answer contains Frankfurt and German regional exchanges (Berlin, Stuttgart, ...), keep only Frankfurt; gold never uses regional exchanges. Similarly, when Tokyo plus Fukuoka and Nagoya all appear, drop the latter two (a shotgun-listing signature).
- **Alias deduplication**: when the same exchange appears under two names ("NYSE" and "New York Stock Exchange"), keep one; bipartite matching counts one true positive, so duplicates only hurt precision.
- **Contested-territory filtering**: drop "Western Sahara" and "Zimbabwe" where gold caliber excludes them, and merge "Bosnia" + "Herzegovina" into "Bosnia and Herzegovina".
- **Rounding-pattern calibration**: for capacity, answers that are round thousands are systematically overestimated; answers that are not round are underestimated. A joint calibration

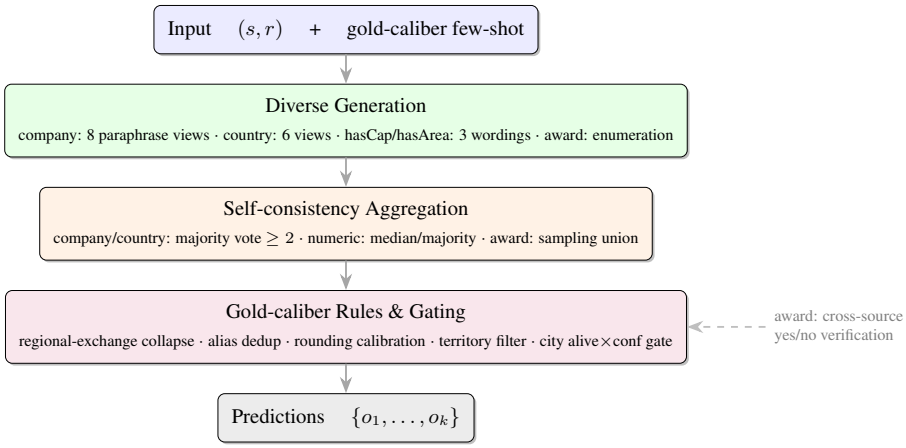


Figure 1: Overview of our pipeline: gold-caliber few-shot prompting, diverse generation, self-consistency aggregation, gold-caliber rules and confidence gating, and prediction output.

($\times 1.05$ for round values $\leq 20{,}000$, $\times 0.95$ for non-round values $< 12{,}000$, else $\times 1.0$) was validated across six independent runs, each improving hasCapacity F1 by about 0.03.

These rules are data-free (derived from the task's annotation caliber), model-agnostic, and consistently transfer across Mistral and Nemotron base models.

3.4 Confidence Gating and Verification

For person death city, we use a two-signal gate: the answer-token average log-probability (confidence) and a separate "Is X still alive?" yes/no log-odds probe. Both signals are orthogonal (confidence captures uncertainty about the death city; the alive probe captures whether the person is alive). This gate improved city F1 from 0.51 to 0.61 on Mistral.

For award recipients, we apply a *verification* step that is genuinely independent: candidates are enumerated with a non-abstention prompt, then verified with "Did X win Y?" using yes/no log-odds. Because a person's biography and an award's recipient list are different knowledge sources, this verification is informative (award F1 improved from 0.001 to 0.24).

Thinking-mode caveat. When deploying a thinking-capable model (Nemotron-3.5-Lightning-30B), we observed that after the server strips the reasoning chain into a separate field, the answer-token log-probabilities become saturated (near zero), because the model emits the answer deterministically after thinking. As a result, log-probability-based gating loses discriminative power under thinking mode; only text-level mechanisms (voting, rules, calibration) remain reliable.

4 Experiments

4.1 Setup

We evaluate on the official validation and test sets. Base models include Mistral-Small-3.2-24B (dense, 24B) and Nemotron-3.5-Lightning-30B (MoE, 30B total), both within the 32B budget and both open-weight. All predictions are generated greedily unless temperature sampling is explicitly described; few-shot $k = 5$ is used by default.

4.2 Reproducibility and Deployment Variance

We observed that numeric-relation outputs vary measurably across runs on the same endpoint: hasCapacity F1 ranged from 0.22 to 0.26 over repeated runs with identical settings, due to vLLM batch scheduling. This implies that (i) single-run validation scores are noisy at the ± 0.005 level, and (ii) small reported improvements must be validated across multiple runs. It also means that "lottery"-style reporting (rerunning and keeping the best run) overstates the expected single-run performance; we report single-run numbers throughout unless noted.

4.3 Base Model Selection

Table 1 shows single-shot baseline results across candidate open-weight models. Mistral-24B is the strongest dense model; a thinking-mode Nemotron-30B achieves a higher baseline, at the cost of log-probability signal saturation.

4.4 Main Results

Table 2 reports per-relation and overall macro-F1 for our best single-model configurations on the validation set.

Model	Macro-F1
Mistral-Small-3.2-24B	0.5625
Nemotron-3.5-Lightning-30B (think)	0.5738
GLM-4-32B	0.5346
Gemma-3-27B	0.5321
Qwen3.5-27B	0.5218

Table 1: Baseline macro-F1 (5-shot, greedy) on the validation set.

Relation	Mistral (val)	Nemotron (val)
awardWonBy	0.237	0.123
company	0.807	0.755
country	0.994	0.998
hasArea	0.720	0.670
hasCapacity	0.309	0.278
city	0.590	0.560
Macro-F1	**0.656**	**0.622**

Table 2: Per-relation macro-F1 on the validation set for our full strategy on each base model.

4.5 Ablation: What Works and What Does Not

Table 3 summarizes the effect of each strategy family. The most consequential findings are:

- **Gold-caliber rules** provide the largest, most transferable gains (company $+0.03$, hasCapacity $+0.03$), and hold across base models.
- **Prompt-view voting** is effective for multi-object relations (company, country) but neutral for single-value relations.
- **Temperature-only self-consistency** is unreliable for string relations (winner's-curse: gains on dev do not reproduce). Combining structured output with higher sampling counts and careful per-relation temperature schedules is a promising direction that we did not fully explore in our final configurations.
- **Within-source verification** (yes/no on the same relation) is consistently ineffective; only cross-source verification (award bio vs. list) works.
- **Log-probability gating** works on non-thinking models but saturates under thinking mode.

4.6 Per-Relation Configuration

Table 4 summarizes the final configuration for each relation. We highlight two design decisions. First, for `companyTradesAtStockExchange`, the system prompt explicitly distinguishes Nasdaq from NYSE and forbids speculation, because the dominant error mode is hallucinated extra exchanges. Second, for `hasCapacity`, the question wording "What is the *seating* capacity?" anchors the model on seat counts rather than venue footprint, which is a stable improvement over the official template.

Prompt example (company). The system prompt states: "List ONLY the stock exchanges where the company's shares are ACTUALLY publicly traded. Do NOT speculate or add exchanges the company might be listed on. A company may be listed on multiple exchanges. If the company is not publicly listed, reply 'none'. Pay attention to the distinction between Nasdaq and NYSE." We found this prompt to be worth about +0.06 F1 on company alone, confirming that *explicit anti-speculation instructions are far more effective on stronger base models*, since they have more knowledge to be misled by.

Few-shot example (hasArea). We render answers as integers (e.g., "74"), and select examples with a magnitude-balanced schedule: sort training answers, then take the 5 evenly spaced examples covering the full range. This prevents the model from anchoring on a single magnitude regime.

5 Thinking-Mode Adaptation

Beyond the 24B dense model, we evaluated thinking-capable models under the same 32B budget, motivated by reasoning-specialized open-weight models (Guo et al., 2025). When serving Nemotron-3.5-Lightning-30B with a reasoning parser, the server strips the reasoning chain into a separate field and returns the final answer in the content field. We encountered three practical issues:

(1) Log-probability saturation. After stripping, the answer-token log-probabilities are renormalized to near zero: the model emits the answer deterministically after thinking, so the answer tokens carry almost no discriminative probability. Confidence-based gating (e.g., the city death-city gate) loses all discriminative power under thinking mode. We confirmed this by inspecting the token-level log-probabilities: the answer tokens $2, 5, 0, 0$ all had log-probability ≈ 0.

Strategy	Target	ΔF1	Mechanism
Paraphrase voting (6–8 views)	company/country	+0.02–0.03	prompt-view diversity
Regional-exchange collapse	company	+0.02	gold caliber
Alias deduplication	company	+0.01	gold caliber
Capacity rounding calibration	hasCapacity	+0.03	numeric calibration
Magnitude-balanced few-shot	hasCapacity	+0.02	anchoring
Confidence + alive gating	city	+0.06	log-prob signals
Cross-source award verification	award	+0.01	independent pathway
Temperature self-consistency	company	≤ 0	unstable
Within-source yes/no verification	company	≤ 0	same source
Log-prob under thinking mode	all	saturates	re-normalized

Table 3: Ablation of strategy families on the validation set.

Relation	Strategy	Macro-F1
awardWonBy	short-name enumeration, sampling union, yes/no verification, bio re-check, cleaning	0.237
company	8-view paraphrase voting, PE prompt, gold-caliber rules	0.807
country	6-view voting, completeness probing, filtering	0.994
hasArea	"total area" prompt, integer 5-shot	0.720
hasCapacity	"seating capacity" prompt, magnitude-balanced few-shot, rounding calibration	0.309
city	greedy + alive/confidence dual gate	0.590

Table 4: Per-relation configuration and validation macro-F1 on Mistral-24B.

(2) Answer extraction from the reasoning chain. Some endpoints do not strip the reasoning chain at all, returning the chain inside the content field. We implemented a generic stripper that handles both standard `<think>...</think>` blocks and the Nemotron format, where the chain begins with "Here's a thinking process:" and ends with a lone `</think>` followed by the answer. We additionally handle an occasional malformed response where a "to=user" or "to=self" prefix appears before the answer or chain.

(3) Few-shot induction of abstention. We found that the first five city-of-death training examples happen to contain four empty-gold instances; the thinking model learns the pattern and answers "none" for 80% of test samples. Replacing them with non-empty examples only marginally reduced abstention (80%→76%), showing that for thinking models the abstention is primarily a conservative model behavior rather than pure few-shot induction.

These issues mean that for thinking-mode models, only text-level mechanisms (voting, rules, calibration) remain reliable; the log-probability-based mechanisms that are most effective on the dense model must be disabled.

6 Failure Analysis

We summarize strategies that failed and why:

- **Within-source verification**: verifying extra company candidates with yes/no probes *decreased* test F1 (0.7854→0.7737) because the probe shares the same knowledge source as enumeration, so the model confidently confirms its own hallucinations. Verification is informative only when it invokes a genuinely independent pathway (as in our award verification).

- **Retrieval-based few-shot**: per-example magnitude retrieval, using the model's own answer as a proxy, degraded hasCapacity F1 (0.2347 vs. 0.2449) because the proxy is noisy.

- **Temperature-only voting for string relations**: gains on development did not reproduce on test (winner's curse). Combining structured output with higher sampling counts and careful temperature schedules is a promising direction, but our final configurations did not adopt it.

7 Discussion

Deployment drift. We observed that the same model served at different times produces measurably different outputs for numeric relations (hasCapacity F1 ranged from 0.22 to 0.26 across runs on the same endpoint). This batch-to-batch variance means that single-run scores on the validation set are noisy, and small improvements must be validated across multiple runs before being trusted. It

also means that "lottery"-style reporting—running the same pipeline several times and reporting the best run—inflates scores relative to the expected single-run performance.

Model-level lessons. Across more than seventy method experiments, three consistent conclusions emerged:

1. *Diversity must come from prompt views, not temperature.* Memory errors are highly correlated across samples of the same prompt; only genuinely different question phrasings recall different subsets of knowledge.

2. *Log-probabilities reveal uncertainty, not correctness.* Confidence signals reliably detect "the model is not sure," but cannot detect "the model is confidently wrong." Verification helps only when it invokes a different knowledge pathway.

3. *Knowledge is bounded, not methods.* Under a closed-book, no-fine-tuning budget, no inference-time technique can create knowledge the model does not possess; the available lever is aligning outputs to the gold annotation caliber.

Cross-model transferability. Because we ran the same strategy families on both a dense (Mistral-24B) and a thinking (Nemotron-30B) base, we can separate transferable mechanisms from model-specific ones. The gold-caliber rules (regional-exchange collapse, alias deduplication, rounding calibration) transfer essentially unchanged: they correct systematic output-caliber errors that are common to both models. Prompt-view voting also transfers, provided the base model can answer the paraphrase questions. In contrast, confidence gating transfers only to non-thinking models: the log-probability saturation under thinking mode (Section 5) destroys the discriminative signal. Temperature sampling behaves differently as well: on the thinking model it is more expensive (each sample incurs a full reasoning chain) and the vote threshold must be retuned. These observations suggest that the *caliber-correction layer* is the most portable component of our system, while confidence mechanisms are tightly coupled to the model's inference mode.

Future work. We plan to combine our gold-caliber rules with structured-output self-consistency pipelines, and to develop confidence mechanisms that remain discriminative under thinking-mode models—for example, by probing the model's uncertainty about its own reasoning chain rather than the answer tokens.

8 Conclusion

We described a practical system for closed-book KB construction under a 32B parameter budget. The central lesson is that *knowledge cannot be engineered into a model at inference time*; the available lever is the alignment of model outputs to the gold annotation caliber, through prompt-view diversity, gold-caliber rules, and confidence gating where log-probabilities remain discriminative. Our best validation configuration reaches 0.6562 macro-F1 with a compliant 24B model. Future work includes combining our gold-caliber rules with structured-output self-consistency pipelines and adapting confidence mechanisms to thinking-mode models.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. 2023. Mistral 7b. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. Lm-kbc 2025: 4th challenge on knowledge base construction from pretrained language models. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 4th Challenge on Language Models for Knowledge Base Construction (LM-KBC 2025), co-located with ISWC 2025*, CEUR Workshop Proceedings. CEUR-WS.org.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, et al. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Pre-training Aware Directed Reasoning for Closed-Book Knowledge Base Construction

Mikołaj Tym
Poznań University of Economics and Business
Poznań, Poland
mikolaj.tym@ue.poznan.pl

Abstract

Extracting facts from the parametric memory of Large Language Models (LLMs) for Automated Knowledge Base Completion (AKBC) under closed-book constraints remains a significant challenge. We describe our system for the 2026 AKBC Shared Task, where retrieval, external knowledge sources and fine-tuning are not permitted. We first evaluate 18 open-weight models on the training set to check how much the choice of the model matters for this task. The results differ significantly, from 0.186 to 0.499 macro F1, and the best model is not the largest one. Moreover, the models that score well on coding benchmarks tend to score worse on knowledge base construction. We then introduce **Pre-training Aware Directed Reasoning (PADR)**, a set of relation-specific reasoning pipelines optimized for the best evaluated model, Mistral Small 3.2 24B. PADR improves the macro F1 score on the training set from 0.499 to 0.614. On the shared task's test set, PADR reaches 0.629 macro F1, more than double the official baseline's 0.296. The code is available at https://github.com/MikolajTym/akbc-padr.

1 Introduction

The capabilities and usefulness of Large Language Models (LLMs) might be significantly extended by using techniques like Retrieval-Augmented Generation (RAG) (Lewis et al., 2020). These techniques allow to retrieve relevant and reliable information, so the LLM can process it instead of relying on its parametric knowledge. However, the retrieval process requires a Knowledge Base (KB) that provides the reliable information, and maintaining such a KB requires significant effort.

Automated Knowledge Base Completion (AKBC) focuses on utilizing LLMs for this task (Chai et al., 2021), and the retrieval approach can be very useful here (Prabhong et al., 2024). However, if we want to create a KB that covers a new topic, we cannot use retrieval, because the KB for this topic does not exist yet. Therefore, an interesting research area is the AKBC task under closed-book rules, i.e., no RAG, no web search tools, no knowledge graphs and no fine-tuning on data that covers the specific topic.

These rules are required by the 2026 AKBC Shared Task (Kalo et al., 2026), the fifth edition of the LM-KBC challenge (Kalo et al., 2025). The objective is to generate all object entities for a given subject entity and relation, i.e., to complete Wikidata triples (Vrandečić and Krötzsch, 2014). The solution must therefore rely on the model's parametric knowledge and on approaches that retrieve it properly. In the 2025 edition participants were forced to use the Qwen3-8B model (Yang et al., 2025). However, the 2026 edition allows any open-weight system within a 32B parameter budget, so the choice of the model becomes a part of the solution.

This work focuses on the following research questions:

1. Do open-weight models differ significantly in their capability to perform the closed-book AKBC task?
2. What explains these differences, the model size or the training set and behaviour that pre-training induces?
3. How much does relation-specific directed reasoning improve the best model, and for which relations?

We answer the first two questions by evaluating 18 models with a single closed-book configuration, and the third one by building a separate reasoning pipeline for each relation. Our submission reaches 0.629 macro F1 on the test set.

2 Related Work

Petroni et al. (2019) showed that pre-trained mod-

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 462–472
October 28, 2026 ©2026 Association for Computational Linguistics

els answer factual queries without any fine-tuning. The LM-KBC challenges extend this idea to complete KB entries (Kalo et al., 2025). The 2025 edition already delivered effective methods for this task. These methods include relation-wise self-consistency, self-retrieval with divide-and-conquer decomposition, and an LLM that judges its own candidates (Albert-Roulhac and Zouaq, 2025; He and Razniewski, 2025; Clay et al., 2025). Some of these methods reuse techniques that are already well known, i.e., Chain-of-Thought (Wei et al., 2022) and self-consistency (Wang et al., 2023).

However, every participant of the 2025 edition worked with the same model, so these methods were developed for one fixed set of parametric knowledge. The 2026 edition changes this rule, and the model becomes a design decision instead of a constant. Therefore, we focus on this decision, as the model's knowledge depends on its size and pre-training data (Roberts et al., 2020).

3 Task and Data

The input is a pair of a subject entity and a relation, and the output is the full set of object entities expressed as strings. A subject may have zero, one or many objects. For example, for the subject "France" and the relation `hasArea`, the model should generate "632702", which is the area of France in square kilometres.

The dataset covers six relations and contains 477 training, 475 validation and 475 test subjects (Kalo et al., 2026). Systems are ranked by the F1 score macro-averaged over all rows in dataset. Predicted strings are normalized and matched against the gold labels and their aliases, and a numeric prediction is correct when it falls within a 5% relative tolerance.

4 Methodology

4.1 Initial Closed-Book Approach

The 2026 edition does not enforce a particular model, so we first verify how different models perform on the AKBC task. For each relation we implement the same two-step strategy. First, the LLM is queried with a direct, question-style prompt, e.g., "What is the capacity of {subject_entity}?".

Simple prompts often cause the model to generate responses that cannot be parsed and evaluated, e.g., because of the model's conversational style. Therefore, the second step uses structured output[1]. This approach forces the model to return a JSON object that conforms to a schema defining null values, relation cardinality and data types such as integers or floats. We refer to this configuration as the Initial Closed-Book approach.

4.2 Pre-training Aware Directed Reasoning

The next part of our research focuses on optimizing the pipeline for one selected model. We introduce **Pre-training Aware Directed Reasoning (PADR)**, a set of relation-specific prompting and extraction strategies. We call the strategy pre-training aware because it aligns the reasoning path and the prompt structure with the next-token prediction task used during causal pre-training (Radford et al., 2019). Four mechanisms are shared by the relation-specific pipelines.

Directed reasoning and structured extraction. Every pipeline separates recalling a fact from formatting it. Directed reasoning means that we prescribe the reasoning path for each relation instead of letting the model decide how to think about the answer. The reasoning part therefore follows a fixed, relation-specific structure, e.g., resolving entity ambiguities, recalling the fact and justifying a decision, before a separate call turns that reasoning into the final answer. This order, reasoning before decision, matters for a model trained with next-token prediction, because each token is generated conditioned on the tokens before it, so a decision that follows its justification can build on that reasoning, while a decision written first cannot. Generating the context from parametric memory before answering resembles the self-retrieval strategy that was already applied to this task in 2025 (He and Razniewski, 2025). Each reasoning call is followed by its own extraction call, which reads that text and returns a JSON object under a grammar-constrained schema. The schema encodes the output contract, i.e., types, nullability, cardinality and closed vocabularies (e.g., a finite list of stock exchanges).

Targeting the pre-training format. Instead of asking a question, some prompts ask the model to recall the Wikipedia infobox fields for the subject. Wikipedia is extensively represented in pre-training corpora, and the infobox is a good source of facts. Therefore, reproducing that format may trigger the memorized fact more reliably than a conversational question.

[1] `https://reference.langchain.com/python/langchain-core/language_models/chat_models/BaseChatModel/with_structured_output`

Inverse verification. The reversal curse means that a model that learned "A is B" struggles to answer "B is A" (Lv et al., 2024). For `awardWonBy` the hard direction is the required one, because the query gives the award and asks for its winners. Therefore, we generate candidates in the hard direction and filter them in the easy direction, by asking whether a given candidate won the given award.

Agreement as a decision signal. For two relations we ask the same question again in forms that cannot copy the first answer, and we sample at a higher temperature. A similar approach, relation-wise self-consistency, applied to this task in the 2025 edition (Albert-Roulhac and Zouaq, 2025), also samples several answers for the same question. However, self-consistency returns the most frequent answer. We use the samples differently, because we measure the ratio of them that agree and treat that ratio as a confidence score. The ratio then decides which reasoning path should be followed.

4.2.1 Relation-Specific Pipelines

For `countryLandBordersCountry`, the reasoning prompt asks for the neighbours of the subject country and, for each of them, for the length and the physical nature of the border. Asking for a justification suppresses maritime neighbours, which the task definition excludes. The prompt also mentions overseas territories and enclaves, because the ground truth counts borders through integral territories.

For `hasArea`, the reasoning prompt follows three guided steps, i.e., identifying the geographic entity, separating it from entities with similar names, and recalling its official area. The prompt states that the value must be the surface of the entity itself and not of the surrounding waters.

For `companyTradesAtStockExchange`, the first stage generates a company profile that resolves the entity and states whether the company is public, private or a subsidiary. This stage also lists explicit rules against guessing a local exchange from the country of origin, because the model tends to answer with the exchange of the company's home country. The second stage selects the listings, and the list of valid exchanges is enforced. Therefore, the model cannot emit an exchange outside the vocabulary.

For `personHasCityOfDeath`, the correct answer is an empty list when a person is still alive or when the place of death is not recorded. Therefore, the decision is not only which city to answer, but whether to answer at all. We query each subject under three framings, i.e., the direct question, a request for the Wikipedia infobox fields, and a step-by-step disambiguation that first asks which person bearing the name is meant. Each framing is sampled three times at temperature 1.0, which gives nine independent answers. A city is predicted only when at least two framings name it and it carries at least a third of the nine runs, and otherwise the system returns an empty list. This approach mitigates hallucination, i.e., cases where the model describes different biographical profiles for the same person. In such cases, the empty prediction is correct if the person is still alive.

For `hasCapacity`, a first reasoning stage builds a venue profile that separates the target facility from training grounds and school fields with similar names. The second stage asks for the capacity. It prioritizes the value in the Wikipedia infobox over record attendance figures. It also forbids rounding. PADR treats this estimate as a hypothesis. It asks the same question again in eight forms, i.e., three choices from a list of options and five free answers sampled at temperature 0.9 without the profile. Venues whose estimate is confirmed by at least five of these answers are correct 55% of the time, against 23% overall. The number of confirmations then decides how many values to return. The metric divides precision by the length of the answer list. Therefore, k values containing the correct answer score $2/(k+1)$ instead of zero. We return the estimate alone when at least five answers confirm it, and two or three common capacities for venues of that size otherwise. Even when the model cannot recall a venue, it usually places it in the right order of magnitude.

For `awardWonBy`, an initial stage resolves the award by identifying its grantor and its active lifespan. Next, parallel calls query five-year windows of that lifespan and explicitly decompose collective winners into their individual members. A similar temporal decomposition was already used for this relation in 2025 (He and Razniewski, 2025), though with fixed decades rather than windows tied to the award's own lifespan. Temporal chunking is necessary because post-trained models tend to refuse a single query that requires thousands of output tokens for hundreds of recipients. Finally, an inverse verification step checks each candidate recipient separately. The verification schema places

the justification before the verdict. Therefore, the model states its reason first.

5 Experiments

5.1 Setup

The cross-model comparison is conducted with Ollama[2] as the model provider. Ollama serves models in the GGUF format with quantized weights, which affects their performance and behaviour (Dettmers and Zettlemoyer, 2023). The PADR experiments use Mistral Small 3.2 24B (Mistral AI Team, 2025) served by vLLM (Kwon et al., 2023) at full precision, with a fixed seed and batch-invariant kernels. The same model therefore occupies about 45 GB under vLLM and about 15 GB under Ollama, so the two configurations are not directly comparable.

When retrieving factual knowledge, it might seem logical to set the sampling temperature to 0. However, in our experiments a temperature of 0 caused the models to enter infinite reasoning loops and produce repetitive text. Therefore, for each model we use the temperature recommended on its Ollama page, which is why this value differs between the models. There are exceptions from this rule when testing model's self-consistency for `personHasCityOfDeath` and `hasCapacity`. All experiments run on a NVIDIA DGX Spark workstation, which combines a GB10 Grace Blackwell chip with 128 GB of unified memory[3]. The full test run takes about 30 minutes on this GPU.

The prompts and the thresholds for `hasCapacity` and `personHasCityOfDeath` were selected on the validation set. Each relation contains at most 100 subjects, which is indeed too little data for the tuned parameters to generalize well. Therefore, the validation scores for these two relations should be read as optimistic. A system tuned directly on the test set would likely score higher there too. However, such a result would not show that the system generalizes, so we never use the test set for tuning.

5.2 Cross-Model Comparison

Our objective is to determine which model is best suited for the AKBC task. Therefore, we evaluate 18 models with the Initial Closed-Book approach on the challenge's training set. Each organisation uses its own training data. We hypothesise that some of these datasets cover the domain knowledge required by our target relations more heavily. It means that even a model which underperforms on general benchmarks may be beneficial here.

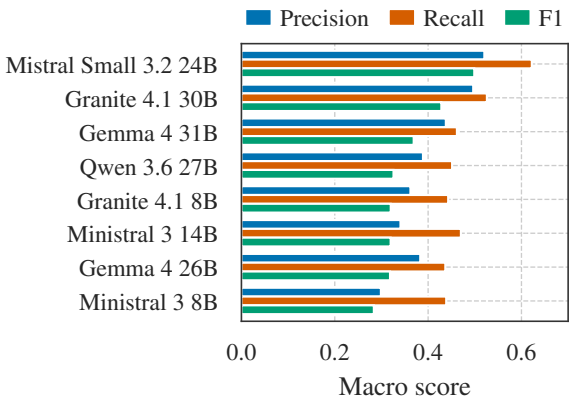


Figure 1: Initial Closed-Book approach macro precision, recall and F1 for the eight best models on the training set, sorted by macro F1.

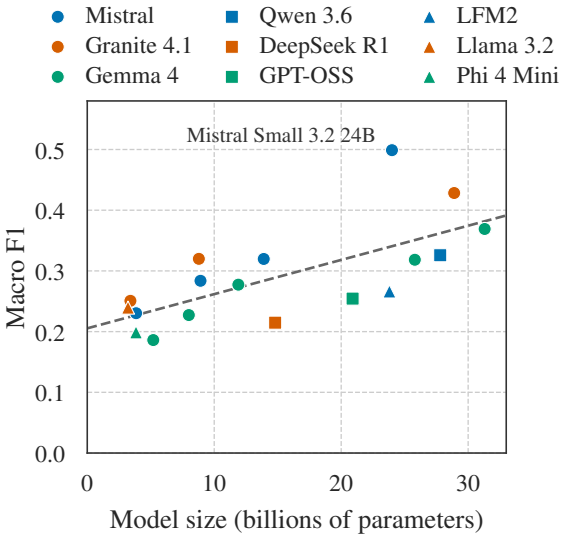


Figure 2: Model parameter count against the macro F1 score of the Initial Closed-Book approach on the training set.

Figure 1 shows the eight best models. The macro F1 scores for all 18 models span from 0.186 for Gemma 4 E2B to 0.499 for Mistral Small 3.2 24B. It means that the choice of the model alone has crucial impact on performance.

Figure 2 relates the score to the parameter count. Model size generally correlates with the score. However, the largest model does not perform the best. Moreover, models from the Mistral and Granite (IBM Granite Team, 2026) families perform above the average trend for their respective sizes.

[2] `https://ollama.com`

[3] `https://www.nvidia.com/en-us/products/workstations/dgx-spark/`

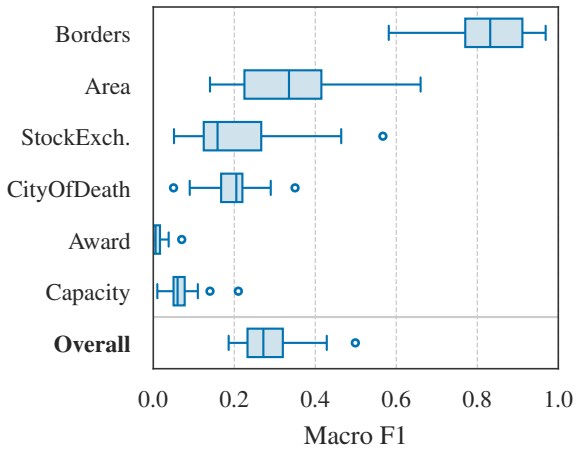


Figure 3: Distribution of macro F1 across the relations for all 18 models evaluated with the Initial Closed-Book approach.

Mistral Small 3.2 24B performs best on almost all relations, and it ranks second only on `personHasCityOfDeath` and `awardWonBy`, where Granite 4.1 30B is slightly better. Its success is caused by a high recall for all relations. We observe that reasoning models, such as Qwen 3.6 27B (Qwen Team, 2026) and Gemma 4 31B (Gemma Team, 2026), tend to hesitate when they generate the answers. These models often enter infinite reasoning loops where they repeatedly question and verify their own assertions. They also mention the need to call a web search tool. This tool is not available for the AKBC task. However, the models were trained to use such abilities. It means that not only the model's knowledge but also its behaviour affects the performance.

But, why are these families better than others? When we inspect the same models on coding benchmarks, it turns out that Mistral and Granite are significantly worse. Terminal-Bench evaluates how a model solves programming tasks through a command line interface (Merrill et al., 2026). Gemma 4 31B achieves 43.4% there, while Mistral Small 3.2 achieves 5.6% and Granite 4.1 30B achieves 2.6%[4]. We hypothesise that models trained heavily on code must allocate a large part of their parametric memory to programming languages, API syntax and technical documentation. This pre-training improves their coding and logical reasoning abilities. However, the volume of syntax data occupies memory that would otherwise store general world facts.

The difficulty also differs strongly between the relations (Figure 3). The relation `countryLandBordersCountry` is relatively easy, with a median F1 of 0.832 across the 18 models. However, the same configuration fails almost completely on `hasCapacity` and `awardWonBy`, with median F1 scores of 0.060 and 0.006.

5.3 PADR Optimizations

We optimized the PADR approach utilizing the best model from our cross-model comparison, Mistral Small 3.2 24B.

Table 1 compares the Initial Closed-Book approach with PADR on the training set. PADR improves the overall macro F1 from 0.499 to 0.614, and the gain comes mostly from precision, which rises from 0.520 to 0.687.

The largest gain is on `awardWonBy`, from 0.038 to 0.521. During the initial evaluation the model can neither generate a long list of winners nor identify the exact award, and the temporal chunking with inverse verification solves both problems. The second largest gain is on `personHasCityOfDeath`, from 0.290 to 0.520, where precision rises from 0.310 to 0.770. This gain comes from the subjects on which the pipeline now declines to answer. On `companyTradesAtStockExchange` the profile stage resolves entity ambiguities and confirms the ownership status before the listings are queried, which raises the F1 score from 0.567 to 0.681.

These two configurations differ in the serving stack, because the initial evaluation used quantized GGUF weights and PADR uses full precision. This may affect the performance and model's behaviour.

5.4 Validation and Test Results

On the validation set, PADR reaches 0.662 overall macro F1, while the official baseline reaches 0.313. Figure 4 presents our test results against the official baseline of the shared task. PADR reaches 0.629 overall macro F1, while the official baseline reaches 0.296.

The approach performance transfers well between the dataset splits. The only relation whose score moves substantially between splits is `awardWonBy`: 0.521 on training, 0.232 on validation and 0.355 on test. This relation contains only 10 subjects per split, so a few errors move the macro score by a large amount. The model also cannot properly disambiguate many of the awards, on validation and on test alike.

[4] https://artificialanalysis.ai/evaluations/terminalbench-v2-1

Relation	Initial Closed-Book			PADR		
	Precision	Recall	Macro F1	Precision	Recall	Macro F1
`countryLandBordersCountry`	0.989	0.956	0.969	0.980	0.988	**0.981**
`personHasCityOfDeath`	0.310	0.690	0.290	0.770	0.630	**0.520**
`hasCapacity`	0.210	0.210	0.210	0.192	0.290	**0.218**
`awardWonBy`	0.303	0.021	0.038	0.733	0.553	**0.521**
`companyTradesAtStockExchange`	0.610	0.765	0.567	0.787	0.792	**0.681**
`hasArea`	0.660	0.660	0.660	0.800	0.800	**0.800**
Overall	0.520	0.622	0.499	0.687	0.677	**0.614**

Table 1: Macro precision, recall and F1 of the Initial Closed-Book approach and of PADR for Mistral Small 3.2 24B, measured on the training set.

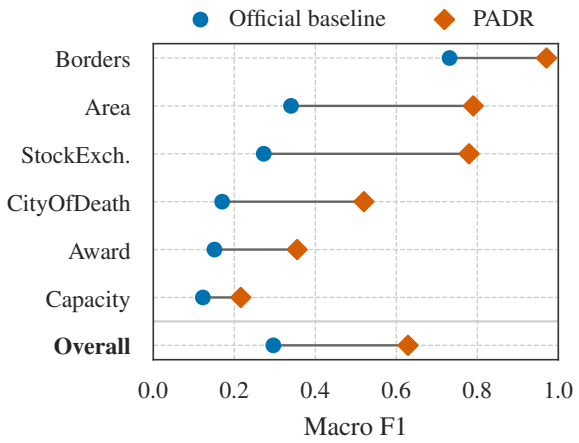


Figure 4: Macro F1 per relation on the test set, against the official Qwen3.5-9B baseline.

5.5 Error Analysis

We inspect the validation predictions with the official evaluation script and group the errors into four categories. The most frequent failure is a wrong value on a subject that the model does not know. It dominates `hasCapacity`, where the model produces a plausible number for almost every venue, and the agreement level can only decide how many values to return. The second category is entity confusion, which affects `awardWonBy` and `personHasCityOfDeath`. For one validation person the model recalls an athlete under one framing and a pianist under another, and it names six different cities across the nine runs. The third category is a hallucinated answer where the gold set is empty, and PADR reduces it the most, because our system returns an empty list for 117 of the 475 validation subjects. The last category is missed recall, and it is the price of the abstention rule, because on the validation set the recall of `personHasCityOfDeath` is 0.620 while its precision is 0.800.

6 Conclusions

RQ1. The models differ significantly, because the same closed-book configuration yields between 0.186 and 0.499 macro F1 depending only on the model.

RQ2. Size explains only a part of the difference. Mistral and Granite perform above the trend for their sizes, and both families score poorly on coding benchmarks. Moreover, models trained to verify their own answers hesitate, loop and ask for tools that are not available. Other prompts or a different way of using these models could raise their scores further. However, the Initial Closed-Book approach applies a fairly simple solution, so a model that already performs well here shows its own quality rather than a clever prompt.

RQ3. Relation-specific directed reasoning improves the best model from 0.499 to 0.614 macro F1 on the training set. However, the improvement is not uniform, because it is large for the relations that the direct question handles badly and absent for the relations that were already close to the ceiling.

We presented our system for the 2026 AKBC Shared Task. The evaluation of 18 models shows that the training data and the trained behaviour affect AKBC performance. It means that the parameter count alone does not indicate how well a model completes a knowledge base, so the model should be selected for this task and not only for its size. Most models are currently trained to use RAG or web search, but for the closed-book task the models that answer alone could be the best ones. PADR then improves the best model by directing its reasoning path separately for each relation, and our submission reaches 0.629 macro F1 on the test set.

Limitations

A key limitation of PADR is that applying it to a new relation requires manual human design. The directed reasoning paths must be tailored to the semantics of the relation and to the format in which the fact was likely memorized. Future research could develop an agent that constructs these paths automatically, for example by selecting inverse verification only for the relations where it applies. However, at present the design still requires human intervention, which limits the generalization of the approach.

The thresholds for decision algorithms were selected on the validation set, which has at most 100 subjects per relation. Their validation scores are therefore optimistic. Moreover, `awardWonBy` has only 10 subjects and `countryLandBordersCountry` has 68, so their macro scores are sensitive to individual errors.

Despite the fixed seed and batch-invariant kernels, two full runs of the pipeline do not always return the same prediction for a given subject, and this affects every relation to some degree. Most predictions repeat, or stay equally wrong, across runs, but there are some exceptions, especially for the relation `hasCapacity`.

Finally, the graded answer set for `hasCapacity` exploits the fact that the metric gives partial credit for a short list of candidates. This behaviour would not transfer to a knowledge base that accepts only a single value.

Declaration on Generative AI

The authors used LLMs for grammar and spelling checks. After using these tools, the authors reviewed and edited the content and take full responsibility for the publication's content.

Acknowledgments

This publication is based upon work from COST Action CA23147 GOBLIN – Global Network on Large-Scale, Cross-domain and Multilingual Open Knowledge Graphs, supported by COST (European Cooperation in Science and Technology, https://www.cost.eu).

References

Edouard Albert-Roulhac and Amal Zouaq. 2025. Rewise: Relation-wise self-consistency for LLM probing. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025) co-located with the 24th International Semantic Web Conference (ISWC 2025), Nara, Japan, November 2, 2025*, volume 4041 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Junyi Chai, Yujie He, Homa Hashemi, Bing Li, Daraksha Parveen, Ranganath Kondapally, and Wenjin Xu. 2021. Automatic construction of enterprise knowledge base. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 11–19, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alex Clay, Ernesto Jiménez-Ruiz, and Pranava Madhyastha. 2025. Judge, generator, executioner: Utilizing an LLM for KBC. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025) co-located with the 24th International Semantic Web Conference (ISWC 2025), Nara, Japan, November 2, 2025*, volume 4041 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Tim Dettmers and Luke Zettlemoyer. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7750–7774. PMLR.

Gemma Team. 2026. Gemma 4 technical report. *Preprint*, arXiv:2607.02770.

Jingbo He and Simon Razniewski. 2025. Combining self-retrieval-augmented generation with divide-and-conquer for language model-based knowledge base construction. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025) co-located with the 24th International Semantic Web Conference (ISWC 2025), Nara, Japan, November 2, 2025*, volume 4041 of *CEUR Workshop Proceedings*. CEUR-WS.org.

IBM Granite Team. 2026. IBM granite 4.1 language models. Hugging Face.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pretrained language models. In *Joint Proceedings of the*

3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025) co-located with the 24th International Semantic Web Conference (ISWC 2025), Nara, Japan, November 2, 2025, volume 4041 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Ang Lv, Kaiyi Zhang, Shufang Xie, Quan Tu, Yuhan Chen, Ji-Rong Wen, and Rui Yan. 2024. An analysis and mitigation of the reversal curse. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13603–13615, Miami, Florida, USA. Association for Computational Linguistics.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, and 66 others. 2026. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv e-prints*, arXiv:2601.11868.

Mistral AI Team. 2025. Mistral small 3.2 24b instruct. Hugging Face.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Thin Prabhong, Natthawut Kertkeidkachorn, and Areerat Trongratsameethong. 2024. KGC-RAG: knowledge graph construction from large language model using retrieval-augmented generation. In *Joint proceedings of the 2nd workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM 2024) and the 3rd challenge on Language Models for Knowledge Base Construction (LM-KBC 2024) co-located with the 23nd International Semantic Web Conference (ISWC 2024), Baltimore, USA, November 12, 2024*, volume 3853 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Qwen Team. 2026. Qwen3.6. GitHub repository.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

A Detailed Validation and Test Results

Table 2 presents the complete validation results of PADR, including the macro precision and recall, the micro-averaged F1, the average number of predicted objects per subject and the number of empty predictions.

Table 3 presents the complete test results of PADR and of the official baseline, as scored by the organizers' leaderboard.

B Cross-Model Evaluation

Figure 5 extends Figure 1 to all 18 evaluated models. Tables 4, 5 and 6 present the macro F1, precision and recall per relation for every model. All these runs use the training set and quantized GGUF weights.

Relation	P	R	F1	Micro	Empty
Borders	0.969	0.989	0.970	0.975	17
CityOfDeath	0.800	0.620	0.550	0.442	57
Capacity	0.287	0.495	0.345	0.310	0
Award	0.629	0.231	0.232	0.224	3
StockExch.	0.873	0.840	0.765	0.740	40
Area	0.810	0.810	0.810	0.810	0
Overall	0.733	0.725	0.662	0.393	117

Table 2: PADR on the validation set. **P**, **R** and **F1** are macro-averaged, **Micro** is the micro-averaged F1 and **Empty** is the number of empty predictions.

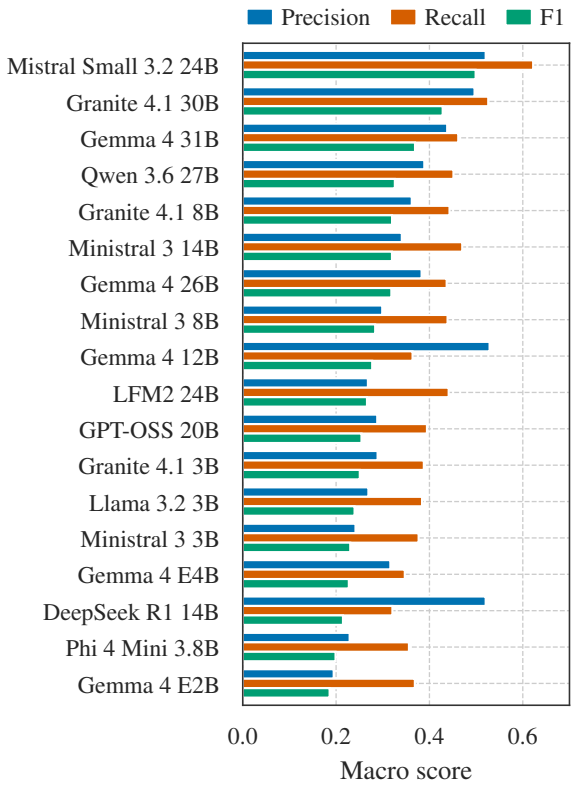


Figure 5: Initial Closed-Book approach macro precision, recall and F1 for all 18 evaluated models on the training set, sorted by macro F1.

Relation	Official Baseline			PADR		
	Precision	Recall	Macro F1	Precision	Recall	Macro F1
countryLandBordersCountry	0.742	0.889	0.732	0.976	0.985	**0.971**
personHasCityOfDeath	0.170	0.650	0.170	0.780	0.640	**0.520**
hasCapacity	0.122	0.122	0.122	0.191	0.286	**0.216**
awardWonBy	0.306	0.141	0.151	0.643	0.322	**0.355**
companyTradesAtStockExchange	0.292	0.707	0.273	0.854	0.829	**0.780**
hasArea	0.340	0.340	0.340	0.790	0.790	**0.790**
Overall	0.305	0.511	0.296	0.701	0.680	**0.629**
Zero-object cases	0.000	0.000	0.000	0.712	0.824	**0.764**

Table 3: Macro precision, recall and F1 of the official baseline and of PADR, measured on the test set by the organizers' leaderboard.

Model	Overall	Borders	CityOfDeath	Capacity	Award	StockExch.	Area
Mistral Small 3.2 24B	**0.499**	**0.969**	0.290	**0.210**	0.038	**0.567**	**0.660**
Granite 4.1 30B	0.428	0.943	**0.350**	0.110	**0.070**	0.405	0.540
Gemma 4 31B	0.369	0.961	0.220	0.080	0.025	0.464	0.350
Qwen 3.6 27B	0.326	0.916	0.220	0.070	0.009	0.170	0.480
Granite 4.1 8B	0.320	0.889	0.210	0.040	0.018	0.278	0.400
Ministral 3 14B	0.320	0.844	0.200	0.140	0.007	0.059	0.560
Gemma 4 26B	0.318	0.926	0.130	0.100	0.034	0.285	0.380
Ministral 3 8B	0.284	0.851	0.200	0.060	0.000	0.103	0.420
Gemma 4 12B	0.277	0.898	0.220	0.030	0.003	0.231	0.240
LFM2 24B	0.266	0.793	0.190	0.060	0.012	0.147	0.340
GPT-OSS 20B	0.254	0.715	0.200	0.070	0.000	0.145	0.320
Granite 4.1 3B	0.251	0.820	0.160	0.050	0.002	0.217	0.220
Llama 3.2 3B	0.240	0.766	0.090	0.070	0.000	0.139	0.330
Ministral 3 3B	0.230	0.819	0.220	0.050	0.000	0.060	0.220
Gemma 4 E4B	0.227	0.782	0.260	0.040	0.004	0.120	0.140
DeepSeek R1 14B	0.215	0.581	0.230	0.010	0.007	0.194	0.200
Phi 4 Mini 3.8B	0.199	0.637	0.130	0.050	0.000	0.051	0.290
Gemma 4 E2B	0.186	0.740	0.050	0.050	0.000	0.142	0.150

Table 4: Macro F1 per relation for all models evaluated with the Initial Closed-Book approach on the training set.

Model	Overall	Borders	CityOfDeath	Capacity	Award	StockExch.	Area
Mistral Small 3.2 24B	0.520	**0.989**	0.310	0.210	0.303	0.610	**0.660**
Granite 4.1 30B	0.497	0.987	0.550	0.110	0.383	0.469	0.540
Gemma 4 31B	0.438	0.980	0.280	0.140	0.325	**0.620**	0.360
Qwen 3.6 27B	0.389	0.947	0.240	0.210	0.275	0.263	0.480
Granite 4.1 8B	0.362	0.925	0.220	0.040	0.408	0.407	0.400
Ministral 3 14B	0.341	0.912	0.210	0.140	0.411	0.063	0.560
Gemma 4 26B	0.383	0.960	0.160	0.190	0.343	0.390	0.410
Ministral 3 8B	0.299	0.938	0.210	0.060	0.000	0.107	0.420
Gemma 4 12B	**0.529**	0.974	0.530	**0.490**	0.500	0.420	0.380
LFM2 24B	0.268	0.789	0.190	0.060	0.280	0.133	0.340
GPT-OSS 20B	0.288	0.750	0.210	0.070	0.400	0.232	0.320
Granite 4.1 3B	0.289	0.952	0.180	0.050	0.250	0.265	0.220
Llama 3.2 3B	0.269	0.870	0.160	0.070	0.000	0.141	0.330
Ministral 3 3B	0.242	0.869	0.230	0.050	0.000	0.070	0.220
Gemma 4 E4B	0.316	0.898	**0.610**	0.040	0.050	0.112	0.140
DeepSeek R1 14B	0.520	0.949	0.500	0.370	**0.600**	0.327	0.590
Phi 4 Mini 3.8B	0.229	0.762	0.170	0.050	0.000	0.072	0.290
Gemma 4 E2B	0.195	0.720	0.110	0.050	0.000	0.137	0.150

Table 5: Macro precision per relation for all models evaluated with the Initial Closed-Book approach on the training set.

Model	Overall	Borders	CityOfDeath	Capacity	Award	StockExch.	Area
Mistral Small 3.2 24B	**0.622**	0.956	**0.690**	**0.210**	0.021	**0.765**	**0.660**
Granite 4.1 30B	0.526	0.925	0.600	0.110	**0.040**	0.635	0.540
Gemma 4 31B	0.462	**0.964**	0.570	0.080	0.014	0.555	0.350
Qwen 3.6 27B	0.451	0.922	0.570	0.070	0.005	0.415	0.480
Granite 4.1 8B	0.443	0.869	0.610	0.040	0.009	0.480	0.400
Ministral 3 14B	0.470	0.832	0.570	0.140	0.005	0.415	0.560
Gemma 4 26B	0.437	0.931	0.510	0.100	0.020	0.468	0.380
Ministral 3 8B	0.439	0.853	0.590	0.060	0.000	0.452	0.420
Gemma 4 12B	0.364	0.881	0.440	0.030	0.001	0.435	0.240
LFM2 24B	0.441	0.928	0.580	0.060	0.007	0.502	0.340
GPT-OSS 20B	0.395	0.703	0.580	0.070	0.000	0.442	0.320
Granite 4.1 3B	0.388	0.789	0.580	0.050	0.001	0.472	0.220
Llama 3.2 3B	0.384	0.742	0.450	0.070	0.000	0.485	0.330
Ministral 3 3B	0.377	0.832	0.610	0.050	0.000	0.360	0.220
Gemma 4 E4B	0.347	0.780	0.450	0.040	0.002	0.503	0.140
DeepSeek R1 14B	0.321	0.605	0.470	0.010	0.004	0.443	0.200
Phi 4 Mini 3.8B	0.356	0.624	0.530	0.050	0.000	0.410	0.290
Gemma 4 E2B	0.369	0.919	0.430	0.050	0.000	0.513	0.150

Table 6: Macro recall per relation for all models evaluated with the Initial Closed-Book approach on the training set.

Exemplars Primarily Shift Retrieval, Verification Instructions Can Shift Rejection: An Anatomy of Inference-Time Interventions for Closed-Book Knowledge Base Construction

Takashi Sasaki
Acroquest Technology Co., Ltd.
sasaki@acroquest.co.jp

Abstract

We describe our AKBC 2026 closed-book knowledge base construction system: Gemma-3-27B-pt, raw text completion, 20 candidates per query, and relation-specific aggregation without an arbiter. It uses 27.4B parameters and scores 0.6670 row-averaged Macro F1 on the official test, placing sixth. Experiments distinguish this final pipeline from an earlier Mistral system with an 8B arbiter. Backbone and access format outweighed prompt-level interventions: under matched prompt formats, the pretrained checkpoint exceeds its instruction-tuned sibling by 0.11 on validation. Numeric example configurations primarily shift retrieval; procedural wording with five added JSON examples changes the selected answers by only one row. Verification instructions can shift rejection. A biography-first city prompt format affects both, raising validation F1 from 0.55 to 0.61 at a fixed vote gate. Gating, information outside the model, and seed stability help interpret transfer. These findings have counterexamples; leaderboard comparisons describe development outcomes, not an untouched final evaluation.

1 Introduction

LM-KBC (Singhania et al., 2023; Kalo et al., 2024) asks a model to output *all* object values for a subject–relation pair, including empty or large sets, from its own weights. We call producing a stored value *retrieval* and discarding a generated candidate *rejection*.

We developed Qwen, Mistral, and Gemma systems (Sec. 4), varying models, prompts, sampling, and aggregation. Section 3 consolidates the final system; controlled experiments then explain the development choices. Experiments use validation and, for threshold selection, train. Hidden-test scores track submissions, with later choices partly informed by leaderboard feedback (Sec. 4.2).

2 Task and Constraints

The 2026 task covers six relations over 475 validation and 475 test subject–relation pairs: two set-valued relations that may be empty (`countryLandBordersCountry`, `companyTradesAtStockExchange`), one single-valued string relation that may be empty (`personHasCityOfDeath`), one large-set relation (`awardWonBy`), and two numeric relations (`hasArea`, `hasCapacity`) scored at 5% relative tolerance. Gold values come from Wikidata (areas `P2046`, capacities `P1083`), with alias lists accepted for strings. We report the official "All Relations" score throughout: a Macro F1 averaged over *rows*, not relations, so the four 100-row relations dominate it. External search, corpora, and fine-tuning are prohibited; inference is capped at 32B parameters.

3 Final Submitted System (V19)

Figure 1 consolidates V19 and its experimental basis. It uses one Gemma-3-27B-pt checkpoint (Q8_0, 27.4B) with `llama.cpp` at a pinned revision. Text completion continues a record- or infobox-style prefix without a chat template. Numeric prompts use four fixed training examples; city uses three hand-written examples, then generates `About:` before `City of death:`. The figure specifies sampling and every relation's aggregation and empty-output rules.

Normalization. Released train/validation aliases are grouped by English-label matches before string voting. Company additionally folds case variants and acronyms into co-occurring long forms (NYSE → New York Stock Exchange). Numeric predictions use `dominant_cluster`. All frequencies include empty candidates.

Scope. Figure 1 separates deployed components from development-only experiments. V19's city gate is 14 votes; Sec. 5.4's fixed-12-vote compar-

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 473–479
October 28, 2026 ©2026 Association for Computational Linguistics

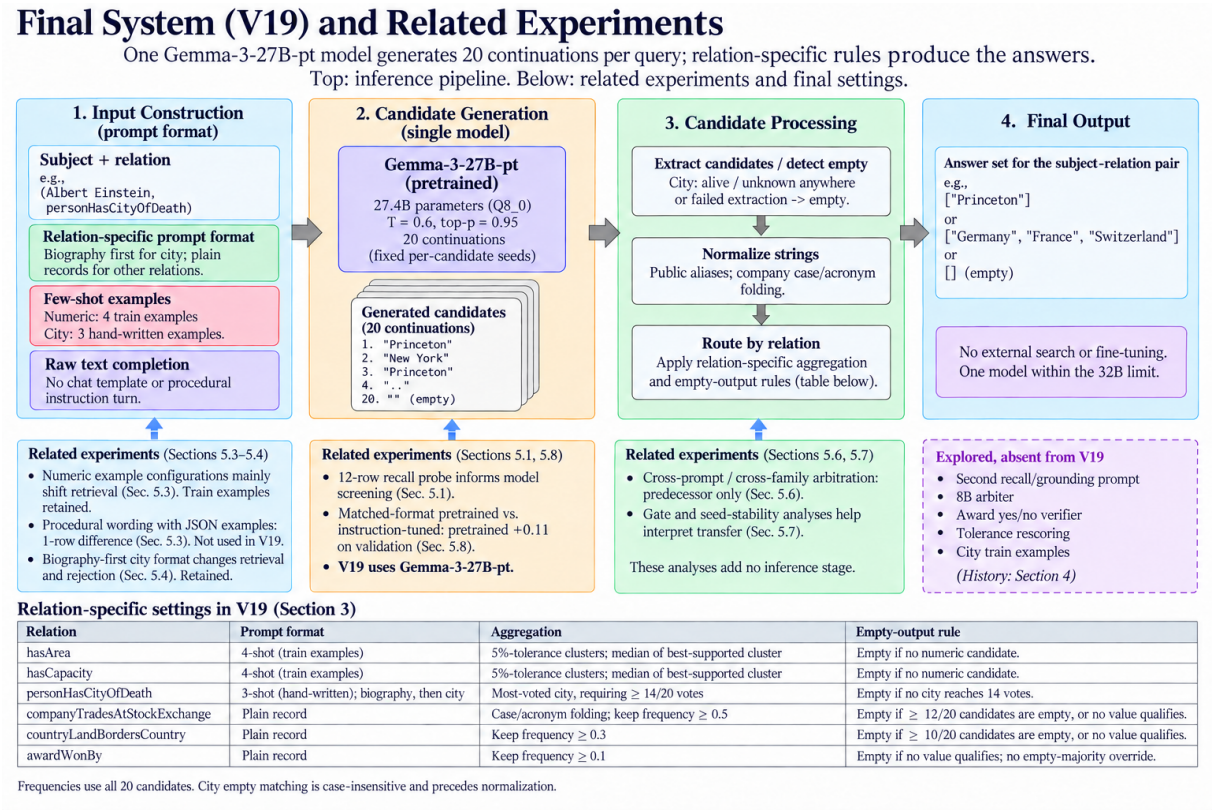


Figure 1: Final V19 system and related experiments. Gray arrows show inference; blue arrows associate experiments with components. All frequencies use the 20-candidate pool. City empty detection precedes normalization and voting. The final answer list is written as `ObjectEntities`, alongside `SubjectEntity` and `Relation`, in official JSONL format.

ison isolates the prompt change with the same candidate-extraction rule.

Reproducibility. Team: `acro-sasaki`. Code, pinned model, container and dataset versions, experiment records, and `docs/V19_REPRODUCE.md`: https://github.com/acrosasaki-rgb/llm-knowledge-task-2026. Builders deterministically reproduce predictions from saved candidates (SHA-256 verified).

4 Development History and Results

4.1 Predecessor system

Most experiments use Mistral Small 3.2 24B with 20 candidates from each of two prompts: **recall**, with two verbalized-recall exemplars; and **grounding**, with five bare-JSON examples naming Wikidata `P2046`/`P1083` (Sec. 5.3). For numeric disagreements above 5%, the mode of 20 Llama-3.1-8B samples selects the secondary answer only when matching it but not the primary (Sec. 5.6). This 31.8B predecessor cannot introduce a third value; an incorrect switch can lose a row.

4.2 Results

Table 1 separates the model and pipeline progression from the dataset re-keying, which corrects what is scored rather than improving the system. The public alias graph improved validation 0.658 → 0.663, driven by company. Section 3 and Fig. 1 define the final configuration. Per-relation test scores for V19 are: borders 0.960, company 0.832, area 0.820, city 0.530, award 0.336, capacity 0.316. The city improvement from v18 to v19 (0.510 → 0.530) combines a prompt format change with a gate change; the fixed-gate prompt format comparison is reported in Sec. 5.4. Some revisions were retained or reverted after leaderboard feedback, notably the removal of city training exemplars. Submission deltas therefore describe development outcomes rather than an untouched final evaluation.

5 Anatomy of Inference-Time Interventions

Controlled comparisons below isolate the stated intervention; submission transitions in Table 1

System	test
Qwen3.5-27B thinking	0.571
Mistral 3.2 grounding	0.595
v3: + recall prompt, area arbitration, merge	0.604
v4: + capacity arbitration	0.606
v5: + acronym merging	0.610
v7: + city merge	0.608
v8: re-keying*	0.625
v10: + tolerance rescoring	0.623
v11: − rescoring	0.625
v12: Gemma-3-27B-pt	0.649
v13: + alias graph	0.648
v16: + train exemplars	0.659
v18: + city vote gate	0.663
v19: + biography, city gate 12→14	**0.667**

Table 1: Official test scores of our submissions (row-averaged Macro F1, three decimals). Rows summarize development changes, not uniformly isolated ablations: v19 changes both the city prompt format and vote gate. Validation scores are not comparable across the whole table—the August release re-keyed the gold—so they are reported per experiment in Sec. 5 instead. The official baseline scores 0.313 on validation. *Following that release, which renamed 15 test subjects and dropped two rows (477→475); v3–v7 score the renamed rows as zero. Versions v9, v14–v15, and v17 are omitted for space.

need not. Numeric testbeds are `hasArea` and `hasCapacity` (100 validation rows × 20 candidates per prompt). **Oracle** is the number of rows whose candidate set contains at least one within-tolerance value; the **mode** row count is what `dominant_cluster` selects. The **anchor** is the first concrete value for the queried quantity in a candidate.

5.1 Backbone Recall Reliability Dominates

Our Qwen-era system plateaued after nine intervention experiments (prompts, temperature, output routes, thinking budgets, judge models, log-probability confidence) whose combined deployable gain was under $+0.01$. A 12-row probe of its wrong rows across candidate models (Table 2) showed Mistral Small 3.2 answering 11 of the 12 somewhere in its candidate set; regenerating the pipeline on Mistral with nothing else changed moved test 0.571→0.595.

The second change was larger and reversed an earlier decision. Gemma 4 31B Instruct, queried through its chat template, had the weakest `hasArea` oracle we measured (43/100) and we closed that lineage early. Probing the *pretrained* Gemma-3-27B checkpoint in a completion format instead gave the best profile of any configuration we tested—oracle 94, mode 84 on `hasArea`, borders 0.986—and a system built on it took test $0.625 \rightarrow 0.649$. It also supplies values no prompt reached on Mistral: secondary stock listings that five Mistral prompt variants returned with zero votes (ČEZ on Warsaw, Questerre on Oslo, Aisin on Nagoya) appear with 7–13 votes out of 20. This mattered more than anything else we varied: closed-book quality is set by which label-to-value pairings pretraining left intact and by whether the prompting format can reach them.

Model (12-row probe)	any	mode
Mistral Small 3.2 Instruct	11	8
Mistral Small 3.1 Base	10	7
Mistral Small 3.1 Instruct	9	4
Llama-3.1-8B Instruct	7	4
Aya Expanse 32B	3	1
Qwen3-8B	1	0

Table 2: The recall probe: 12 `hasArea` rows stratifying the Qwen-era system's failures, 20 candidates each; *any* = rows with a correct candidate, *mode* = rows whose modal cluster is correct. Qwen3.5/3.6-27B, Gemma 4 31B, GLM-Z1-32B and llm-jp-4-32b (not shown) sit between Llama-8B and Aya. It costs ∼30 GPU-minutes per model.

5.2 Answers Commit Early

In persisted reasoning traces of the Qwen-era system, the answer value first appears at a median 17% of the trace, in 95–98% of candidates inside a retrieval template ("Commonly cited: X"); the remaining 80% of the trace never dislodges it. The verification text that follows is consistent with post-hoc justification rather than independent checking: fabricated Wikidata Q-IDs and cross-language "confirmations" agree with whatever the anchor was, and wrong candidates carry *more* of this apparatus than correct ones (6.9 vs. 4.2 source mentions). Early commitment replicates on Gemma 4 31B and Qwen3.6; the confabulated verification does not. Extended thinking is thus an *exploration* aid: it raises the oracle where recall is unreliable (Qwen $+9$ rows, Gemma $+7$) and has no measurable effect on Mistral or Magistral.

5.3 Exemplar Style Primarily Shifts Retrieval

Table 3 distinguishes procedural wording from the examples embedded in that wording. With five added bare-JSON examples, the two configurations differ by only one mode-correct row (71 vs. 72).

Instruction	In-text recall	Added JSON	Mode	Oracle
Grounding	0	5	71	89
Grounding	0	0	41	59
Recall-format	2	5	72	83
Recall-format	2	0	78	84

Table 3: Mistral 3.2, 100 validation `hasArea` rows, 20 candidates each. Example columns show counts and placement. These are four prompt configurations, not a factorial isolation of example style: count, content, and placement also differ. The last row uses the original recall format; an independently relabeled variant also reaches mode 78, with oracle 89.

Removing those examples from the recall-format prompt retains two verbalized-recall examples and raises the mode from 72 to 78; removing all examples from grounding instead lowers it to 41. The recall-format candidates verbalize recall in 99% of generations without the added JSON examples, but in 0% with them. Thus example configuration strongly affects generated answers here; this comparison does not isolate style from example count or content, or establish that instructions generally have no effect. The independently relabeled recall format retains mode 78.

In the final backbone, replacing our hand-written exemplars with four answers taken verbatim from the *training* split—same prompt format, same generation, same aggregation, only the examples change—moves validation `hasArea` $0.84 \rightarrow 0.88$ and `hasCapacity` $0.309 \rightarrow 0.361$. The gain holds across three sampling seeds (area 0.88/0.89/0.87, capacity 0.361/0.351/0.361), and no exemplar subject occurs in validation or test.

These comparisons locate the clearest exemplar effect upstream in candidate generation; they do not establish that exemplars have no downstream effect on abstention or rejection.

5.4 Verification Instructions Can Shift Rejection

Some verification instructions acted downstream on the *discarding* of already-generated candidates. Verifying each `awardWonBy` candidate with three yes/no votes and dropping only unanimous-no candidates raised precision $0.241 \rightarrow 0.280$ and F1 $0.172 \rightarrow 0.186$; the yes votes carried no information, so the usable signal was one-sided; this verifier is not retained in V19 (Sec. 3). A digit-check step inside an otherwise inert step-by-step procedure likewise cut votes for order-of-magnitude twin

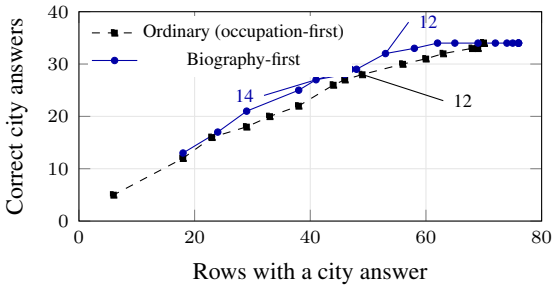


Figure 2: City coverage and correctness on 100 validation rows, 20 candidates each. Points sweep integer vote gates 1–20 on the saved pools; lines only connect observed points. Labels mark the fixed 12-vote comparison and V19's 14-vote biography gate. Correct empty outputs are excluded from the vertical axis.

values from 38 to 32 with the oracle unchanged.

For `personHasCityOfDeath`, generating a one-line biography before the city field raises validation F1 from 0.55 to 0.61 at a fixed 12-of-20 gate; the change also helps on train (0.52 to 0.55) and a second validation seed (0.60). The parser in Sec. 3 turns generated living-person cues into empty candidates. The result therefore reflects the prompt format together with this fixed extraction rule, not an additional neural verifier.

Figure 2 recomputes the vote-gate sweep from saved validation pools. At 12 votes, correct city fills rise from 28 to 32 and correct empty outputs from 27 to 29, accounting for the six-point F1 gain. At matched coverage, biography also retains more correct cities: 25 vs. 22 at 38 filled rows, and 28 vs. 27 at 46. This supports effects on both retrieval and rejection, rather than merely a lower answer rate. The final 14-vote setting is marked separately; the figure is a validation analysis, not a test-set threshold search. Applying biography to area, capacity, and company gives no gain ($0 \ldots -2$ rows); fifteen further city gates improve validation by at most two rows.

5.5 The Exemplar Effect Is Conditional

Verbalized recall is a *focusing* intervention: it sharpens the anchor distribution around the most frequently recalled value (oracle $89 \rightarrow 84$ as the mode improves), so its sign follows how reliable recall already is, the opposite way round from thinking. Thinking explores, gaining $+9$ rows where recall is unreliable (Qwen) and nothing where it is reliable (Mistral); verbalized recall does the opposite, zero or negative on Qwen and on Mistral's weakest relation (`hasCapacity` oracle $60 \rightarrow 47$) but $+6$

Component	Δ val	Δ test	V19
recall exemplars (§5.3)	+6	+5	No
company folding (§3)	+1	+2	Yes
8B arbitration (§5.6)	+2	+1	No
train exemplars: area, capacity	+8	+5	Yes
capacity prompt (best of 6)	+2	−2	No
second arbiter (increment)	+1	0	No
tolerance rescoring	+2	−1	No
train exemplars: city	+4	−5	No
city two-stage merge	+11	−1	No

Table 4: Row-level validation and test deltas for interventions deployed in at least one submitted system, measured in their original experiment settings. V19 marks retention in the final pipeline, not the system on which the delta was measured: recall exemplars, company folding, and 8B arbitration here were measured on Mistral. Company folding is distinct from the later public alias graph. All final relation-specific choices are consolidated in Sec. 3 and Fig. 1.

on `hasArea`. In Magistral (Mistral's RL-tuned sibling) it disappears: the verbalized form is copied at 99.9% compliance in three different channels, yet the anchor distribution does not move in any of them (70–72), suggesting that RL post-training may decouple response style from retrieval.

5.6 We Find No Usable Within-Model Selection Signal in the Features We Tested

Within one model's candidate set, minority correct values were indistinguishable from minority wrong ones on every feature we measured (votes, dispersion, trace features, log-probabilities; base rate 8%), which is why judges, confidence weighting, and consensus schemes failed in our experiments. What did extract signal is *disagreement between prompts of different style*: on the ∼25 rows where the recall and grounding prompts disagree by more than 5%, one aggregated decision from an unrelated-family 8B model resolves the tie, giving `hasArea` 78 → 80 and `hasCapacity` 26 → 28 on validation; of the four `hasArea` rows it changed, three were corrected and one harmed. Controls locate the conditions: same-style prompt pairs produce too little disagreement to arbitrate; promoting minority clusters by the arbiter's mode is *negative*, since an 8B cannot overrule a 24B mode, only break ties; and a weak arbiter (Ministral-8B, mode accuracy 23 vs. Llama's 53) gains nothing. The isolated v3→v4 ablation confirms the transfer on test: capacity F1 0.204 → 0.214.

5.7 Which Validation Gains Transfer

Our submissions give per-component test measurements (Table 4). The components that transferred share two properties: each is *gated*, touching only rows that meet an explicit condition, and each gate uses information from outside the model—an unrelated-family vote, or the gold alias structure that makes two surface forms one count. The exemplar effect is the apparent exception: it changes what the backbone retrieves rather than adjudicating between candidates.

Non-transferring components include the best of six capacity prompts, a second arbiter, and a city merge that gained 11 validation rows (Table 4). Tolerance rescoring is especially instructive: values with ratio below $1.05/0.95 \approx 1.105$ admit one prediction correct for either. Shifting predictions to cover a runner-up cluster gained 1–2 validation rows across prompt pairs and arbiter seeds, but lost one test area row (0.800 → 0.790); we removed it in v11. Unlike the retained selection gates, this adjustment was applied globally. Later gates based on the same model's unit agreement or answer-blind priming also admitted confident errors. A third predictor appeared late. When we replaced our exemplars with training answers, the same change helped `hasArea` and `hasCapacity` on all three seeds we ran and helped `personHasCityOfDeath` on one seed only (0.56/0.50/0.49); we deployed all three, and on test the two seed-stable relations transferred (+4 capacity rows) while the seed-unstable one cost five rows. We now treat single-seed differences below about four rows on a 100-row relation as unusable. A 100-row validation set may not reliably expose these failures; in our experiments, whether a component is gated, whether its gate looks outside the model, and whether its gain survives reseeding predicted transfer better than effect size or the presence of a mechanism-level rationale.

5.8 Residual Errors Look Like Values Learned From the Corpus

The residual errors we examined are not reasoning failures. Corfu (gold 626 km^2) is answered as 592.9 by every Mistral-family instruct configuration, 593 by Qwen3.5, and 580 by Gemma: stable, model-specific wrong values that survive every prompt. Three observations support the hypothesis that they were learned from training data. First, *turnover across generations*: Qwen3.6 repairs 7 of Qwen3.5's wrong rows and breaks 11

new ones with nothing else changed, and the union oracle reaches 89/100. Second, *unrelated training lineages solve mainstream-broken rows*: llm-jp-4-32b and Aya Expanse are far weaker overall yet each partially solves Corfu, which no mainstream-corpus model solves. Third, *instruction tuning costs recall*: under raw completion Mistral 3.1 Base scores 10/12-any against its own instruct's 9, and the gap replicates on Llama-3.1-8B (mode accuracy 60 vs. 53 of 100).

The last comparison is the one our final system rests on, so we ran it without the prompt format confound: the instruction-tuned sibling of our backbone, queried in the *same* completion prompt formats with the same sampling and aggregation, scores 0.553 on validation against 0.665 for the pretrained checkpoint (`hasArea` 0.61 vs. 0.84, `hasCapacity` 0.24 vs. 0.31, `personHasCityOfDeath` 0.38 vs. 0.52), on two seeds. Since weights are the only difference, the loss belongs to instruction tuning rather than to the chat template, at least in this lineage—and the abstention that makes the biography prompt format work (Sec. 5.4) is absent in the tuned sibling, which fabricates instead. The ceiling is a property of the checkpoint, not of prompt engineering within one setting.

6 Related Work

Probing factual knowledge in LMs goes back to LAMA (Petroni et al., 2019); the LM-KBC series (Singhania et al., 2023; Kalo et al., 2024) extends it to KB construction with unconstrained cardinalities. Self-consistency (Wang et al., 2023) aggregates sampled answers by majority vote and ReWiSe (Albert-Roulhac and Zouaq, 2025) tuned it relation-wise; we extend it to empty answers, numeric tolerance, surface folding, and cross-family arbitration. Sprague et al. (2025) find chain-of-thought helps reasoning but not knowledge recall; in our numeric comparisons, example configuration shifts generated answers, while the wording comparison with added JSON examples differs by only one row. A chain-of-thought prompt format helps city of death, where a living-person cue supports abstention as well as correct city recall. Ma and Hewitt (2026) report prompting moves parametric knowledge access by 1–2% and RL fine-tuning by ∼10%; our exemplar effect sits between, and fine-tuning is prohibited here. Yuchi et al. (2026) find probe-recoverable numeric knowledge exceeds what models verbalize, one reading of our oracle–mode gaps, and Kalyan et al. (2021) and Park et al. (2022) document the magnitude and unit fragilities our components target.

7 Conclusion

V19 (Sec. 3) scored 0.6670, placing sixth. Backbone choice outweighed prompt interventions. Example configurations primarily shifted retrieval; verification affected rejection, and biography affected both. Gating, outside information, and seed stability helped interpret transfer within this task edition.

Limitations

Mechanism findings rely mainly on Qwen and Mistral; the final backbone is least ablated. Only exemplar, arbitration, and prompt format results have seed replications. The 12-row probe samples one system's failures and was not tested on held-out models. Counts on 100-row relations vary by 1–2 rows. Only Sec. 5.8's Gemma control fixes the prompt format; earlier Mistral/Llama comparisons do not. Transfer observations cover one task edition with repeated leaderboard feedback.

References

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-wise self-consistency for LLM probing. In *Joint Proceedings of KBC-LM and LM-KBC @ ISWC 2025*, volume 4041 of *CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2024. Preface: LM-KBC challenge 2024. In *Joint Proceedings of the KBC-LM Workshop and the LM-KBC Challenge @ ISWC 2024*, volume 3853 of *CEUR Workshop Proceedings*, Baltimore, USA.

Ashwin Kalyan, Abhinav Kumar, Arjun Chandrasekaran, Ashish Sabharwal, and Peter Clark. 2021. How much coffee was consumed during EMNLP 2019? Fermi problems: A new reasoning challenge for AI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Melody Ma and John Hewitt. 2026. Improving parametric knowledge access in reasoning language models. ArXiv:2602.22193.

Sungjin Park, Seungwoo Ryu, and Edward Choi. 2022. Do language models understand measurements? In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP 2019*, pages 2463–2473.

Sneha Singhania, Jan-Christoph Kalo, Simon Razniewski, and Jeff Z. Pan. 2023. Preface: LM-KBC challenge 2023. In *Joint Proceedings of the KBC-LM Workshop and the LM-KBC Challenge @ ISWC 2023*, volume 3577 of *CEUR Workshop Proceedings*, Athens, Greece.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2025. To CoT or not to CoT? chain-of-thought helps mainly on math and symbolic reasoning. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Fengting Yuchi, Li Du, and Jason Eisner. 2026. LLMs know more about numbers than they can say. In *Proceedings of EACL 2026 (Short Papers)*. ArXiv:2602.07812.

COVER-KBC: Evidence-Centric Adaptive Inference for Closed-Book Knowledge Base Construction

Quoc-Linh Vo
Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam
vqlinh2433@apcs.fitus.edu.vn

Abstract

Large Language Models (LLMs) encode substantial factual knowledge, but reliably extracting complete knowledge-base entries remains challenging when relations differ in cardinality, semantics, and missingness. We present COVER-KBC, our system for the AKBC Shared Task 2026, which formulates closed-book knowledge extraction as relation-typed, evidence-centric inference. Each relation is compiled into an inference program that controls elicitation, verification, stopping, and final selection. Model observations are accumulated in a provenance-aware evidence state and scored using independent support, calibrated verification, contradiction, and disagreement. To distinguish candidate correctness from answer-space completeness, COVER-KBC estimates residual search need and uses a budget-aware controller to adaptively acquire further evidence when necessary. Relation-specific finalization handles heterogeneous output structures and difficult numeric or nullable cases. Using a single Mistral-Small-3.2-24B checkpoint without additional training or external retrieval, COVER-KBC achieves a Macro-F1 of 0.5878 on the hidden test set. Our implementation is available at https://github.com/vquclinh/cover-kbc.

1 Introduction

Pretrained language models encode substantial relational knowledge in their parameters, making them attractive sources for closed-book knowledge base construction (KBC) (Petroni et al., 2019). Yet factual knowledge that can be elicited from a model is not always stable across prompts (Cao et al., 2021), and constructing a knowledge-base entry is harder than recalling a single fact. A system must recover the appropriate object set, reject unsupported candidates, and determine whether valid objects may still be missing. This becomes especially challenging when relations range from nullable and numerical attributes to small closed sets and large open-ended sets.

We argue that heterogeneous closed-book KBC is therefore not merely a generation problem, but an active evidence-acquisition problem. The objective is to recover an object set that is both precise and sufficiently complete under a bounded inference budget. Importantly, confidence that an individual candidate is correct does not imply that the answer set is complete (Razniewski et al., 2024); deciding whether to trust a candidate and whether to continue searching are distinct inference problems.

We propose COVER-KBC, a relation-typed, evidence-centric framework built around this distinction. First, each relation is compiled into an executable inference contract that governs elicitation, verification, stopping, and final selection. Second, model observations are accumulated as independence-aware atomic evidence, preserving provenance, support, contradiction, and verification signals rather than collapsing repeated generations into a flat vote. Third, COVER-KBC separately models candidate confidence and residual search need, allowing a coverage-aware controller to decide whether to acquire further evidence, verify unresolved candidates, or stop. Thus, both *what to ask next* and *when enough evidence has been collected* become relation-dependent decisions.

AKBC Shared Task 2026. This shared task (Kalo et al., 2026) evaluates closed-book object-set prediction. For each query $q = (s, r)$, where s is a subject entity and r is a relation, the system must predict an object set $\hat{O}_{s,r}$ using only the parametric knowledge of a language model. The benchmark covers six heterogeneous relations: land borders, city of death, venue capacity, award recipients, stock exchanges, and geographic area [1]. Predictions may be empty, singleton, or multi-valued; external factual retrieval and additional model train-

[1] https://www.akbc.ws/2026/shared-task.html

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 480–487
October 28, 2026 ©2026 Association for Computational Linguistics

ing are prohibited, while multi-step inference is allowed under a 32B neural-parameter budget. Using a single Mistral-Small-3.2-24B checkpoint without additional training or external retrieval, COVER-KBC achieves 0.5878 Macro-F1 on the hidden test set.

2 Related Work

2.1 Closed-Book Knowledge Construction

Prior LM-KBC systems use multi-step or relation-specific prompting, relation-wise self-consistency, and relation-dependent decomposition (Alivanistos et al., 2022; Das et al., 2024; Albert-Roulhac and Zouaq, 2025; He and Razniewski, 2025). Multi-stage language-model programs similarly organize multiple model calls into structured inference procedures (Opsahl-Ong et al., 2024). COVER-KBC instead represents each relation as an executable contract governing elicitation, verification, stopping, and finalization.

2.2 Evidence, Verification, and Calibration

Self-consistency aggregates repeated generations (Wang et al., 2023), while Atomic Self-Consistency combines reliable subcomponents across samples (Thirukovalluru et al., 2024). Knowledge-fusion methods preserve source-specific evidence when estimating factual reliability (Dong et al., 2014). Chain-of-Verification separates verification from initial generation (Dhuliawala et al., 2024), and contextual calibration reduces systematic label bias (Zhao et al., 2021). COVER-KBC combines these ideas in an independence-aware evidence state with candidate-level blind verification.

2.3 Adaptive and Coverage-Guided Inference

Adaptive-Consistency stops sampling when additional generations are unlikely to change the decision (Aggarwal et al., 2023), while FLARE and Self-RAG condition further information acquisition on uncertainty or self-reflection (Jiang et al., 2023; Asai et al., 2024). Knowledge-base completeness work distinguishes correctness of observed facts from whether relevant facts remain missing (Razniewski et al., 2024). COVER-KBC operationalizes these ideas in a closed-book setting by separating candidate confidence from residual search need and using relation-specific evidence to determine further acquisition or stopping.

Relation	Type
`countryLandBordersCountry`	SMALLSET
`companyTradesAtStockExchange`	SMALLSET
`personHasCityOfDeath`	NULLSINGLE
`hasArea`	NUMERIC
`hasCapacity`	NUMERIC
`awardWonBy`	LARGEOPENSET

Table 1: The six relations are compiled into typed inference programs.

3 Methodology

We formulate closed-book KBC as relation-typed active evidence acquisition. Given the task setting above, COVER-KBC does not map a prompt response directly to a final object set. Instead, each model call is parsed as an observation and accumulated into an explicit inference state containing candidate objects, evidence provenance, verification signals, residual search need, and remaining budget. The system has four components: relation-typed inference contracts, independence-aware evidence aggregation, blind candidate verification, and budget-aware control with relation-specific finalization.

3.1 Relation-Typed Inference Programs

We define each relation through an executable inference contract

$$C_r = (\tau_r, \mathcal{V}_r, \mathcal{P}_r, \mathcal{G}_r, F_r),$$

where τ_r is the relation type, $\mathcal{V}_r$ is the set of eligible evidence-acquisition views, $\mathcal{P}_r$ defines parsing and normalization, $\mathcal{G}_r$ defines verification and stopping behavior, and F_r is the final selector.

This contract is our mechanism for relation conditioning: it specifies the semantic definition of the relation, positive cases, hard negatives, cardinality, mandatory views, optional views, verifier policy, stopping policy, and output selector. Relation conditioning therefore affects the program executed by the system, not only the wording of the prompt. Table 1 shows how the six task relations are mapped to COVER-KBC program types.

The four program types correspond to different inference problems. SMALLSET relations require high-precision set closure. NULLSINGLE relations require distinguishing true empty answers from failed recall. NUMERIC relations require parsing and clustering scalar values. LARGEOPENSET re-

lations require continued discovery while avoiding unbounded tail expansion.

3.2 Independence-Aware Evidence State

Each model call produces a parsed observation. COVER-KBC stores these observations in an evidence state

$$S_t = (O_t, R_t, B_t),$$

where O_t is the current set of candidate objects, R_t is the set of evidence records, and B_t stores budget, gate, and controller state. Each record attaches to one candidate object and carries the source view, prompt provenance, parser result, evidence type, and an independence group.

We do not count repeated samples from the same view as independent corroboration. Evidence from different acquisition mechanisms, such as direct recall, structural decomposition, contrastive questioning, missingness search, or reverse checking, can provide separate support. This separates repeated wording effects from genuinely distinct evidence.

For a candidate object o, let $m(o)$ be the set of acquisition groups that could express o under the relation contract, and let $g(o) \subseteq m(o)$ be the groups that support it. We define the independent support of candidate o as

$$F(o) = \frac{|g(o)|}{|m(o)|}.$$

Verifier agreement is represented separately because verifier calls are shown the candidate.

3.3 Blind Verification and Candidate Confidence

COVER-KBC combines the evidence attached to a candidate into a single score:

$$S(o) = \alpha F(o) + \beta L(o) + \gamma X(o) - \delta C(o) - \eta U(o),$$

where $F(o)$ is the independent acquisition support of Section 3.2, $L(o)$ is the calibrated verifier signal defined below, and $X(o)$ is optional heterogeneous recall from a second model family. Because our submitted system uses one Mistral checkpoint for every neural role, $X(o)$ is inactive throughout the results we report.

The two penalties capture evidence against a candidate rather than absence of evidence for it. $C(o)$ is the normalized strength of explicit contradiction, where the denominator covers the acquisition mechanisms together with the blind verifier; a view that simply does not mention the candidate, an optional view that was never run, and an UNKNOWN verifier label are all excluded, since none of them is a signed judgement against o. $U(o)$ is the normalized disagreement among verifier label distributions across the prompt templates of a relation, so a candidate whose verifier judgement changes with wording is discounted. All five weights are constants of the released configuration; their values are listed in Appendix A.

Verification is deliberately blind. The verifier receives the subject, relation definition, hard-negative rules, and one candidate; it sees neither the generator prompt and rationale nor the support count or competing candidates, so its judgement cannot inherit the generator's confidence. It predicts one of three labels: VALID, INVALID, or UNKNOWN. We calibrate the verifier logits against a content-free baseline:

$$\tilde{p}_j = \frac{\exp((z_j - b_j)/T)}{\sum_k \exp((z_k - b_k)/T)},$$

where z_j is the raw label logit, b_j is the content-free baseline logit for the same template, and T is the temperature. The verifier score is the log-odds of VALID against the remaining mass, rescaled to a bounded signal:

$$L(o) = \mathrm{clip}_{[-1,1]}\left(\frac{\log \frac{\tilde{p}_{\text{VALID}}+\epsilon}{\tilde{p}_{\text{INVALID}}+\tilde{p}_{\text{UNKNOWN}}+\epsilon}}{3.0}\right),$$

with ϵ a small constant for numerical stability. Placing UNKNOWN in the denominator makes abstention count against a candidate rather than leaving it neutral: under closed-book inference an unverifiable candidate is one we prefer not to emit, and the conservative treatment is what keeps precision high on the small-set relations.

3.4 Residual Search Need and Adaptive Control

Candidate confidence estimates whether an observed candidate should be trusted. Completeness is handled separately through a residual search signal q_{res}, which estimates whether another action is likely to add useful verified information.

The signal aggregates three groups of relation-dependent gap indicators:

$$q_{\text{res}} = \mathrm{clip}_{[0,1]}(r_t),$$
$$r_t = \max\big(\bar{\phi}_w,\ \phi_{\text{mandatory}},\ \phi_{\text{blocking}}\big),$$

where $\bar{\phi}_w$ is the weighted mean of the soft gap signals admitted by the relation contract, $\phi_{\text{mandatory}}$ is one whenever a mandatory view has not yet been executed, and ϕ_{blocking} is a program-typed blocking condition. We take the maximum rather than a weighted sum so that a single unsatisfied mandatory view or blocking gap forces continuation regardless of how settled the remaining signals are.

The blocking term is used only where a hard gap is not already captured by mandatory views. SMALLSET and LARGEOPENSET relations set $\phi_{\text{blocking}} = 0$. NUMERIC relations set it to the normalized support of the strongest rival numeric cluster relative to the dominant one, so a competitive second cluster keeps the relation open. NULLSINGLE relations set it to the maximum of an unresolved existence gate and locality competition between the top two candidates, except that a confidently negative existence gate sets it to zero: the empty set is a valid answer, not a failure to recall.

The relation type also determines which soft signals enter $\bar{\phi}_w$: SMALLSET contracts use set instability and unresolved mass, NUMERIC contracts use numeric dispersion, and LARGEOPENSET contracts use marginal yield and facet coverage. NULLSINGLE contracts rely on the blocking term instead, since existence and locality already determine whether the relation may stop.

The controller stops when any of a fixed sequence of conditions holds: the call or token budget is exhausted, every mandatory view has run and the typed program declares the relation closed, no verification is still pending, and finally q_{res} has fallen below the relation's stopping threshold. Acquisition is bounded in any case by a maximum number of steps per query. The threshold is part of the contract's stopping policy; the component weights inside $\bar{\phi}_w$ are shared configuration constants. The scalar and its components play different roles: q_{res} is the last gate, deciding whether an otherwise-closable relation is in fact settled, while the individual gap signals it aggregates determine which action is worth taking next.

At each step the controller selects the highest-scoring legal action among running an untried view, querying a facet, verifying an unresolved candidate, performing a reverse check, resampling with a different template, and stopping:

$$A_t(a) = \lambda_y Y_t(a) + \lambda_g G_t(a) + \lambda_i I_t(a) - \lambda_c \operatorname{cost}(a) - \lambda_d D_t(a),$$

where $Y_t(a)$ is the expected yield of a, $G_t(a)$ scores how directly a closes the specific gaps behind q_{res}, such as unresolved candidate mass, mechanism gaps, and facet gaps, $I_t(a)$ is the expected uncertainty reduction, $\operatorname{cost}(a)$ is model-call cost, and $D_t(a)$ penalizes redundancy with actions already taken. The relation contract determines which actions are legal and which gaps count, while the utility coefficients are global constants of the frozen configuration; both are fixed in advance, so the policy is deterministic and budget-aware.

3.5 Relation-Aware Finalization

After acquisition stops, the relation contract maps the evidence state to the final object set. The principle shared by all four selectors is that finalization operates on the evidence state rather than on generated text: a string that was never resolved into supported evidence is not eligible for output. SMALLSET selectors emit validated entities and drop unresolved candidates; NULLSINGLE selectors emit at most one object and otherwise preserve the empty set; LARGEOPENSET selectors retain accepted candidates after normalization and deduplication; NUMERIC selectors operate on value clusters rather than raw strings.

Numeric finalization clusters parsed values by relative distance,

$$d(a,b) = \frac{|a-b|}{\max(|a|,|b|)},$$
$$a \sim b \iff d(a,b) \le \tau,$$

which absorbs rounding and unit-conversion variation while keeping genuinely different reported values apart. The core selector uses $\tau = 0.025$, deliberately stricter than the 5% relative tolerance of the official evaluator, so that two values merge only when they agree well within the scoring margin. When no cluster reaches sufficient independent support, a second round of numeric acquisition arbitrates among the competing values at $\tau = 0.05$, the granularity the metric actually rewards. In either case a source-blind numeric judge compares the values without treating the upstream prediction as an additional vote, so a confidently generated but poorly supported number cannot win by repetition.

The same separation applies to existence. For city-of-death queries the system first resolves whether the subject is deceased, and only then asks for the city, with the response checked against a strict city parser. Treating existence as a prior gate rather than a post-filter is what lets the system return a principled empty set instead of a failed recall.

Relation	Macro Precision	Macro Recall	Macro F1
awardWonBy	0.3255	0.3707	0.3105
companyTradesAtStockExchange	0.8992	0.7963	0.7385
countryLandBordersCountry	0.9712	0.9295	0.9291
hasArea	0.6700	0.6700	0.6700
hasCapacity	0.2449	0.1633	0.1633
personHasCityOfDeath	0.9600	0.5900	0.5700
All relations	**0.7268**	**0.6055**	**0.5878**
Zero-object cases	**0.6038**	**0.9412**	**0.7356**

Table 2: Official hidden-test results. Zero-object cases denote rows whose gold object set is empty.

Relation	Direct	COVER-KBC
awardWonBy	0.079	**0.250**
companyTradesAtStockExchange	0.640	**0.650**
countryLandBordersCountry	0.941	**0.959**
hasArea	0.460	**0.580**
hasCapacity	0.103	**0.250**
personHasCityOfDeath	0.490	**0.510**
All relations	0.492	**0.560**

Table 3: Macro-F1 on the validation set. Direct denotes a single-pass 5-shot baseline using the same Mistral-Small-3.2-24B checkpoint as COVER-KBC. COVER-KBC values are those reported for the submitted system.

A related pattern appears in stock-exchange queries. An empty prediction triggers candidate-blind views targeting direct listing evidence, primary listing, ordinary-share listing, and anti-prior verification, since an empty output more often reflects failed elicitation than a genuine absence. Structural validation then rejects ticker symbols, indexes, regulators, market segments, country names, and company names, and an exchange is emitted only with sufficient independent support.

Together these selectors close the loop opened in Section 3.1: one contract fixes both which evidence a relation may acquire and how that evidence becomes an answer, so relation conditioning shapes the whole inference path rather than only the prompt.

4 Experimental Setup

We use the official AKBC Shared Task 2026 benchmark and evaluator.[2] Hidden-test labels are not released, so test results are reported by the official hidden-test evaluation service.

All neural inference uses a single frozen Mistral-Small-3.2-24B checkpoint.[3] The model has 24,011,361,280 published parameters, below the 32B task limit, and is reused for all logical roles in the system. We run inference on Google Colab with one NVIDIA A100 GPU using the Hugging Face backend, `mistral-common` tokenizer, `nf4` quantization, and `bfloat16` dtype.

COVER-KBC is evaluated in a closed-book setting: no external retrieval, web search, external knowledge base, manual answer table, fine-tuning, LoRA, or additional neural model is used. All generation uses greedy decoding rather than sampling. Downstream parsing, aggregation, clustering, and final selection are deterministic given the model outputs.

5 Results

5.1 Hidden-Test Performance

Table 2 reports the official hidden-test results for COVER-KBC. The system achieves 0.5878 Macro-F1 overall. Performance is strongest on `countryLandBordersCountry`, where the relation has a bounded small-set structure and benefits from conservative evidence aggregation. `companyTradesAtStockExchange` also obtains strong performance, with high precision and improved recall from the escalation views that revisit empty predictions.

The main weakness is `hasCapacity`. The task defines capacity as the highest published spectator capacity, so the difficulty is not that venues have several reported figures but that the system does not reliably target the maximum among them: precision and recall are both low, indicating that the accepted cluster is often centred on a seated or pre-renovation figure rather than the intended one. The open-set `awardWonBy` relation is also challenging, mainly because many awards have long recipient lists and recall is difficult under closed-book inference; with only ten test subjects, its scores are also the noisiest in the table.

[2] https://github.com/lm-kbc/dataset2026

[3] Hugging Face model id: mistralai/Mistral-Small-3.2-24B-Instruct-2506.

5.2 Comparison with a Same-Backbone Direct Baseline

Table 3 compares COVER-KBC with a single-pass 5-shot baseline using the same Mistral-Small-3.2-24B checkpoint; the direct baseline achieves 0.492 Macro-F1, while COVER-KBC reports 0.560, with the largest gains on `awardWonBy`, `hasArea`, and `hasCapacity`.

5.3 Relation-Level Observations

The results support the central design choice of COVER-KBC: different relation types require different inference and finalization behavior. Small-set relations benefit from strict validation and conservative set closure. Nullable relations require preserving empty outputs unless the evidence supports a singleton. Numeric relations benefit from multi-view scalar clustering, but remain sensitive to ambiguity in the target definition. Large open-set relations remain recall-limited because the system must discover many valid objects without external retrieval.

6 Conclusion

We presented COVER-KBC, a closed-book KBC system that treats object-set prediction as relation-typed active evidence acquisition. Instead of mapping one prompt response directly to a final answer, COVER-KBC accumulates structured evidence, separates candidate confidence from residual search need, verifies candidates blindly, and finalizes outputs with relation-specific selectors.

Using a single frozen Mistral-Small-3.2-24B checkpoint under the 32B parameter budget, COVER-KBC achieves 0.5878 Macro-F1 on the official hidden test set. The results show that heterogeneous KBC relations benefit from specialized inference programs: small-set relations benefit from conservative set closure, nullable relations require explicit empty-answer handling, numeric relations benefit from multi-view clustering, and open-set relations require continued discovery. These findings support relation-aware evidence acquisition as a practical direction for closed-book knowledge-base construction.

Limitations

COVER-KBC is designed for the six relations in the shared task. Its relation contracts, evidence views, parsers, and final selectors are manually specified, so extending the system to new relations would require additional relation-level engineering. The approach is therefore not a universal relation-agnostic KBC method.

The system is also limited by the factual knowledge stored in the frozen language model. Because no retrieval, external knowledge base, or additional training is used, COVER-KBC can miss facts that are absent, stale, or weakly represented in the model parameters. This is most visible for `awardWonBy`, which remains recall-limited, and for `hasCapacity`, where the model rarely commits to the maximum published figure that the task definition requires.

Finally, multi-step evidence acquisition improves control but increases inference cost compared with a single-prompt baseline. Hidden-test labels are not released, so detailed error analysis on the official test set is limited to official aggregate scores and relation-level aggregates.

A Hyperparameters

Table 4 lists the candidate-scoring weights, verifier calibration parameters, and per-query budgets of the released configuration. Most values are constants of that configuration; the action-planner coefficients are calibrated once on the provided training split and then frozen. No neural parameters are updated at any point, and no validation or test data is used for calibration.

Component	Parameter	Value
Candidate score	α (support)	1.0
	β (verifier)	0.6
	γ (cross-model)	0.5†
	δ (contradiction)	1.5
	η (disagreement)	1.0
Verifier	T	1.0
	ϵ	10^{-6}
	log-odds clip divisor	3.0
Budget	call budget per query	12‡
	token budget per query	6000‡
Stopping	residual threshold	0.15§

Table 4: Configuration of the submitted system. Scoring and verifier values are global constants; the stopping threshold is set per relation by the contract. †Configured but inactive: $X(o) = 0$ when all neural roles share one checkpoint. ‡Core controller cap; relation-specific escalation calls are separately capped in the released configuration. §Contract default.

Acknowledgments

This research is partially supported by the research funding from the Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam.

References

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396, Singapore. Association for Computational Linguistics.

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-wise self-consistency for LLM probing. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025) co-located with the 24th International Semantic Web Conference (ISWC 2025)*, volume 4041 of *CEUR Workshop Proceedings*, Nara, Japan. CEUR-WS.org.

Dimitrios Alivanistos, Selene Báez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. In *ISWC 2022 Posters, Demos and Industry Tracks*, CEUR Workshop Proceedings. CEUR-WS.org.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021. Knowledgeable or educated guess? revisiting language models as knowledge bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1860–1874, Online. Association for Computational Linguistics.

Arunav Das, Nadeen Fathallah, and Nicole Obretincheva. 2024. Navigating nulls, numbers and numerous entities: Robust knowledge base construction from large language models. In *Joint Proceedings of the 2nd Workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM 2024) and the 3rd Challenge on Language Models for Knowledge Base Construction (LM-KBC 2024) co-located with the 23rd International Semantic Web Conference (ISWC 2024)*, volume 3853 of *CEUR Workshop Proceedings*, Baltimore, USA. CEUR-WS.org.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2024. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand. Association for Computational Linguistics.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 601–610, New York, NY, USA. Association for Computing Machinery.

Jingbo He and Simon Razniewski. 2025. Combining self-retrieval-augmented generation with divide-and-conquer for language model-based knowledge base construction. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025) co-located with the 24th International Semantic Web Conference (ISWC 2025)*, volume 4041 of *CEUR Workshop Proceedings*, Nara, Japan. CEUR-WS.org.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. 2024. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, Miami, Florida, USA. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Simon Razniewski, Hiba Arnaout, Shrestha Ghosh, and Fabian M. Suchanek. 2024. Completeness, recall, and negation in open-world knowledge bases: A survey. *ACM Computing Surveys*, 56(6):1–42.

Raghuveer Thirukovalluru, Yukun Huang, and Bhuwan Dhingra. 2024. Atomic self-consistency for better long form generations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12681–12694, Miami, Florida, USA. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

MAVE: Multi-Angle Voting for Emission in Closed-Book Knowledge Base Construction

Jonas Kunze and Andreas Both
Web & Software Engineering (WSE) Research Group
Leipzig University of Applied Sciences (HTWK Leipzig)
Leipzig, Germany
{jonas.kunze, andreas.both}@htwk-leipzig.de

Abstract

Pretrained language models hold factual knowledge in their parameters, and the LM-KBC challenge measures how much of it can be turned into a knowledge base: given a subject and a relation, a system must return every object that completes the pair, and nothing at all when there is none. The 2026 edition forbids any lookup and caps all models used together at 32B parameters. Because the model may be chosen freely for the first time, a system has three things to decide: which model to run, how to put the question to it, and how to choose an answer from what comes back. We propose MAVE, which puts each item to one open-weight 24B model as several different questions and gives each question one vote. Each of the six relations applies its own rule for how much agreement it takes before an object is emitted. It reaches an official test macro-F1 of 0.6908, which places it second on the final leaderboard.

1 Introduction

Reading a pretrained language model as a knowledge base has been possible since Petroni et al. (2019) recovered relational facts from masked models, and the LM-KBC challenge has measured how far it carries since 2022 (Singhania et al., 2022). How far it carries has depended on what each edition allowed: with a lookup in Wikidata the 2024 winner reached an average F1 of 0.9224, and without it the 2025 winner reached 0.4439 (Bara, 2024; Albert-Roulhac and Zouaq, 2025). The 2026 edition keeps that closed book and replaces the fixed model of 2025 by a budget of 32B over open-weight models, which leaves a system three things to get right: which model to run, how it asks, and what it does with the answers.

A system that asks in many ways collects more candidate objects than it can emit, and it has to decide which of them to keep.

This paper therefore asks:

RQ With retrieval forbidden and the total LLM parameter count capped, how can the factual knowledge held in a pretrained model be extracted into a knowledge base?

Our answer is MAVE (Figure 1).[1] It puts each item to the model through several questions, lets each question cast exactly one vote whatever its number of calls, and decides per relation how much agreement a candidate needs.

2 Task and Data

An item is a subject entity together with a relation, for instance "Portugal" and `hasArea`, and the expected answer is the complete set of objects for that pair, taken from what the model memorized during pretraining (Kalo et al., 2026). That set may hold one object, many, or none at all.

The six relations. They differ in the shape of their answer, and Table 1 gives their sizes alongside the score our system reaches on each. Two ask for a number, the surface area of a geographic entity (`hasArea`) and the capacity of a venue (`hasCapacity`); one for at most one entity, the city in which a person died (`personHasCityOfDeath`); three for a set, the countries sharing a land border with a country (`countryLandBordersCountry`), the stock exchanges a company is listed on (`companyTradesAtStockExchange`), and the people who received an award (`awardWonBy`), the extreme case with 111.1 objects per item on average.

Scoring. The official score is the mean of the per-item F1 over all items of a split, so every item weighs the same, regardless of which relation it belongs to. Within an item, the predicted set is matched one-to-one against the gold set, and a prediction matches an object when it equals any of

[1] The code, the resolved configuration of every run, and the verbatim prompts are available at https://github.com/WSE-research/MAVE-LM-KBC-2026.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 488–495
October 28, 2026 ©2026 Association for Computational Linguistics

Relation	Answer	Items			Empty gold	F1, MAVE		
		Train	Val	Test		Train	Val	Test
`countryLandBordersCountry`	set	67	68	67	22%	0.99 ±0.001	0.98 ±0.002	0.99
`companyTradesAtStockExchange`	set	100	100	100	36%	0.73 ±0.012	0.78 ±0.010	0.85
`hasArea`	number	100	100	100	0%	0.83 ±0.012	0.85 ±0.008	0.84
`personHasCityOfDeath`	one entity	100	100	100	41%	0.52 ±0.025	0.61 ±0.012	0.59
`awardWonBy`	set	10	10	10	0%	0.51 ±0.006	0.24 ±0.008	0.36
`hasCapacity`	number	100	97	98	0%	0.25 ±0.006	0.26 ±0.020	0.31
Macro-F1		477	475	475		0.637 ±0.005	0.670 ±0.007	**0.6908**
Organizers' baseline								0.3085

Table 1: The six relations, sorted by our F1 on the test split. "Answer" is the shape the task asks for, and every one of them may also be empty: "Empty gold" is the share of items where it is. Test scores are the organizers' scorer on our submission; train and validation of our own evaluation against the released gold. Each ± is the standard deviation over five repeated draws of the same system.

that object's aliases after normalization; on the two numeric relations, a prediction counts as correct within a tolerance of five percent. An item whose gold answer is the empty set scores 1.0 exactly when the prediction is empty as well, and 0.0 for any prediction.

Three degrees of freedom. The 2026 edition is the first in which the model may be chosen freely (Section 3), so all three questions named above are open at once. The first is settled in Section 5 by measurement, and the rest of the paper works on the other two.

The rules. Two constraints shape every design decision in this paper. The task is closed book, so retrieval and any lookup in an external corpus or knowledge base are forbidden at inference time, and fine-tuning of any kind is excluded. The parameter budget is 32B summed over every neural component used at inference, counted on the published total of each model, which quantization does not reduce and a mixture-of-experts model does not escape. What the rules leave open is multi-step inference and post-processing without a neural component. The system of Section 4 works entirely within that space.

3 Related Work

What a model has memorized. Closed-book question answering established that a model can answer from its parameters alone and that how much it answers grows with its size (Roberts et al., 2020). What it has kept is uneven, and the unevenness is not random: a fact is recalled roughly in proportion to how often documents about it appeared in pretraining, so entities in the tail of that distribution are missing however large the model (Kandpal et al., 2023), and the popularity of the subject predicts when the parameters suffice and when they do not (Mallen et al., 2023). The six relations of this task differ along exactly that axis.

Four editions, four sets of rules. The challenge turned the question into a shared task with a fixed dataset, metric, and leaderboard (Singhania et al., 2022); 2026 is its fifth edition. The first two placed no restriction on the model (Singhania et al., 2022; Kalo et al., 2023), the 2024 edition capped it at ten billion parameters and left retrieval open (Kalo et al., 2024), and the 2025 edition removed retrieval and fixed one model for every participant (Kalo et al., 2025), which is where the two winning scores (cf. Section 1) come from.

Recurring techniques. Four moves are used by more than one system. The same question is put to the model in more than one wording (Alivanistos et al., 2022; He and Razniewski, 2025); chain-of-thought reasoning is added as a level of a prompt taxonomy (Bara, 2024), as reasoning paths distilled from a larger model (Albert-Roulhac and Zouaq, 2025), and as trainable prompts inside the chain (Creo et al., 2025); the model is asked to check its own candidates before they are emitted (Alivanistos et al., 2022; Clay et al., 2025); and the empty answer is treated as a case of its own rather than as a failure to answer (Alivanistos et al., 2022; Das et al., 2024). Two moves appear once each and are closest to ours: sampling one prompt twenty times and aggregating per relation (Albert-Roulhac and Zouaq, 2025), and having the model write a description first and read the answer out of it (He and Razniewski, 2025).

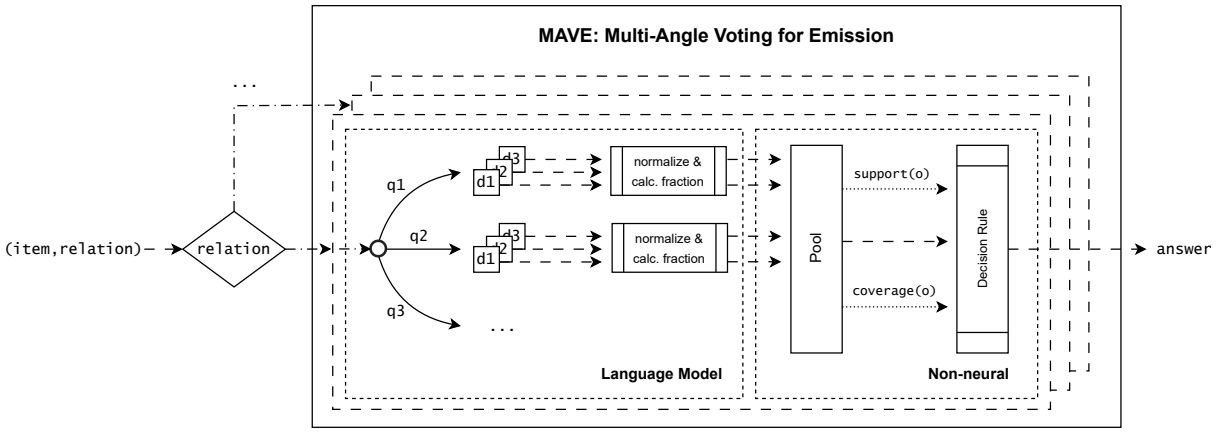


Figure 1: The pipeline for one item. The relation selects both the questions and the decision rule. Each question q_i is put to the model k times; its answers d_j are normalized and collapsed into one fraction per candidate object. The support and the coverage of each candidate o are then computed over the pool, and both go into the decision rule, which picks the final answer.

4 Method

MAVE asks the same fact in several different ways and lets the answers decide among themselves. Two rules govern the pool of candidates that comes back. A formulation counts once, however often it was asked, so a question put twenty times cannot outvote one put once. A formulation that recalls nothing abstains, so ignorance is never evidence against a candidate. What is then left to decide is a single quantity: how much agreement an object needs before it is emitted. Section 4.2 settles it.

The five steps below describe how the system turns a subject and a relation into an answer (Figure 1).

Step 1: Ask. Each relation has its own set of questions, between two and five of them, and they are not rewordings of one another but different text genres, listed in Table 2. Every question is put to the model k times. At $k = 1$ the call is greedy; for $k > 1$ the calls are sampled at a temperature above zero, so the k answers may differ.

Step 2: Normalize. The model maps the surface forms of one object onto a canonical form, so that "NYSE" and "New York Stock Exchange" become one candidate and two spellings the official scorer would accept as equal and never split a vote. The spelling finally printed is the one that occurred most often.

Step 3: From answers to fractions. The k answers to one question are first collapsed into a single fraction per candidate, namely how often that question produced it, defined as

$$\text{share}_q(o) \;=\; \frac{|\{\, d \in D_q \;:\; o \in d \,\}|}{|D_q|}, \qquad (1)$$

where D_q are the k answers to question q. A question asked once contributes 0 or 1; a question asked twenty times contributes anything in between. It is these fractions, one per question and candidate, that go into the pool.

Step 4: Two numbers over the pool. Let Q be the questions of the relation. For every candidate o, the pool yields exactly two quantities. The support averages what the questions gave it,

$$\text{support}(o) = \frac{1}{|Q|} \sum_{q \in Q} \text{share}_q(o), \qquad (2)$$

while the coverage ignores how often a question produced the candidate and counts only how many questions produced it at all,

$$\text{coverage}(o) = |\{\, q \in Q \;:\; \text{share}_q(o) > 0 \,\}|. \qquad (3)$$

The denominator of the support is the number of questions asked, not the number that answered: a question that recalled nothing lowers the support of every candidate, but it is never a vote against one.

Step 5: Decision rule. Every relation has its own decision rule, which reads the pool together with these two numbers. The shape of the answer the task asks for, a number, at most one entity, or a set, narrows down which rules are possible, and within that the rule is chosen per relation.

4.1 Questions and Decision Rule of Each Relation

The six paragraphs below sketch each relation; the rules in full, with every prompt and every parameter, are in the repository.[1]

Countries that share a land border. Three questions, emitted at a coverage of two. The third asks which landmass the subject sits on and who else is on it, and it has a second use: a country the model places alone on an island can have no land neighbors whatever the other two just said, so that answer can remove a neighbor, never add one.

Stock exchanges a company trades at. Five questions in five different text genres, emitted at a coverage of two. Several of them also let the model state that the company is not listed at all; if more answers deny a listing than affirm one, nothing is emitted, while an answer that recalls nothing counts for neither side.

The surface area of a geographic feature. Three questions, two of which first have the model write a passage and then read the number out of it. Here support decides: the system slides a window of five percent, the scorer's own tolerance, over the numbers in the pool and emits the center of the window whose numbers carry the most support.

The city in which a person died. Four questions of one shape, each having the model write a passage and then read the city out of it. The city with the most support is emitted only when its coverage is more than half of the four, and nothing otherwise, which matters because for 41% of the items the gold answer is empty and an empty prediction scores a full point.

The people who received an award. Four questions that cut the same question differently, emitted at a coverage of two. Cutting is the only way to ask at all here, because the gold holds 111 names per item on average, far more than any single answer returns.

The capacity of a venue. Two questions, each asked twenty times at a temperature of 1.0. Numbers are weighed as for the area, and this is the one relation where the system also decides how many numbers to emit: a second one caps the item at 0.667, and it is emitted when its support is at least half that of the first.

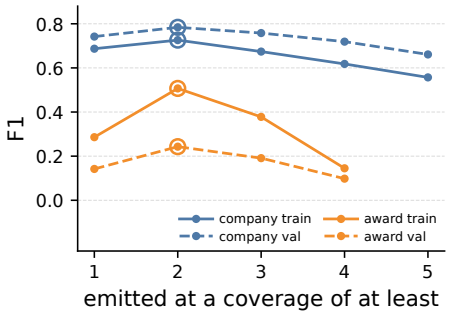


Figure 2: F1 of the two set-valued relations that can reach more than two agreeing questions, at every coverage level, with everything else held fixed. The rings mark the level the system uses.

4.2 How Much Agreement a Candidate Needs

The three relations that answer with a set leave one quantity open: how many of their questions must have named a candidate before it is emitted. Emitting at a lower coverage takes in more candidates, which raises recall and lowers precision inside the same item; emitting at a higher one does the reverse. Both directions cost score, so the level is settled by measurement on the labelled splits, with everything but the level held fixed. Two is best for `companyTradesAtStockExchange` and `awardWonBy` on train and on validation alike, and the fall-off on either side is steep (Figure 2). `countryLandBordersCountry` has only two questions feeding its pool and therefore no level above two; there one scores 0.017 higher than two on each split, on a relation that already reaches 0.99, so two is applied throughout.

5 Experimental Setup

Which model. We settled the choice of model by measurement. Running the same pipeline on three open-weight models inside the budget, with the questions, draw counts, seeds, temperatures, decision rules, and post-processing all held fixed, gives a validation macro-F1 of 0.6699 for `mistral-small-3.2` at 24B, 0.5645 for `qwen3.6` at 27B, and 0.5293 for `gemma-4` at 31B. The distance is large and does not follow the parameter count, since the 31B model is the weakest of the three.

That comparison is unfair in one direction: every question, draw count, and decision rule was chosen while `mistral-small-3.2-24b-instruct` was in place, so it says how much of our score the backbone carries and not which model is strongest when each is tuned for itself. Both of the others did hold

that position earlier in this project, and `gemma-4` was re-tuned rather than only swapped in; its emission thresholds refit relation by relation and its numeric questions rebuilt four ways; it still lost on every relation that asks for an exact figure about an entity that is not famous. Every prompt therefore goes to `mistral-small-3.2-24b-instruct`, served through a commercial provider with its reasoning mode disabled.

The budget also allows a second model of up to 8B besides the 24B one, and for the three, we tried it, and it did not pay off. Switching `llama-3.1-8b` off was better or equal on both relations it had served and both splits; `ministral-8b` lost on both labelled splits, and `qwen3-8b`, the model the 2025 edition prescribed, reached 0.567 as a partner where the pair it was compared against reached 0.582. A smaller model names few objects that the larger one misses, so it corroborates what is already there and adds false positives of its own.

Temperature. The first draw of every question is greedy and the further draws are sampled at temperature 0.7, where the question varies a judgment, and at 1.0, where it widens a pool of numbers. Seeds are fixed per question, and every response is cached, so re-aggregating a decision over an unchanged pool of draws is exactly reproducible. Across the six relations, the shipped configuration issues 32.0 calls to the model per item.

How configurations were selected. Every design decision was made on the training and validation splits; the test split ships without gold answers, so every test number in this paper comes from the organizers' scorer on a submission of ours.

6 Results

MAVE reaches an *official test macro-F1 of 0.6908*, scored by the organizers on the submission platform. That is second on the final leaderboard, behind a 31B model at 0.7060. The organizers' baseline, a single direct generation per item, reaches 0.3085 on the same split. Table 1 gives the score per relation on the three splits. The six are taken one at a time below against Table 2, which counts for each question the share of the gold it names (recall) and the share of what it names that is gold (precision).

6.1 The Six Relations

The relation `countryLandBordersCountry` is the one that the model simply knows. A single question already names 195 of the 198 gold neighbors on train at a precision of 0.99, and the second question adds none of the missing three there and two on validation, so the pool is at its ceiling with one question and the rule only has to avoid spoiling it.

`companyTradesAtStockExchange` shows the trade-off the whole system is built around. The opening of an encyclopedia article reaches the most exchanges of the five questions and is the least precise of them, while the plain question reaches the fewest at the highest precision. The pool that collects them is therefore less precise than any single question in it, 0.42 against 0.52 to 0.73 on train, and buying that precision back is what the quorum of Section 4.2 spends its two agreeing questions on. A third of the companies are not listed anywhere, and there the full point goes to an empty prediction, so part of what the quorum buys is the ability to stay silent.

The objects for the relation `hasArea` are asked three ways that turn out to be one. All three land within four points of one another in recall and in precision alike, and the pool ends up three points above the best of them on train and five on validation, so a fourth way of asking would be wasted.

`personHasCityOfDeath` gains the most from asking differently and loses the most at the decision. Having the model narrate a career or the final years and then reading the city out of that passage reaches 0.59 and 0.53 of the gold cities on train, where the two questions that ask more directly reach 0.40 each, and the pool reaches 0.64 against a score of 0.52. At a pool precision of 0.07, the right city is usually in the pool and usually outvoted, so what is missing here is the choice among candidates the model did produce.

In our setting, the best results for the relation `awardWonBy` were achieved while asking in slices. With 111 names per item on average, no single answer holds the set, and the question that asks one year at a time names 609 of the 764 gold names on train, 90 of which no other question names at all. With ten items per split, though, nothing measured on this relation resolves below a tenth of a point.

`hasCapacity` is the second relation whose loss sits in the decision. Its two questions each find twelve to sixteen values the other misses, which lifts the pool to 0.68 on train, where the best single question reaches 0.54, and the relation still scores 0.25. The scorer's tolerance of five percent is wide, so some of that recall could be luck: a pool of the same size drawn at random from the training

	Recall		Precision	
Question	Tr	Va	Tr	Va
`countryLandBordersCountry`				
which countries border it	.98	.96	.99	.93
how many, and which ones	.98	.93	.97	.92
any of them	.98	.97	.98	.91
`companyTradesAtStockExchange`				
the plain question	.51	.62	.73	.85
a company record as JSON	.53	.53	.69	.78
an encyclopedia opening	.72	.76	.52	.62
the home-market language	.56	.66	.57	.62
the ticker symbol	.56	.67	.65	.76
any of them	.81	.84	.42	.46
`hasArea`				
a gazetteer entry	.85	.87	.60	.60
comparable features	.85	.86	.63	.60
the two nearest in area	.83	.84	.59	.58
any of them	.88	.92	.52	.51
`personHasCityOfDeath`				
a short biography	.40	.44	.19	.22
the year, then the city	.40	.41	.34	.38
the career city by city	.59	.67	.11	.14
the final years	.53	.62	.11	.14
any of them	.64	.74	.07	.09
`awardWonBy`				
one period at a time	.58	.16	.57	.25
the plain question, 8 draws	.47	.14	.27	.16
one field of work at a time	.33	.16	.30	.16
one year at a time	.80	.28	.28	.14
any of them	.85	.36	.19	.10
`hasCapacity`				
an encyclopedia opening	.54	.65	.12	.12
a venue record as JSON	.52	.66	.10	.14
any of them	.68	.78	.11	.12

Table 2: What each question puts into the pool, before any decision rule sees it, on train (Tr) and validation (Va): the share of the gold it names (recall), and the share of what it names that is gold (precision). The "any of them" row is the whole pool, and its recall is the ceiling on the relation.

distribution of capacities lands inside the window for 42.0% of the items. The model reaches 68.0% and 78.4%, six standard deviations above, so the values are recalled and then lost at the decision.

6.2 Against the 2025 Winner

ReWiSe won the 2025 edition with one question asked twenty times, chain-of-thought examples, and one aggregation rule per relation (Albert-Roulhac and Zouaq, 2025). Re-running it on the 2025 validation split scores 0.424 against the 0.415 its authors report, with no relation off by more than 0.023. Table 3 then carries it to this year's data and to our backbone: on `qwen3-8b` the two systems are level, so the distance on the stronger model is as much a property of the backbone as of the method.

2026 validation	`qwen3-8b`	`mistral-24b`
ReWiSe	0.435	0.624
MAVE	0.443	0.670

Table 3: Validation F1 of the 2025 winner and of MAVE on the 2026 data, run on the model that edition prescribed for everyone and on ours.

6.3 How Far the Same System Drifts

The same configuration does not return the same answer twice. Four of the six relations sample at a temperature above zero, so their pools are redrawn on every run, and even the greedy calls do not repeat, because a model served through a hosted endpoint answers the same prompt differently from one run to the next (Atıl et al., 2025). The spread beside each score in Table 1 is what redrawing alone does, and it understates the whole: drawn a second time from scratch, one configuration moved by 0.011 on train and 0.031 on validation, and single relations further. A difference of one or two items is therefore below what two runs resolve.

7 Conclusion

On RQ, what carries under a closed book and a capped parameter count is asking one open-weight 24B model in many ways: from the 0.3085 of a single direct generation to an official macro-F1 of 0.6908. Asking again pays where the questions reach the fact differently, above all where the model writes a passage first and the answer is read out of it, and it pays nothing where they repeat one another. It is reached on 24B of the 32B the rules allow, a quarter under the cap, and none of the three smaller partners we tried beside that model paid for its parameters: a competitive system here is a matter of how the model is asked rather than of how many parameters are spent. What limits the system now is no longer only what the model recalls: on the two weakest relations the right answer is usually in the pool and gets outvoted. Of the 21 system papers of the four earlier editions, one gives an interval around its headline score (Bara, 2024); on splits this size a spread belongs beside every number.

Limitations. The results rest on one model of one family, and the decision layer had to be re-derived when we changed models, which suggests the quorum is a property of the model as much as of the relation. Our score is higher on test than on either labelled split, so the numbers we can verify are the conservative ones.

Acknowledgments

We thank Optimax Energy GmbH for their collaboration as a practice partner. This work was funded by the German Federal Ministry of Research, Technology and Space (BMFTR), grant number 03FHP239.

References

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-wise self-consistency for LLM probing. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 4041 of *CEUR Workshop Proceedings*.

Dimitrios Alivanistos, Selene Baez Santamaria, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. In *Proceedings of the Semantic Web Challenge on Knowledge Base Construction from Pre-trained Language Models (LM-KBC)*, volume 3274 of *CEUR Workshop Proceedings*.

Berk Atıl, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. 2025. Nondeterminism of "deterministic" LLM system settings in hosted environments. In *Proceedings of the 5th Workshop on Evaluation and Comparison of NLP Systems*, pages 135–148, Mumbai, India. Association for Computational Linguistics.

Davide Mario Ricardo Bara. 2024. Prompt engineering for tail prediction in domain-specific knowledge graph completion tasks. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 3853 of *CEUR Workshop Proceedings*.

Alex Clay, Ernesto Jiménez-Ruiz, and Pranava Madhyastha. 2025. Judge, generator, executioner: Utilizing an LLM for KBC. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 4041 of *CEUR Workshop Proceedings*.

Aldan Creo, Christophe Gueret, Alberto Bernardi, Adrianna Janik, and Luca Costabello. 2025. Soft thinking: Enhancing knowledge base completion through trainable prompts in the chain of thought. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 4041 of *CEUR Workshop Proceedings*.

Arunav Das, Nadeen Fathallah, and Nicole Obretincheva. 2024. Navigating nulls, numbers and numerous entities: Robust knowledge base construction from large language models. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 3853 of *CEUR Workshop Proceedings*.

Jingbo He and Simon Razniewski. 2025. Combining self-retrieval-augmented generation with divide-and-conquer for language model-based knowledge base construction. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 4041 of *CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2024. Preface: LM-KBC challenge 2024. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 3853 of *CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pre-trained language models. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 4041 of *CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Sneha Singhania, Simon Razniewski, and Jeff Z. Pan. 2023. LM-KBC 2023: 2nd challenge on knowledge base construction from pre-trained language models. In *Proceedings of the Workshop on Knowledge Base Construction from Pre-trained Language Models (KBC-LM) and the Challenge (LM-KBC)*, volume 3577 of *CEUR Workshop Proceedings*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Sneha Singhania, Tuan-Phong Nguyen, and Simon Razniewski. 2022. LM-KBC: Knowledge base construction from pre-trained language models. In *Proceedings of the Semantic Web Challenge on Knowledge Base Construction from Pre-trained Language Models (LM-KBC)*, volume 3274 of *CEUR Workshop Proceedings*.

MuseList: Confidence-Gated Multi-Surface Recall for Closed-Book Knowledge Base Construction

Adrian Zimmermann
Technische Universität Dresden, ScaDS.AI Dresden/Leipzig
adrian.zimmermann@mailbox.tu-dresden.de

Abstract

Knowledge base construction requires complete answer sets, not just one plausible fact. We present MuseList, a closed-book system for the six-relation AKBC Shared Task 2026. A single open-weight 29.8B-parameter Muse Glimmer model generates candidate answers in several formats, such as recipient lists, recipient–year tables, and venue records. All such records are model-generated; the system retrieves no external documents. Relation-specific rules combine these candidates, select numerical answers using token-level confidence signals, and check proposed reciprocal borders with three separately prompted roles. The complete system scores 0.6700 hidden-test macro-F1 and ranks fourth among twelve systems. Its development validation score is 0.69160; this split informed method selection and is not an untouched evaluation. We describe the method and release an archive that reconstructs the scored submission exactly. Fresh-call support is separate and partial: it regenerates later-stage responses, not an end-to-end submission.

1 Introduction

Language-model knowledge probing commonly asks whether a model can produce or rank a known fact (Petroni et al., 2019; Jiang et al., 2020). LM-KBC instead asks for the complete object set of each subject–relation pair (Alivanistos et al., 2022; Kalo et al., 2025). Correct cardinality may be zero, one, or hundreds: hallucinations lower precision, omissions lower recall, and the best behavior for a country border differs sharply from that for an award with a century of recipients. The 2026 task additionally forbids web retrieval and extra training and caps the sum of inference-time neural parameters at 32B (Kalo et al., 2026).

Our approach separates two problems: eliciting candidate facts and deciding which candidates to retain. Instead of repeating one question, we ask the same model to produce different structured representations. For an award, for example, one prompt requests a recipient list and another requests a table of recipients and years. We call these prompting formats *surfaces*. They are cues for generation, not retrieved sources or independent factual evidence. Agreement between calls can be useful, but the same model may repeat the same error.

Different relations need different acceptance rules. Long award lists benefit from partitioning the recall task; numerical relations require choosing among conflicting values; land borders admit reciprocal proposals. A prediction that names country B as a neighbor of A suggests checking whether B should also name A. City of death retains a direct, conservative prediction route.

MuseList combines these relation-specific generation and acceptance steps in one model-based system. Our contributions are the resulting method, an analysis of its cumulative development and final test performance, and a public archive linking model requests and responses to the scored output.

2 Related Work

Prompt wording and answer extraction materially affect factual knowledge exposed by a fixed language model (Jiang et al., 2020; Alivanistos et al., 2022; Ghosh, 2023). Self-consistency samples multiple reasoning paths and aggregates agreement (Wang et al., 2023); our structured-output consensus similarly reuses one model, but varies output format and conditions acceptance on typed records rather than unconstrained prose.

Recent LM-KBC systems make relation-specific policies central. ReWiSe uses relation-wise self-consistency and generated reasoning traces (Albert-Roulhac and Zouaq, 2025). Divide-and-conquer systems separate candidate generation, refinement, and relation-specific

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 496–517
October 28, 2026 ©2026 Association for Computational Linguistics

extraction (He and Razniewski, 2025). Other shared-task papers study compact prompting and quantization (Sam, 2025) or reuse one model as generator and judge (Clay et al., 2025). These works motivate explicit call schedules, prompt examples, and relation-level analysis. Our setting differs in two important ways: external retrieval is prohibited in 2026, and all distinct neural components must fit a summed 32B budget.

Our source-form prompts ask the model to complete text resembling RDF, a table, or a URL. The system never opens those URLs or queries a knowledge base. This is structured generation conditioned on the task and permitted training examples, rather than retrieval of external evidence.

3 Task and Constraints

Each input row contains `SubjectEntity`, one of six relations, and an empty `ObjectEntities` list. The relations are award recipient, stock exchange, land border, area, venue capacity, and city of death. We use the post-disambiguation August release: train has 477 rows and validation and test have 475 rows each (validation/test capacity counts are 97/98) (LM-KBC 2026 Organizers, 2026a). Predictions retain exact input order and contain unique strings. The official evaluator computes precision, recall, and F1 per row and macro-averages across rows. It uses maximum bipartite alias matching after Unicode normalization and normalized deduplication; numeric area and capacity predictions match within 5% relative tolerance.

Our only neural component is Muse Glimmer 30B, accessed through OpenRouter. The public weight repository reports approximately 29.8B parameters and an Apache-2.0 license (Meta, 2026; OpenRouter, 2026). DeepInfra, Together, and Fireworks are serving routes for that model, not additional models. Provider fallback is disabled. We use no external retrieval, additional model training, test labels, or manual factual corrections. Released-train labels supply demonstrations and training-side policy selection; exact target keys are excluded from demonstration selection. The appendix records the model route and inference settings.

Why Muse Glimmer? Muse met the parameter and licensing constraints, supported long structured outputs needed for award enumeration, and was available with provider pinning. Released-train probes produced usable JSON, ticker, TSV, wikitext, and table completions. We therefore fixed this model and studied prompting and aggregation. This was an engineering choice, not a controlled comparison establishing that Muse was the best eligible model.

4 System

4.1 Common inference and parsing

Direct prompts establish initial answer sets. They contain the subject, a relation instruction, a natural-language question, demonstration examples, and an output schema. Stock, borders, and capacity use the first five eligible training rows for that relation; area and city of death use five examples selected for lexical relevance and output diversity. Awards use complementary recall instructions and recipient–year records. The stock route ultimately uses a ticker-based primary-exchange prompt, followed by secondary-listing recall.

The parser extracts JSON or typed records, removes empty markers, and merges duplicate strings after case folding, Unicode decomposition, punctuation removal, and whitespace normalization. It retains the first display form. Numeric relations retain parseable numbers. Final assembly checks the required keys, object strings, and row order before writing the submission.

4.2 Relation policies

An acceptance rule, or *gate*, determines whether generated evidence adds objects, replaces the current answer set, or leaves it unchanged. Table 1 summarizes the rules. Not all require repeated support: some enumeration stages retain every syntactically valid candidate. Rejected selection and verification steps preserve their immediate input, not necessarily the original direct answer.

4.3 Source-diverse set completion

For stock exchanges, a listing-code prompt asks Muse to recall a company's exchange-qualified ticker internally and return the exchange name. Three further prompts ask for secondary listings using annual-report, market-data, and security-master formats. Only exchanges appearing in at least two of these outputs are appended. These are model-generated views, not documents retrieved from financial sources.

For awards, the system first combines recipient lists and recipient–year records, then asks for additional names through list continuations and alter-

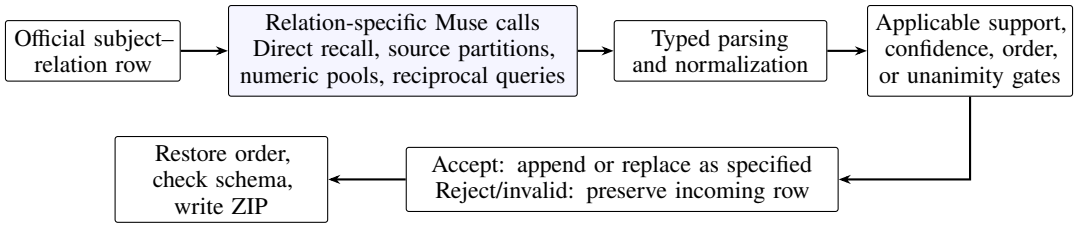


Figure 1: Execution data flow for MuseList. All blue boxes call the same Muse model; source-shaped strings are never dereferenced. Candidate generation and acceptance are separate, and failure of a gate preserves the incoming complete row. The appendix expands every relation-specific call and acceptance path.

Relation	Candidate source	Acceptance and action	Signal
Award	Lists, tables, and partitioned recall	Combine initial lists; apply the list-table replacement gate; prune aliases and append later support-qualified names	Parsing, length, support
Stock	Ticker-based primary listing; three secondary-listing roles	Generate the primary exchange; append a secondary exchange at normalized support $\geq 2/3$	Cross-role support
Borders	Missing reciprocal graph edge	Append after three unanimous land-border classifications	Unanimity
Area	Twelve numeric representations and an in-pool choice	Cluster candidates at 5%; replace the current answer with a selected value only at margin ≥ 20.7734	Token margin
Capacity	Coordinate surfaces and candidate-order variants	Nonempty surface union replaces the row; subsequent confidence and margin choices replace or preserve it (§4.4)	Parsing, confidence, margin
Death locality	Direct locality or abstention	Normalize one locality conservatively; allow empty	Direct output

Table 1: Final relation policy matrix. Every neural call uses the same Muse model; thresholds were selected on released train.

native table formats. Temporal, geographic, and given-name partitions divide long recall tasks into smaller ones. Further prompts generate award-edition and issuer records associated with candidates. Two groups of checking prompts ("witness families") require agreement on an award edition and selection of the target award over alternatives, respectively. Their prescribed support rules determine which candidates are added. A separate identity-grouping call, with the award title withheld, removes duplicate name variants within each answer set.

Before identity pruning and the later geographic, issuer, witness, and given-name additions, one award route asks the model to name a Wikipedia-style recipient-list page and then generate its table from memory. It replaces the incoming set only when the generated title has the required list prefix and at least 50 unique recipients parse. The page is never opened. This length rule is not evidence that the generated list is complete or correct.

4.4 Confidence-gated numeric choice

For capacity, prompts first recall venue coordinates and use those generated coordinates to elicit venue-record values. A nonempty union of these surface outputs replaces the preceding capacity set; if empty, that set is retained. Later choice prompts use a broader pool assembled from thirteen numerical sources. Muse must choose a listed value or abstain. We retain a first-pass choice only when the final nonempty-list decision token has log probability at least -0.0001. A later gate repeats the decision under two deterministic rotations. It counts only in-pool choices with mean token margin at least 22.4922 and accepts the uniquely most-supported value; one qualifying vote suffices, whereas a tie preserves the input. For each token overlapping the rendered answer, the margin is its log probability minus that of the highest-scoring alternative returned in the provider's token-score list; finite margins are averaged across answer tokens. A rejected later gate preserves its immediate input, including any accepted first-pass choice. For area, twelve distinct prompting formats—for example native units, geometry, gazetteer, temporal and co-

ordinate surfaces—are clustered under the official 5% tolerance. A no-demonstration in-pool choice replaces its input only above mean token margin 20.7734. All thresholds are fixed by strict leave-one-row-out train selection. Token scores are selection signals, not calibrated probabilities of factual correctness.

4.5 Reciprocal-border verification

From the current border predictions we derive only missing reciprocal edges: if A predicts B, both occur as task subjects, and B does not predict A, then $B \rightarrow A$ becomes a candidate. Three fixed Muse roles—a political-boundary editor, physical cartographer, and legal-geography auditor—separately return `BORDER` or `NOT_BORDER` at temperature zero. The exact pair is appended only under 3/3 agreement. On released train the gate yields 24 true positives, no false positives, eight true negatives, and no false negatives, including unanimous rejection of the false Qatar–United Arab Emirates proposal. Frozen validation adds South Ossetia to Russia and Palestine to Jordan.

5 Experimental Protocol

Proposed components are checked on released training data before transfer. Candidate prompts, parsers, thresholds, and acceptance criteria are fixed before the corresponding validation or test run. A transformation is applied to the entire split: individual validation rows are not accepted, reverted, or repaired after inspecting their labels. A favorable in-sample slice alone is insufficient; train-neutral or leave-one-out-negative alternatives are rejected.

We nevertheless observed complete-system validation scores repeatedly and used them to select configurations. Validation therefore measures development performance, not an untouched generalization estimate. Table 3 records this cumulative process rather than controlled component ablations.

The development analysis does not use intermediate test submissions as experimental evidence. The final test configuration was fixed before submission. One documented validation/test prompt difference concerns award-list traversal: test uses a train-qualified oldest-first variant, while the reported validation artifact retains the earlier prompt. The acceptance rule is shared; the appendix and archive specify both exact configurations. Codabench evaluates the final 475-row test submission and provides aggregate relation metrics, not private row-level labels.

6 Results

The final system scores 0.6700 hidden-test macro-F1 and ranks fourth among twelve systems in the official leaderboard (LM-KBC 2026 Organizers, 2026b). Table 2 gives its test precision, recall, and F1 alongside development-validation F1. Scores are averaged over rows, not equally over the six relations; award enumeration contributes ten rows despite its long answer lists.

Relation	Test P	Test R	Test F1	Val. F1
Award recipient	.4574	.3345	.3557	.2496
Stock exchange	.8900	.9123	.8400	.8333
Land borders	.9823	.9683	.9652	.9953
Area	.8600	.8500	.8500	.8500
Capacity	.3980	.2449	.2449	.3608
Death locality	.8800	.6100	.5700	.5500
All rows	.7840	.6936	.6700	.6916

Table 2: Final-system test results from Codabench, with development-validation F1 for context. Each split has 475 rows. Validation informed development, and the award traversal prompt differs between splits (§5).

Cumulative development configuration	Val. F1
Initial generation and selection	.66756
Secondary listings + award identity pruning	.67376
Geographic award partitions	.67384
Coordinate-based capacity replacement	.67524
Further source/witness award consensus	.67530
Capacity answer-confidence gate	.68302
Twelve-surface area selection	.68933
Order-varied capacity selection	.69144
Reciprocal-border verification	.69160

Table 3: Cumulative development trace with semantic component names. Values were observed adaptively and are descriptive, not isolated causal effects.

7 Analysis and Discussion

The largest increases in the cumulative validation trace coincide with confidence-based numerical selection. Smaller changes accompany additional set-completion prompts and reciprocal-border verification. These observations help describe development, but cannot establish the isolated contribution or necessity of any component: configurations were added sequentially and chosen using repeatedly observed validation performance. Several additional prompt panels, omission audits, and

numerical aggregation rules failed training-side checks and were excluded.

On hidden test, capacity has the lowest F1 (.2449). One possible failure mode is a confident historical or configuration-specific capacity rather than the target value; aggregate test metrics alone do not diagnose this cause. Award recall (.3345) is lower than precision (.4574), indicating that missing recipients remain a substantial problem; long lists and name variants also create opportunities for false positives. City-of-death recall (.6100) is lower than precision (.8800), consistent with its conservative output policy.

Border prediction is strongest overall, but exact-string reciprocity has a limitation: a proposed reverse edge can name an entity already present under another alias. Inspection of generated test outputs found that six of seven accepted additions were alternate names for already predicted entities. This is an output-identity observation, not a claim about private test labels or a measured test-score contribution.

8 Reproducibility

The public archive provides exact requests, raw responses, deterministic transformations, and intermediate outputs. From a clean checkout, `bash reproduce_improved_submission.sh OUTPUT` reconstructs the submitted 475-row prediction and ZIP byte-for-byte from archived model records. It checks input coverage, schema, order, and artifact identity. Released validation labels permit local scoring; the test score cannot be independently recomputed because its labels are private.

A separate driver can regenerate 2,917 later-stage verification and set-completion responses using recorded inference settings. It excludes the 811 core-generation test calls and does not assemble newly generated responses into a submission. This partial fresh-call facility must not be confused with end-to-end fresh reproduction. Shared prompt templates, split-specific settings, selection rules, and complete row traces are documented in the appendix; the code and archived records are available at `https://github.com/UnGast/muselist-lmkbc2026`.

9 Limitations

Separate prompts to one model can share errors; neither agreement nor high token confidence establishes factual correctness. Source-shaped outputs can be stale, truncated, or noncanonical. The primary-exchange prompt is narrower than the task's full listing definition, although secondary-listing prompts seek additional exchanges. The death-city prompt can produce districts or broader localities that conservative normalization does not repair.

Hosted inference does not expose a verifiable immutable serving revision, so new calls need not reproduce archived responses. Exact reconstruction applies to the archived submission, not a guarantee of identical fresh generations. No private test labels were available for row-level error analysis.

10 Ethics and GenAI Disclosure

Generated facts can be wrong and should not be treated as an authoritative knowledge base without verification. The system processes public entity names and challenge files and makes no consequential decisions about individuals. Generative AI assistance was used for experiments, code, analysis, and drafting; all scores, counts, and artifact identities reported here were checked against the archived runs and official evaluator.

11 Conclusion

MuseList scores 0.6700 hidden-test macro-F1 using one 29.8B open-weight model and no external retrieval. It combines structured recall formats with relation-specific aggregation and numerical selection rules. Its development trace motivates further controlled comparisons of elicitation and selection. The released archive enables exact reconstruction of the scored submission and inspection of every retained transformation.

References

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-wise self-consistency for LLM probing. In *Joint Proceedings of KBC-LM and LM-KBC 2025*.

Dimitrios Alivanistos, Selene Báez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction. *arXiv preprint arXiv:2208.11057*.

Alex Clay, Ernesto Jiménez-Ruiz, and Pranava Madhyastha. 2025. Judge, generator, executioner: Utilizing an LLM for KBC. In *Joint Proceedings of KBC-LM and LM-KBC 2025*.

Shrestha Ghosh. 2023. Limits of zero-shot probing on object prediction. In *Joint Proceedings of KBC-LM and LM-KBC 2023*, pages 77–85.

Jingbo He and Simon Razniewski. 2025. Combining self-retrieval-augmented generation with divide-and-conquer for language model-based knowledge base construction. In *Joint Proceedings of KBC-LM and LM-KBC 2025*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? In *Transactions of the Association for Computational Linguistics*, volume 8, pages 423–438.

Jan-Christoph Kalo, Tuan-Phong Nguyen, Simon Razniewski, and Bohui Zhang. 2026. AKBC Shared Task 2026: Predicting complete knowledge base entries from language models. In *Proceedings of the 10th Workshop on Automated Knowledge Base Construction (AKBC) at EMNLP 2026*. Association for Computational Linguistics.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pretrained language models. In *Joint Proceedings of KBC-LM and LM-KBC 2025*.

LM-KBC 2026 Organizers. 2026a. AKBC Shared Task 2026: Dataset, relation definitions, evaluator, and baseline. `https://github.com/lm-kbc/dataset2026`. Post-disambiguation August release; accessed 2026-08-12.

LM-KBC 2026 Organizers. 2026b. AKBC Shared Task 2026: Final leaderboard. `https://www.akbc.ws/2026/shared-task.html`. Final leaderboard linked under Results. Accessed 2026-09-12.

Meta. 2026. Muse Glimmer 30B model card and weights. `https://huggingface.co/meta-models/Muse-Glimmer-30B`. Accessed 2026-08-12.

OpenRouter. 2026. Meta: Muse Glimmer 30B model metadata. `https://openrouter.ai/meta/muse-glimmer-30b`. Canonical slug meta/muse-glimmer-30b-20260810; accessed 2026-08-12.

Fabio Petroni, Tim Röcktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of EMNLP-IJCNLP*, pages 2463–2473.

Irene Mary Sam. 2025. Examining 4-bit quantized Qwen3-8B for object prediction @ LM-KBC 2025. In *Joint Proceedings of KBC-LM and LM-KBC 2025*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Appendix: Method and Reproduction Details

Companion to "MuseList: Confidence-Gated Multi-Surface Recall for Closed-Book Knowledge Base Construction"

1 Model Configuration and Evaluation

MuseList uses one approximately 29.8B-parameter open-weight model through OpenRouter: `meta/muse-glimmer-30b`, canonical slug `meta/muse-glimmer-30b-20260810`. The public repository `https://huggingface.co/meta-models/Muse-Glimmer-30B` reports 29,776,626,688 BF16 parameters and an Apache-2.0 license. The manifest records the inspected public revision, but the hosted service does not attest that it serves that immutable revision. DeepInfra, Together, and Fireworks are serving routes for this same model; provider fallback is disabled.

The August task release contains 477 training rows and 475 rows each for validation and test. The capacity counts are 97 and 98 respectively on the latter splits. The evaluator computes precision, recall, and F1 per row and then averages over rows. It normalizes strings and matches aliases by maximum bipartite matching; unmatched duplicate aliases can lower precision. Numerical values match within 5% relative tolerance. Empty predictions receive precision 1, and empty gold sets receive recall 1.

All fact-generation calls use only task inputs, permitted training examples, and model-generated intermediate records. No external source is retrieved. Exact subject–relation keys are excluded when selecting demonstrations for a target. The constraints, model-selection rationale, and final test metrics are reported in the main paper. Table 1 retains the more detailed development-validation breakdown.

Relation	Precision	Recall	F1
Award recipient	.3888	.2514	.2496
Stock exchange	.9050	.8850	.8333
Land borders	.9953	.9963	.9953
Area	.8600	.8500	.8500
Capacity	.4897	.3711	.3608
Death locality	.8300	.6000	.5500
All rows	.7970	.7153	.6916

Table 1: Development-validation results for the final configuration. This split was repeatedly observed during method development.

The sections below group operations by relation. Figures show the execution order, and the final traces follow complete rows through all relevant stages. Shared and core prompt templates are reproduced verbatim; the archive contains the exact materialized request for every call, including later-stage prompts. These archived requests, not a shortened prose description, define the precise model inputs.

2 Pipeline Diagrams

Figure 1 expands the paper's Figure 1 into the actual executable branches. Blue nodes are Muse calls; gray nodes are deterministic. Every call consumes only the subject, relation, frozen instructions, permitted released-train demonstrations, and same-run model outputs shown on its incoming edges. No source-looking string is opened.

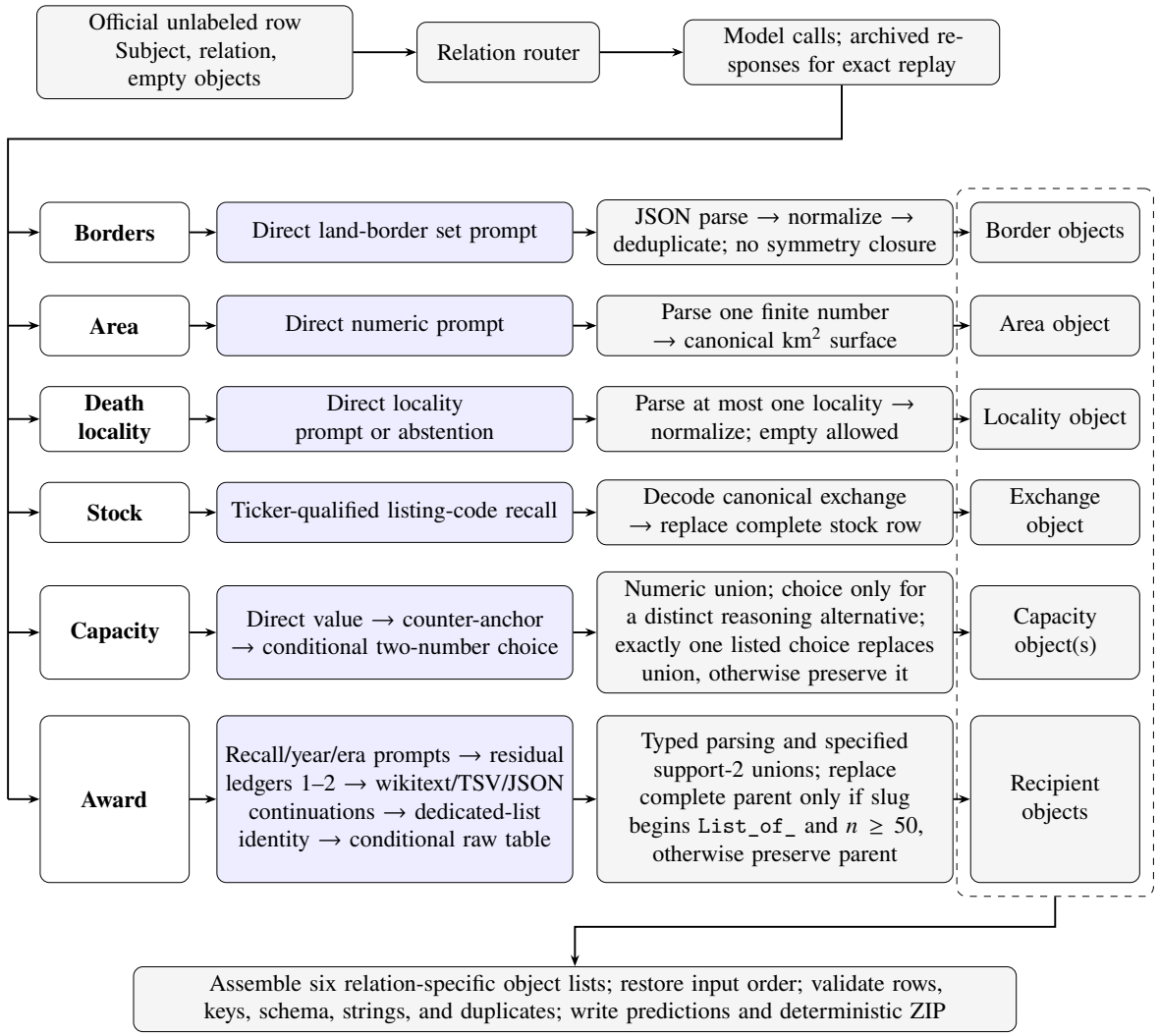


Figure 1: Expanded relation-routing graph. The six horizontal lanes expose the call families, parsers, relation policies, and common assembly. The multi-call capacity and award branches are expanded in Figure 2.

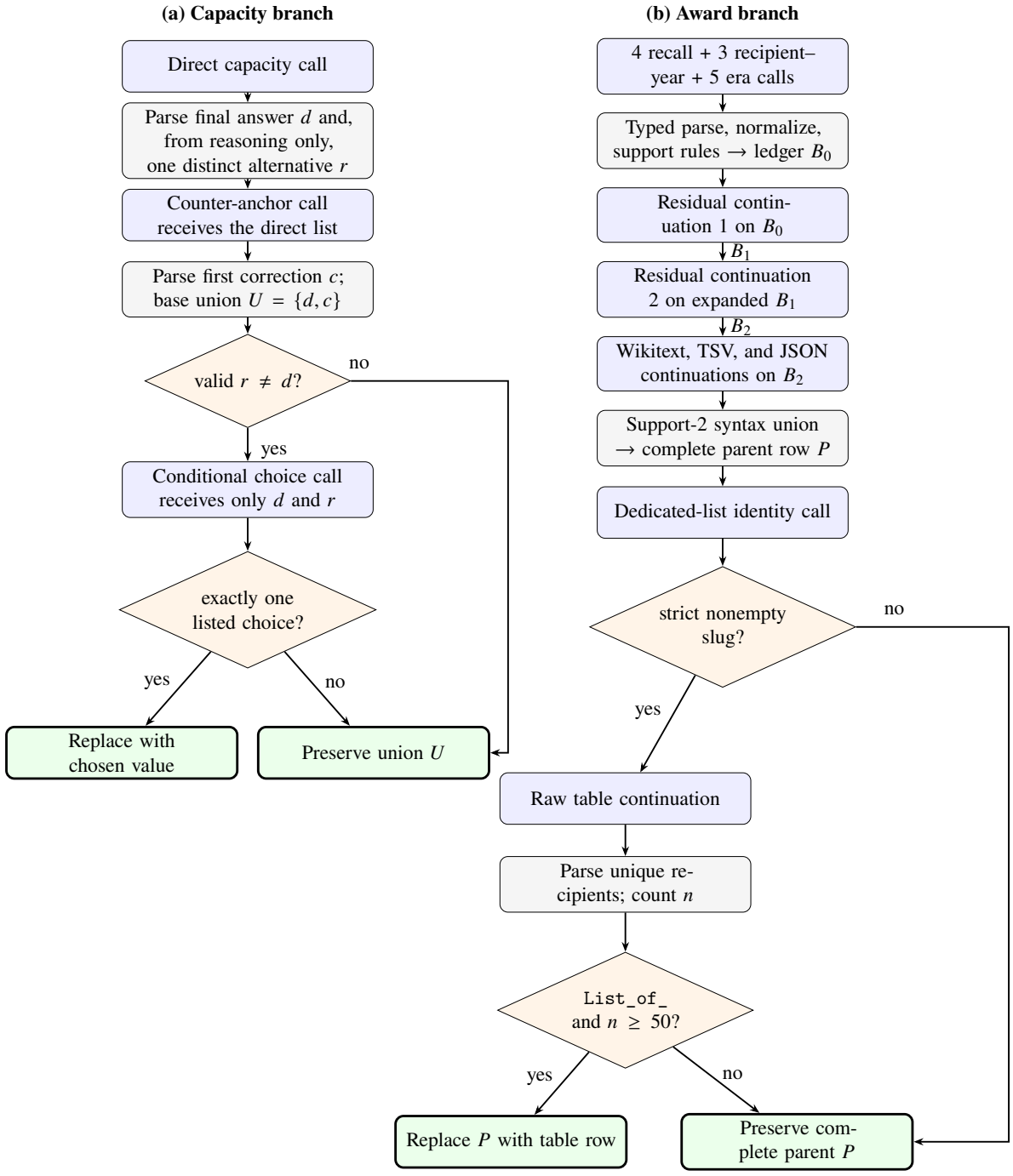


Figure 2: Branch-level expansion for the two multi-call relations. Blue boxes are Muse calls, gray boxes are deterministic transformations, diamonds are executable decisions, and green boxes are the only final outcomes. The capacity choice never receives the counter-anchor value; the award table call exists only after a valid identity result.

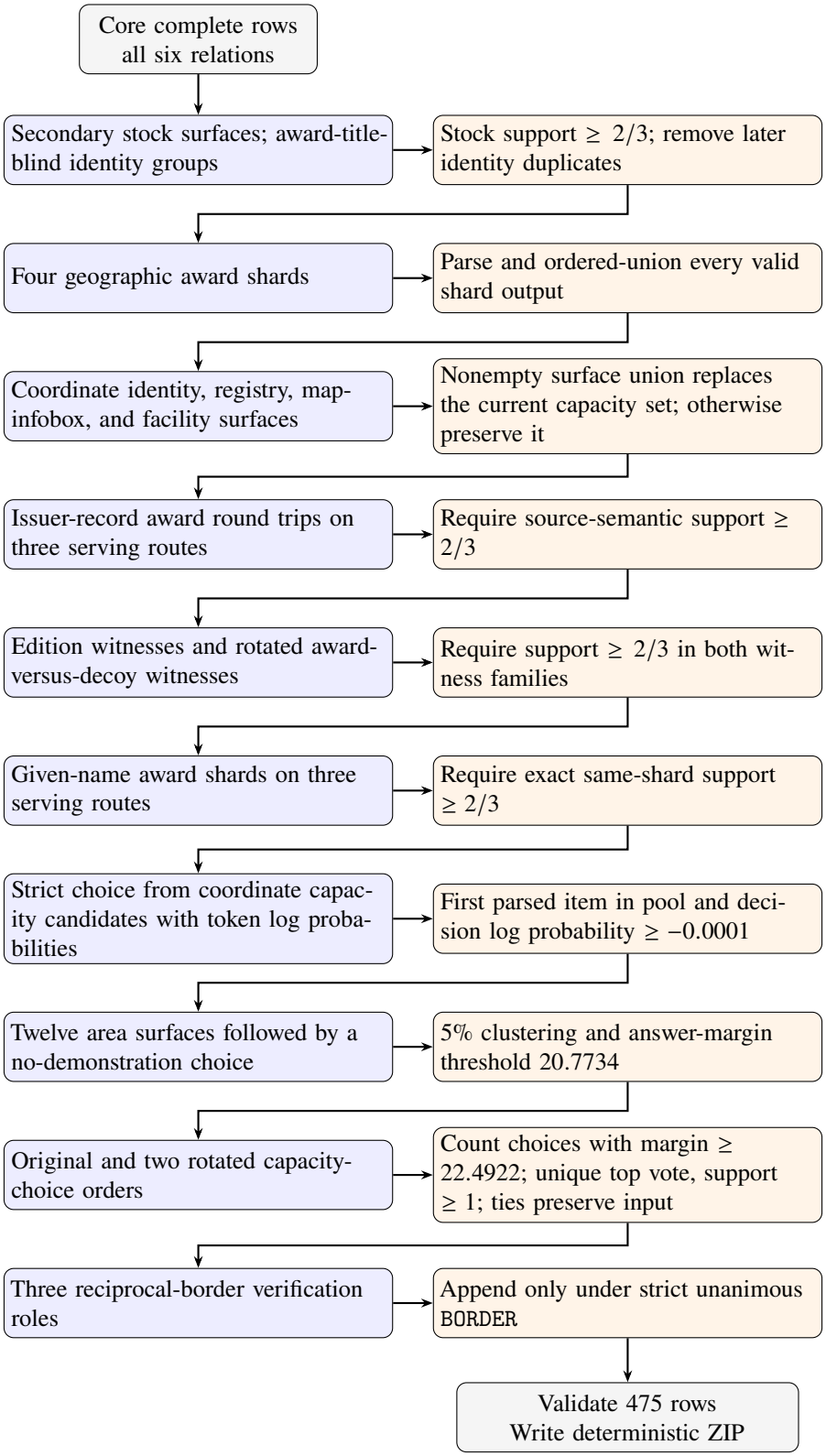


Figure 3: Expanded MuseList execution graph. Each blue call family uses Muse Glimmer; each orange box is deterministic and frozen from released train evidence before target transfer. A failed gate preserves its complete incoming row. This figure expands Figure 1 through every final call and acceptance layer.

3 Shared Prompting and Parsing

The shared direct prompt supplies initial candidates for borders, area, capacity, city of death, and an earlier stock route. The final stock route uses ticker-based generation instead. The direct-output examples below illustrate this initial stage only; they are not complete final-system traces.

3.1 Direct prompt and demonstrations

The system message is identical for the five direct relation paths:

```
You are a closed-book knowledge-base
↪ construction system. Use only your
internal knowledge. No web, no retrieval.
↪ Output valid JSON only, with this
exact schema: {"ObjectEntities": ["answer 1",
↪ "answer 2"]}. Use an empty
array when there is no answer. Do not include
↪ explanation.
```

The user message is constructed exactly as follows. Each `DEMO` is one single-line JSON object copied from released train.

```
Examples:
{DEMO_1}
{DEMO_2}
{DEMO_3}
{DEMO_4}
{DEMO_5}
SubjectEntity: {SUBJECT}
Relation: {RELATION}
Relation definition: {DEFINITION}
Question: {QUESTION}
```

Stock exchange. *Definition:* Return the stock exchange names where this company's shares are publicly traded. If the company is not separately listed, return an empty list.

Question template: Where do shares of {SUBJECT} trade?

Land borders. *Definition:* Return all currently recognized countries or comparable territories that share a land border with the subject. Exclude maritime borders. If there are none, return an empty list.

Question template: Which countries share a land border with {SUBJECT}?

Area. *Definition:* Return one number: total surface area in square kilometers. For countries use total area, including inland water.

Question template: What is the area of {SUBJECT} in square kilometers?

Capacity. *Definition:* Return one integer: the maximum spectator capacity of the venue. Use the highest published capacity if variants exist.

Question template: What is the total capacity of {SUBJECT}?

Death city. *Definition:* Return only the city where the person died. If the person is living or no city is known, return an empty list. Do not return a country or region.

Question template: In which city did {SUBJECT} die?

3.2 Concrete direct examples for every relation

These are materialized test requests, not handwritten illustrations. Stock target `TORM` receives demonstrations Farfetch → New York Stock Exchange; McColl's → London Stock Exchange; Bangladesh Cement Manufacturers Association → empty; West Japan Railway Company → Tokyo, Nagoya, and Fukuoka Stock Exchanges; and Chubu Electric Power → Tokyo and Nagoya Stock Exchanges.

Border target `Somalia` receives Sierra Leone → Guinea, Liberia; New Zealand → empty; Germany → its nine listed neighbors; Niger → its seven listed neighbors; and Liberia → Guinea, Ivory Coast, Sierra Leone. No reverse-border fact is added at this direct-generation stage; later verification is separate.

Area target `Niutao` receives Wellington Island → 5556; Mangareva → 15.4; Israel → 20770; Nantucket → 123.8; and Nigeria → 923768. The requested unit is square kilometers.

Capacity target `Pitt Stadium in Pennsylvania` receives Sungui Sports Complex → 35000; John A. Farrell Stadium → 7500; Estadio Revolucion Ciudad de Guatemala → 5000; Ibn-e-Qasim Bagh Stadium → 18000; and Nisa-Candybil Stadium → 1500. The archived direct answer is `"ObjectEntities":["78000"]`.

Death-city target `Ad Gerritsen` receives Harald Gerlach → Leimen; Marianna Rumjantseva → empty; Peter Eisenberg → Berlin; Hariton Pushwagner → Oslo; and Konrad Rufus Muller → Konigswinter. The prompt explicitly prohibits country or region answers.

The full Unicode-preserving strings and all five complete JSON demonstrations are stored in `archived/{split}/packs/nonaward.pack.jsonl`; the ASCII transliterations above are only for

compact typesetting.

3.3 Demonstration selection

For stock, borders, capacity, and awards, *source* selection takes the first five relation-specific train rows by released-file index after excluding the exact target key. Area and death city use *relevant-diverse* selection. For target t and candidate c,

$$s(t,c) = .35J(T_t,T_c) + .25J(Y_t,Y_c) + .25J(L_t,L_c) + .15J(G_t,G_c),$$

where T is the set of case-folded word tokens; Y is the subset matching a fixed type vocabulary (e.g., island, city, stadium, university); L is the token suffix after the first "in", "at", or "of"; G is the set of character trigrams; and J is Jaccard similarity. Candidates sort by descending score and then source index.

The selector first takes the highest-ranked candidate for each unseen output signature, then fills remaining slots in rank order. Numeric signatures are base-10 magnitude plus object count. Set signatures are object count. This uses labels only to diversify demonstrations, never to inspect a target answer. Selection sidecars record source index, subject, score, and signature for every request.

3.4 Parsing, normalization, and assembly

For every response, code takes the final answer segment, parses the requested JSON, and falls back to conservative list or numeric extraction only when needed. Empty markers are removed. String identity is Unicode decomposition, case-folding, punctuation removal, and whitespace collapse; the first display form is retained. Award support is counted once per frame per normalized name. Capacity numeric identity parses commas and numeric strings and formats the number to 12 significant digits before deduplication.

Rows are then restored to exact official input order and emitted with only the three schema fields `SubjectEntity`, `Relation`, and `ObjectEntities`. Validation checks row count, unique subject–relation keys, exact order, schema, string objects, and duplicate objects. The deterministic ZIP contains exactly one root member, `predictions.jsonl`, with fixed metadata and byte-identical content.

4 Stock Exchanges

The final stock answer combines ticker-based primary-exchange recall with secondary-listing proposals. The earlier direct/RDF route is described separately below because the replay retains it, not because its answers survive the whole-relation ticker replacement.

4.1 Primary-exchange recall

The ticker surface replaces the entire preceding stock relation. It makes one temperature-zero, 4,096-token call per stock row. Provider fallback is disabled. The system message is verbatim:

```
You recover a company's primary stock
↪  exchange through a closed-book
listing-code memory cue. Use internal model
↪  knowledge only; no tools,
browsing, retrieval, or inference merely from
↪  headquarters. Resolve the
exact company and distinguish an operating
↪  company from similarly named
funds or parents. Return valid JSON only as
{"ObjectEntities":["canonical stock
↪  exchange"]}, or an empty list if the
specific primary listing is not recalled. Do
↪  not return tickers, indices,
countries, or prose.
```

The user template is:

```
Exact company: {SubjectEntity}
Relation: companyTradesAtStockExchange
Memory route: Silently reconstruct the
↪  company's primary quoted ticker
exactly as an investor page writes it,
↪  including an exchange prefix or
suffix (for example EXCHANGE:TICKER or
↪  TICKER.SUFFIX). Decode only that
listing code into the canonical exchange
↪  name. Do not output the ticker.
Return one primary public stock exchange
↪  only. If the company is private,
defunct without a recalled listing, or
↪  ambiguous, abstain.
```

The parser accepts only the requested object list, normalizes exchange names, and writes the component for every stock row. There is no union, fallback, threshold, or validation-selected row mask. The released-train replacement raises stock F1 from 0.7446667 to 0.7550000; validation stock F1 rises from 0.7969048 to 0.8040000 and 21 rows change.

This exact prompt is intentionally preserved for submission reproducibility. It is narrower than the final organizer definition: it requests one primary exchange although multiple current listings and exchange-traded ADR/GDR/CDI venues count. The resulting recall loss is part of the scored method, not a reason to rewrite the archived prompt after evaluation.

4.2 Secondary listings

Three separately prompted roles recall secondary listings from annual-report, market-data, and security-master formats, excluding the generated primary exchange. A normalized exchange name must occur in at least two of the three outputs before being appended. The prompts are not external financial sources; all three responses are generated by Muse. Training-side stock F1 increases from .755 to .771 under this rule, and final validation stock F1 is .83333.

4.3 Earlier direct/RDF route retained in replay

Each stock row receives one additional closed-book source-form call. It uses no demonstrations and does not receive the direct answer. The exact system message is:

```
You complete a masked corporate source
↪ fragment using only memorized internal
model knowledge. Do not browse, retrieve, use
↪ tools, infer from headquarters,
or guess. Silently reconstruct the public
↪ stock exchange listing for the exact
company, then output valid JSON only as
↪ {"ObjectEntities":["exchange"]}.
Return canonical stock-exchange names, not
↪ ticker symbols, indices, countries,
parent-company listings, or prose. Multiple
↪ exchanges are allowed only when
independently remembered. Return an empty
↪ list if the company is private,
unlisted, a non-listed subsidiary, or the
↪ exchange is not precisely remembered.
```

The exact user template is:

```
SubjectEntity: {SUBJECT}
Relation: companyTradesAtStockExchange
Source form: rdf
Replace [MASK] internally, then return only
↪ the JSON answer.

# entity label: "{SUBJECT}"@en
wd:COMPANY wdt:P414 wd:[MASK] .
```

The RDF symbols are a memory-retrieval surface form, not a query: no endpoint, entity identifier, triple, or exchange candidate is supplied. After standard parsing and normalization, assembly applies exactly this row-level rule:

```
if direct_stock_objects is empty and
↪ rdf_stock_objects is nonempty:
    final_stock_objects = rdf_stock_objects
else:
    final_stock_objects = direct_stock_objects
```

Thus RDF output cannot modify, prune, or add to a nonempty direct stock answer. The rule was selected and frozen on released train before validation inference.

5 Award Recipients

Award completion has three purposes: elicit an initial roster, recover omissions using different output formats and partitions, and apply prescribed support and duplicate-removal rules. Some additions require repeated support; others use a union of all syntactically valid outputs. Those policies are specified separately below.

5.1 Four long-precise frames

Each award first uses the same five source-order award demonstrations and this system message:

```
You construct closed-book KB entries for
↪ award-recipient lists. The goal is
high recall without generic famous-name
↪ hallucination. Output bare recipient
entity names only, as JSON. Output valid JSON
↪ only, with this exact schema:
{"ObjectEntities": ["answer 1", "answer 2"]}.
↪ Use an empty array when there
is no answer. Do not include explanation.
```

After the examples, all four user messages contain the exact award, relation definition, and these hard rules:

```
Hard rules:
1. Include only recipients of this exact
↪ award title.
2. Do not include nominees, presenters,
↪ jurors, honorary mentions, related
   awards, predecessor awards, successor
   ↪ awards, countries, years, or
   explanations.
3. Do not pad the list with generally famous
↪ people. If the subject is an
   honorary degree or membership, famous
   ↪ activists and politicians are wrong
   unless you remember this exact institution
   ↪ awarded them.
4. Use one canonical name per recipient; no
↪ duplicate aliases, parentheticals,
   dates, or categories.
5. If a recipient is a work, album,
↪ organization, or team, output that entity
   rather than an associated person.
Question: Which entities have won or received
↪ {SUBJECT}?
```

The four instruction frames are: (1) recall the official recipient list chronologically and return up to 120 confident names; (2) focus on exact-award winners from authoritative winner-list memory and skip generic celebrities; (3) for honorary degrees or fellowships, require memory of the exact institution; and (4) for prizes or medals, include only exact-award membership. Names are normalized per Section 3.4; a name must occur in at least two of the four outputs.

Concrete example: for test subject `Max Planck Medal`, the five demonstrations are Nobel Prize in Physiology or Medicine, Premier League Player of the Season, FAI Gold Air Medal, Fulbright Prize, and Ballon d'Or, each with its released recipient list. The first frame begins "Recall the official recipient list chronologically" and asks for up to 120 names. The complete large prompt is preserved in `award_long_precise_4frame.pack.jsonl` rather than abbreviated.

5.2 Recipient-year evidence calls

Three further calls do not see validation/test answers or external facts. They ask the same model to make each proposed recipient carry an internally recalled year. The system message is:

```
You are a closed-book knowledge-base
↪ construction system.
Use only internal model knowledge. Do not use
↪ tools, web search, retrieval,
external corpora, or knowledge-base lookup.
Return valid JSON only with this schema:
{"Candidates":[{"recipient":"canonical
↪ recipient name","year":2000}]}
Use an empty Candidates array when no exact
↪ recipient-year pair is remembered.
Do not explain.
```

The user asks for up to 50 direct recipients of the exact award, paired with a four-digit year, excluding nominees, presenters, committee members, related awards, predecessor/successor awards, uncertain years, and duplicates. The three pass instructions are verbatim:

```
Recall from a chronological official-winners
↪ list.
Recall only recipient-year pairs for which
↪ both fields are specifically
remembered.
Independently reconstruct the precise
↪ recipient-year list and omit doubtful
pairs.
```

Parsing accepts only a nonempty recipient and integer year in 1000–2100. The year is then discarded. All accepted normalized recipient names, i.e. support at least one of three, are unioned with the support-two long-precise result. For `Max Planck Medal`, the first request ID is `00001-000-awardWonBy-Max Planck Medal-year-anchor`. This is evidence conditioning, not year verification against a database.

5.3 Disjoint temporal enumeration

Five additional calls partition the search space rather than repeat the same question. Their system message is reproduced verbatim:

```
You construct closed-book award-recipient
↪ records using only internal model
knowledge. Do not browse, retrieve, call
↪ tools, use external corpora, or query
a knowledge base. Output valid JSON only as
{"Candidates":[{"recipient":"canonical
↪ name","year":2000}]}.
Use an empty Candidates array when no exact
↪ pair is remembered.
```

The user template is:

```
Exact award: {SUBJECT}
Time partition: {ERA}
Enumerate direct recipients whose award year
↪ lies only in this partition. The
time partition is a search cue, not evidence.
↪ Include a recipient only when
both exact-award membership and a four-digit
↪ year are remembered. Exclude
nominees, presenters, jurors, related awards,
↪ predecessor or successor awards,
countries, explanations, and duplicate
↪ aliases. Return up to 80 recipient-year
pairs. JSON only.
```

The five literal substitutions for `ERA`, in fixed order, are `before 1950`, `from 1950 through 1979`, `from 1980 through 1999`, `from 2000 through 2009`, and `from 2010 through the present`. Parsing requires a nonempty recipient and an integer year in 1000–2100. The assembler does not separately re-check the prompt's era boundary. The year is discarded, every syntactically valid name from the five calls is normalized and unioned, and that component is unioned with the complete preceding award result. No support threshold, era selection, or row fallback is applied.

For test subject `Max Planck Medal`, the pre-1950 call returns seven pairs, beginning Max Planck (1929), Albert Einstein (1930), and Max von Laue (1931); the other four archived calls return empty candidate lists. Thus only the seven valid names from the first partition can enter this row's era component. The years are model outputs, not verified or supplied facts.

5.4 Residual list continuations

The first pass starts from the complete award candidate set after stock replacement. The second repeats the same operation from the expanded first-pass candidates. Each split uses ten temperature-zero, 4,096-token calls. The system and user templates are:

```
SYSTEM: Continue the assistant's unfinished
↪ award-recipient archive from
internal memory only. No browsing, tools,
↪ search, explanation, nominees,
```

```
presenters, jurors, related awards, or
↪  placeholders. The bullets already
written are candidate recipients and must not
↪  be repeated. Continue with
only additional entities that directly
↪  received this exact named award.
Stop when no further distinct recipient is
↪  strongly remembered.

USER: Exact award: {SubjectEntity}
Continue the existing assistant-side ledger
↪  at its final empty bullet.
Do not restart or repeat earlier entries.

ASSISTANT PREFILL:
== {SubjectEntity}: Recipients ==
* {existing recipient 1}
...
*
```

Only text after the final empty bullet is parsed. Bullet markers are removed; blank, malformed, repeated, and normalized duplicate values are rejected. All distinct residual names are unioned with the parent row. The first pass raises released-train award F1 from 0.5528621 to 0.5573066 and validation award F1 from 0.2052748 to 0.2246002. The second raises train F1 to 0.5684942 and validation F1 to 0.2318209. No output is selected or reverted row by row.

5.5 Agreement across three output formats

This stage makes three temperature-zero, 4,096-token calls per award row with minimal reasoning effort. All variants share this system message:

```
Continue the assistant's unfinished award
↪  archive from internal memory
only. No browsing, tools, search,
↪  explanation, nominees, presenters,
jurors, related awards, or invented
↪  placeholders. Resolve the exact named
award and emit only entities that directly
↪  received it. Recipients may be
people, organizations, teams, or works.
↪  Continue the supplied syntax
rather than restarting it.
```

The three verbatim user/prefill forms are:

```
WIKITEXT USER: Exact award: {SubjectEntity}
Relation: awardWonBy
Continue the unfinished recipients section
↪  with one direct recipient per
bullet. Do not emit years.
PREFILL: == {SubjectEntity}: Recipients ==
*

TSV USER: Exact award: {SubjectEntity}
Relation: awardWonBy
Continue the unfinished archival table with
↪  one award year and direct
recipient per row, tab-separated. Use the
↪  exact award's complete remembered
history.
PREFILL: award<TAB>year<TAB>recipient
{SubjectEntity}<TAB>

JSON USER: Exact award: {SubjectEntity}
Relation: awardWonBy
Continue the unfinished JSON array with
↪  direct recipient names only.
Produce valid JSON and close the array.
PREFILL: {"award": "{SubjectEntity}",
↪  "recipients": ["
```

Each syntax is parsed independently. A normalized name contributes at most one vote per syntax; names with support at least two are unioned with the second residual-pass parent. The threshold was selected on released train, where award F1 rises from 0.5684942 to 0.5700759. Validation award F1 rises from 0.2318209 to 0.2319206.

5.6 Recipient-list identity and table generation

This roster-generation stage adds no non-award calls. Identity uses temperature 0 with provider fallback disabled and this exact prompt pair. The test call ceiling is 768 tokens; validation uses 1,536 after the same frozen prompt was rerun to correct reasoning-budget exhaustion. No top-p or reasoning-effort override is sent.

```
SYSTEM: Resolve the exact English Wikipedia
↪  article whose primary content
is the chronological list, table, or roll of
↪  direct recipients of this
exact award. Work from internal memory only
↪  and do not name recipients.
Return JSON only:
↪  {"list_article_slug":string|null,
"table_section":string|null}. Use the exact
↪  title slug after /wiki/ with
underscores. Do not return the general award
↪  article unless its primary page
itself contains the recipient table. If no
↪  exact list surface is specifically
remembered, use null. No browsing, tools,
↪  guessed titles, placeholders, or
prose.
USER: Exact award: {SubjectEntity}
```

For a strict nonempty slug, ledger continuation uses 8,192 tokens and:

```
SYSTEM: Continue the unfinished raw-wikitable
↪  recipient roll from internal
pretrained memory of the identified list
↪  article. Emit chronological rows
only as `year || direct recipient`, one
↪  recipient per row. Continue as many
specifically remembered rows as possible.
↪  Exclude nominees, teams unless
directly awarded, related awards, prose,
↪  headings, guesses, browsing, and
tools.
USER: Continue only table rows.
ASSISTANT PREFILL:
```

```
SOURCE: https://en.wikipedia.org/wiki/{slug}
RAW WIKITABLE EXPORT FOR EXACT AWARD:
↪ {SubjectEntity}
|-
| year || recipient
|-
|
```

This is the deterministic traversal used in the final system. On the test split, the frozen independently train-positive forward traversal is used. Its system message says "Begin with the oldest specifically remembered edition and work forward." Its assistant prefix places `ORDER: CHRONOLOGICAL (OLDEST TO NEWEST)` immediately after the `RAW WIKITABLE EXPORT FOR EXACT AWARD:` heading and before the table rows. That variant improved released-train award F1 to 0.5732859, was frozen, and was run on validation before test; its validation run was a no-op because Turing Award yielded 46 parsed names, below the unchanged 50-name gate. Thus the gate and source-surface method are shared, but the two scored split artifacts do not use a byte-identical ledger prompt. Both exact prompt packs are archived and the fresh driver pins the split-specific choice explicitly.

The URL-shaped string is never dereferenced. Slug parsing requires one JSON object and a conservative character grammar. Ledger parsing accepts only well-formed rows with integer year 1000–2100 and nonempty recipient. The parent row is replaced iff the slug begins `List_of_` and at least 50 unique normalized recipients parse. Otherwise the complete multi-syntax row survives. These predicates are target-label-independent after train-side selection. The 50-name cutoff is a substantial-output gate; it does not prove that the list is complete, factually correct, or longer than its parent.

Examples of replacement and preservation. For validation Turing Award, identity returns `List_of_Turing_Award_laureates`. Strict ledger parsing yields 78 unique recipients. Both gate predicates pass, so the 99-name multi-syntax row is replaced. For Presidential Medal of Freedom, identity returns `List_of_Presidential_Medal_of_Freedom_recipients`, but only nine names parse; the parent count of 120 is preserved. For test Nobel Prize in Literature, identity returns `List_of_Nobel_laureates_in_Literature`, 119 names parse, and the 127-name parent is replaced. Thus success, short-ledger fallback, and null/invalid-slug fallback are all executable paths.

5.7 Identity grouping and partitioned recall

Award identity pruning. *Candidate construction:* Award-title-blind within-row identity grouping

Frozen acceptance and effect: Keep the lowest-index surface in every duplicate group

Geographic award shards. *Candidate construction:* Europe; Americas; Asia/Oceania; Africa/stateless/multinational

Frozen acceptance and effect: Support-one union of every valid parsed name from the four disjoint shards

Given-name award shards. *Candidate construction:* The same first-token shard through three pinned serving routes

Frozen acceptance and effect: Add only exact same-shard support $\geq 2/3$

5.8 Issuer-record and two-family checks

Issuer-record checks ask the model to associate a candidate with an award edition and a source identifier, then reconstruct the recipient. A response qualifies only if its verification field is true, its reverse-recipient name normalizes to the candidate, and its edition/source fields pass the fixed issuer-source predicate. At least two of three serving-route responses must qualify. This is consistency checking of generated fields, not opening or verifying an external source. Conservative identity matching prevents adding a name already represented in the row.

The two-family check uses three edition prompts and three reverse award-choice prompts. The edition family requires at least two received=true responses agreeing on a parseable edition. The reverse family requires at least two choices of the target award among the supplied alternatives. A candidate is added only if both families pass and it is not an existing identity. These separate prompt families may still share model errors.

6 Venue Capacity

The initial capacity route forms a numerical answer set and may reduce it to one value. Coordinate-conditioned generation can then replace that set. Later confidence-based choices operate on a broader candidate pool. Keeping these stages separate is important: rejection preserves the immediate input, which may differ from the original direct answer.

6.1 Correction of the initial draft

The direct capacity list is passed as an explicitly untrusted draft to one more call of the same model. The system message is:

```
You perform closed-book factual error
↪  correction using only internal model
knowledge. Do not browse, retrieve, call
↪  tools, or estimate. Output valid JSON
only as {"ObjectEntities":["candidate
↪  1","candidate 2"]}. Return at most
three distinct numeric candidates in
↪  descending confidence, or an empty list
if no exact value is remembered.
```

The user template is:

```
SubjectEntity: {SUBJECT}
Relation: hasCapacity
Untrusted draft candidates: {BASE_JSON}
Audit frame: Treat the draft values as
↪  potentially misleading anchors.
Independently recall the exact published
↪  fact, then correct the draft.
Return normal spectator/event capacity as a
↪  bare number, not pitch area,
attendance, or temporary expansion.
The draft is evidence of what another attempt
↪  guessed, not a fact. Return only
corrected JSON.
```

Only the first parsed correction value is retained. It is unioned with the base under numeric identity; no value is multiplied, rounded, range-expanded, or selected using validation labels. Concrete example: Pitt Stadium's base draft is `["78000"]`; the correction call returns `"ObjectEntities":["78000"]`, so the final set remains one value. A changed correction would produce two candidates, trading precision for recall.

6.2 Choice between direct and reasoning candidates

Capacity self-adjudication adds one same-model call only when the saved direct response exposes two distinct candidates. Candidate extraction is deterministic. From the final answer segment, parse the first numeric value as `direct`. From the provider-returned reasoning, discard everything before the final literal `Question:` marker so numbers in demonstrations cannot become candidates. Recognize positive values from 100 through 500,000 using either 3–6 digits or comma-grouped digits. Exclude numeric identity equal to `direct`. Rank remaining values by descending mention count and then descending final mention position. Retain only the first as `alternative`. A row is eligible if `direct` and `alternative` are both present and distinct.

The adjudication system message is exactly:

```
You construct closed-book numeric KB entries
↪  for venues. Choose only from the
provided candidate capacity numbers. Do not
↪  use web, retrieval, databases,
tools, or external lookup. Output valid JSON
↪  only with schema
{"ObjectEntities": ["candidate"]}. Use an
↪  empty array if none of the
candidates is likely to be the official or
↪  commonly listed capacity.
```

The user template is:

```
SubjectEntity: {SUBJECT}
Relation: hasCapacity
Definition: maximum spectator or event
↪  capacity of the exact venue/facility.
Candidate capacity numbers from the current
↪  system: {CANDIDATE_JSON}
Task:
1. Identify the exact venue named by the
↪  subject, using location text only for
   disambiguation.
2. Select at most one candidate number that
↪  is likely to be the official or
   commonly listed capacity.
3. Return [] if the candidates look like
↪  generic guesses, city populations,
   years, areas, or if you are unsure.
4. Do not return any number that is not in
↪  the candidate list.
```

There is one call per eligible row at temperature 0, top-p 1, low reasoning effort, and 512 maximum tokens. Parsing normalizes the returned values and matches them against the two listed values. Exactly one distinct listed match becomes the adjudication component. Zero matches, an empty response, an unlisted value, or two listed values is an abstention. The frozen *fallback-nonempty* rule is:

```
if adjudication_component contains exactly
↪  one listed candidate:
    final_capacity_objects =
    ↪  adjudication_component
else:
    final_capacity_objects =
    ↪  pre_adjudication_capacity_objects
```

The call does not receive the counter-anchor output, validation gold, a retrieved value, or an external candidate vocabulary. Both candidates were already generated inside the same direct Muse call.

For validation subject `St Andrew's in Birmingham`, the direct answer is `29409` and the top distinct reasoning mention is `30079`. The pre-adjudication union contains both numbers; the frozen choice call returns only `30079`. The released validation value is `30016`, so both old

numbers fall within the evaluator's 5% tolerance but the second false-positive slot lowers precision. Selecting one value improves this row. This example was not used to define the rule: on all released-train capacity rows, the policy changed 20 rows and raised F1 from 0.23667 to 0.25667 before validation inference. The frozen validation transfer made 70 calls, changed six complete rows, and raised capacity F1 from 0.30241 to 0.30584. Test has 67 eligible rows and ten replacements. Gold answers and row-level correctness remain private; Codabench reports aggregate capacity precision/recall/F1 .3980/.2449/.2449 for the final submission.

6.3 Coordinate-conditioned generation

The model first generates coordinate information and then recalls capacity values in venue-record formats. The surface-union transformation parses at most the first value from each response variant, normalizes and deduplicates those values, and replaces the current row if this union is nonempty. Otherwise it retains the current row. This is a prediction transformation, not merely construction of an unused candidate pool.

The later choice prompts use a broader thirteen-source numerical pool, combining the specified capacity sources. Pool generation and row replacement are distinct operations. The complete source packs and their ordering are recorded in the archive.

6.4 Confidence and candidate-order selection

Capacity confidence. *Candidate construction:* The prompt requests one choice from the supplied coordinate pool. The parser takes the first parsed item, requires it to be in the pool, and ignores any additional items. This parser is also used for area and order-varied capacity selection.

Frozen acceptance and effect: Replace with that one item only when the final nonempty-list decision token has log probability ≥ -0.0001

Order-varied capacity. *Candidate construction:* Original candidate order and two deterministic rotations

Frozen acceptance and effect: Count only in-pool choices with margin $\geq$ 22.4921875 across original/rotated orders. Accept the uniquely most-supported value with support ≥ 1; a tie or no qualifying vote preserves the input. One qualifying vote suffices; cross-order agreement is not required.

For each token overlapping the selected answer, the margin is the chosen token's log probability minus the largest alternative log probability returned in the provider's top-token list. Only finite margins are averaged. A missing usable feature cannot pass the corresponding threshold. The selectors use Fireworks with temperature zero, provider fallback disabled, and completion log probabilities enabled. A confident model answer can still be factually wrong.

7 Geographic Area

Twelve-format area. *Candidate construction:* Temporal, geometry, gazetteer/prefill, Wikipedia-like, native-unit, coordinate, and core numeric surfaces

Frozen acceptance and effect: Cluster at 5%; use a strict in-pool, no-demonstration choice only at `answer_mean_margin` $\geq$ 20.77343735098843

The twelve representations are distinct prompting formats, not statistically independent measurements. Numerical candidates are generated from recalled temporal, geometric, gazetteer, Wikipedia-like, native-unit, coordinate, and direct-answer records. The model selects a listed candidate without demonstrations; the fixed answer-margin rule determines whether it replaces the incoming answer. A rejected choice preserves that answer. The same token-margin definition and pinned serving configuration described for capacity apply here. Exact prompt texts and candidate ordering are preserved in the request packs.

8 Land Borders and City of Death

Borders begin with the shared direct set prompt. For every generated edge $A \rightarrow B$, a reverse proposal $B \rightarrow A$ is eligible only when both countries are task subjects and that exact reverse string is absent. Three fixed Muse roles—political-boundary editor, physical cartographer, and legal-geography auditor—return strict BORDER or NOT_BORDER decisions. Only three BORDER decisions append the proposed neighbor; otherwise the row is preserved. The calls use Fireworks, temperature zero, and disabled provider fallback. Exact-string absence is not entity-level absence: an accepted name can duplicate an existing entity under a different alias.

City of death uses only the shared direct locality-or-abstain prompt and conservative parsing. No additional completion or verification layer changes

this relation. A returned district or broader locality is not automatically converted into the correct city.

9 Development Protocol and Call Accounting

Each deployed policy was specified and checked on released train before its target-split execution. Target rows were excluded from demonstrations where relevant, and no validation row was accepted, reverted, or repaired individually. Complete-system validation was observed repeatedly, so the main paper's cumulative trace documents development rather than estimating isolated causal effects.

Train-neutral or leave-one-out-negative alternatives were not transferred. Rejected experiments are not required by, and do not enter, the final replay. Fine-grained train diagnostics remain available in the public run reports but are omitted here because they are experiment history rather than part of the method specification.

9.1 Training-side evidence

Secondary stock support improves released-train stock F1 from .755 to .771. The first capacity confidence threshold improves the 69 coordinate-matched rows under strict leave-one-row-out evaluation from .25604 to .26087 and is selected in 68/69 folds. The area margin rule improves strict leave-one-row-out area F1 from .81 to .82 and is selected in 99/100 folds. The order-varied capacity rule reaches .27536 in leave-one-row-out evaluation and is selected in 68/69 folds. The border verifier produces 24 true positives, no false positives, eight true negatives, and no false negatives on released train. Neither validation gold nor evaluator output appears in a model payload.

Validation scores were nevertheless observed after complete frozen transformations, so development is adaptive. Source-consensus components are associated with the first cumulative validation change; numeric confidence and order gates with the largest later changes; and reciprocal verification with the final .691599 value. This trajectory is development evidence, not a held-out estimate or an estimate of isolated causal effects.

9.2 Core inference settings

Core direct and four-frame award calls use temperature 0, top-p 1, reasoning effort `low`, and maximum 4,096 non-award or 8,192 award tokens. Award-year calls use temperature 0.35, top-p 0.95, and 8,192 tokens. Counter-anchor calls use temperature 0, top-p 1, and a 4,096-token API ceiling. Award-era calls use temperature 0.2, top-p 1, reasoning effort `low`, and an 8,192-token API ceiling. Stock RDF calls use temperature 0, top-p 1, reasoning effort `low`, and a 512-token API ceiling. Each request has a 360-second timeout and two retries; failed coverage is reconciled and retried once. Assembly requires exactly one response for every materialized request. Capacity self-adjudication also uses temperature 0, top-p 1, reasoning effort `low`, and a 512-token ceiling.

9.3 Call coverage and runtime

The retained core rows use 817 validation and 811 test calls: awards 185/181, stock 100/100, borders 68/67, area 100/100, capacity 264/263, and city of death 100/100. The earlier direct-generation stage instead makes 852/850 calls, including 200 stock calls later superseded by 100 ticker-based calls and only 120 initial award calls before continuations. These totals refer to different stages; neither is a whole-system cost estimate.

Split	Calls	Tries	Sum (s)	Max (s)
Val.	817	847	4,753.005	116.198
Test	811	840	4,397.442	34.608

These retained-core figures are an archive audit, preserved as `execution_accounting.json`; they are not regenerated by the prediction replay's final PASS assertion. The latency column sums per-request elapsed times and is not wall-clock runtime because calls were concurrent. Archived response objects do not contain provider billing token counts or charged amounts, so an exact historical cost cannot be reconstructed without the private account ledger. Token ceilings alone do not determine actual billing.

The verification and set-completion layers comprise 2,917 successful test calls: 310 for secondary listings and identity pruning, 40 geographic award shards, 71 coordinate capacity surfaces, 143 source-semantic award checks, 786 dual-witness award checks, 120 given-name checks, 17 capacity-confidence choices, 1,375 area surface and selector calls, 34 rotated capacity choices, and 21 reciprocal border checks. DeepInfra, Together, and Fireworks are serving routes for the same Muse model. Response metadata confirms Muse for every archived call; an obsolete exporter label in one

area request pack is overridden by the public fresh-call driver and did not determine the served model.

10 Complete Final-System Row Traces

All six examples start at the official test input: a subject, a relation, and an empty `ObjectEntities` placeholder, not an intermediate prediction. They trace archived model outputs and frozen transformations; none asserts correctness against hidden test labels. Source-shaped prompts are closed-book model calls, not retrieved records. Components targeting other relations leave the row unchanged. Complete ordered arrays are in `dataset2026/reproduction/muse_glimmer30b_improved_candidate/checkpoints/test/10_reciprocal_border_verification.jsonl`; the line numbers below refer to that file and the official test input. The companion `TRACE_EVIDENCE.md` locates every preceding stage.

10.1 Award recipients: Sydney Peace Prize

The input is `Sydney Peace Prize`, `awardWonBy` (line 205). All four long-precise recall frames and all three recipient–year calls parse empty. Of five temporal calls, only the final era supplies a parsed name, Adolfo Pérez Esquivel; its generated year is not a gold fact. The first residual continuation adds nothing; the second appends Mairead Maguire and Malala Yousafzai to this one-name row. Three-syntax consensus adds nothing: the wikitext and TSV lists have no common name, and the malformed JSON continuation parses as fragments rather than its apparent recipient list. The dedicated-list identity response supplies no qualifying slug, so no table replacement occurs and the three-name row survives.

Within-row identity grouping returns no duplicate groups. Geographic shards then add Desmond Tutu from the Africa/stateless shard under support-one union; the other three shards add nothing. Source-semantic consensus yields no qualifying addition for this award. The dual-witness gate also adds nothing: for example, the candidate The International Campaign to Abolish Nuclear Weapons passes edition consensus for the generated year 2017 but has only one supporting reverse-title witness, below two. Given-name shard consensus adds nothing. Later numeric and border stages preserve the entire ordered four-name output:

> [Adolfo Pérez Esquivel, Mairead Maguire, Malala Yousafzai, Desmond Tutu].

This is the complete emitted row, not merely the last additions or a claim that these four names exhaust the real award's recipients.

10.2 Stock exchange: TORM

The input is `TORM`, `companyTradesAtStockExchange` (line 209). The five-demonstration direct answer parses to [Nasdaq Copenhagen]. An archived RDF-shaped answer also gives Nasdaq Copenhagen, but its empty-base-only fallback cannot change this nonempty row. Whole-relation ticker-surface replacement independently returns the same singleton. The three secondary-listing prompts exclude Nasdaq Copenhagen: annual-report returns an empty list, while market-data and security-master each return New York Stock Exchange. Normalized support is therefore exactly $2/3$, meeting the frozen append threshold. No later component targets stock; the complete final output is [Nasdaq Copenhagen, New York Stock Exchange].

10.3 Land borders: Zambia

The input is `Zambia`, `countryLandBordersCountry` (line 373). The five-demonstration direct call yields the ordered list [Angola, Botswana, Democratic Republic of the Congo, Malawi, Namibia, Tanzania, Zimbabwe]. Parsing retains all seven surfaces, and every stage before reciprocal verification preserves this row. The generated row for Mozambique (line 311) already contains Zambia; both countries are input subjects, and the reverse surface is missing from Zambia's row. This generates the candidate Zambia–Mozambique without consulting labels. Political-boundary editor, physical cartographer, and legal-geography auditor each return the strict JSON decision `BORDER`. Unanimous $3/3$ acceptance appends Mozambique, giving the complete final array [Angola, Botswana, Democratic Republic of the Congo, Malawi, Namibia, Tanzania, Zimbabwe, Mozambique].

10.4 Area: Niutao

The input is `Niutao`, `hasArea` (line 1). The target-excluded five-demonstration direct call returns `["2.5"]`; all stages before the area selector preserve it. In the twelve-source pool, temporal and

shoreline components also supply 2.5; bounding-box reconstruction supplies 2.07; survey-ledger, infobox prefill, and coordinate-area components supply 2.99. Gazetteer, Wikipedia-like, and all three native-unit components contribute no parsed value; the core row supplies 2.5. The frozen 5% clustering therefore gives the complete ordered pool `["2.5","2.07","2.99"]`. The no-demonstration, in-pool choice returns `["2.99"]` with `answer_mean_margin` 21.5625, exceeding 20.77343735098843. It replaces 2.5. Subsequent capacity and border gates do not target this relation, leaving the final array `["2.99"]` in square kilometres. The two unselected numeric alternatives are not unioned into the final answer.

10.5 Capacity: Pitt Stadium in Pennsylvania

The input is `Pitt Stadium in Pennsylvania`, `hasCapacity` (line 101). Direct recall returns `["78000"]`; the counter-anchor call repeats 78000, so their numeric union remains a singleton. The direct reasoning also mentions 56000. The conditional self-chooser receives `["78000","56000"]` and selects 78000, preserving the core row. Later coordinate identity returns (40.4444, –79.9569); registry, map-linked infobox, and geocoded-facility calls each return 56000. Their support-one surface union is `["56000"]` and *replaces*, rather than extends, the earlier row.

The separate thirteen-source selector pool is `["78000","56000","35000","1925","1999","10000","5000","20000","10300"]`. It includes reasoning-derived and prior candidates, not just the three coordinate answers; even the year-like values are preserved as archived. The coordinate-conditioned chooser returns 78000 with decision-token log probability approximately $-4.76837 \times 10^{-7} \geq -0.0001$, so the confidence gate changes 56000 back to 78000. Original order and two rotations then select 78000, 56000, and 78000, with answer margins 22.78125, 23.1875, and 23.1875. All exceed 22.4921875: 78000 wins by two qualifying votes to one and the order-varied gate leaves the row unchanged. Border verification does not target capacity. Final output: `["78000"]`.

10.6 Death locality: Ad Gerritsen

The input is `Ad Gerritsen`, `personHasCityOfDeath` (line 376). The target-excluded selector supplies demonstrations for Harald Gerlach, Marianna Rumjantseva, Peter Eisenberg, Hariton Pushwagner, and Konrad Rufus Müller. The direct city-or-abstain response is `{"ObjectEntities": ["Amsterdam"]}`. Parsing retains this single locality; normalization changes neither its spelling nor cardinality. There is no death-specific secondary gate in the final pipeline. Every subsequent checkpoint preserves it, yielding the complete final array `["Amsterdam"]`. This is a test prediction, not a gold-verified death-place assertion.

11 Reproducing the Submission

11.1 Exact final-system rebuild

From a clean checkout, one command first bootstraps the four pinned Python packages if needed, then executes the complete final pipeline from archived model records and verified deterministic components, validates the official test schema and order, and creates the exact submitted ZIP:

```
bash reproduce_improved_submission.sh
↪  /tmp/muselist-improved-replay
```

An independent clean temporary-directory replay produced the submitted 475-row prediction file and one-member ZIP byte-for-byte. The command exits nonzero on any artifact, schema, order, or package mismatch. The machine-readable manifest maps every semantic component to its handoff, prompt pack, raw responses, input, deterministic transformer, output, and evaluator report. Codabench submission 890031 evaluated this frozen package and reports displayed macro-F1 0.6700. The score is release metadata: local replay proves artifact identity but cannot recompute hidden-label evaluation.

11.2 Partial fresh-response regeneration

The independent fresh-response audit is invoked from the repository root with:

```
export LMKBC_API_BASE=https://openrouter.ai/a⌋
↪  pi/v1
export LMKBC_API_KEY=...
python3 dataset2026/reproduction/\
  muse_glimmer30b_improved_candidate/rerun_mo⌋
  ↪  del_calls.py
  ↪  \
  --output-dir /tmp/muselist-fresh-calls
  ↪  --resume
```

The driver reads the public call manifest and executes every verification and set-completion prompt pack with a global Muse-model override, explicit provider order, disabled fallback, temperature zero,

recorded token ceiling, and the recorded reasoning and log-probability options. The 811 core-generation test calls are not part of this fresh-response driver. Credentials are accepted only through environment variables and are not written to the repository. A `--plan-only` mode prints the complete 2,917-call schedule without network access.

The frozen request packs contain no target labels or evaluator output. This mode regenerates fresh responses and tests whether the same prompts remain stable under a new hosted-model execution; it does not yet assemble those new responses into a fresh prediction and does not promise the historical response bytes or validation score. Exact artifact reproduction uses the archived command above.

The response writer records request IDs, returned provider/model metadata, attempt counts, elapsed time, reasoning text when supplied, completion log-probabilities when requested, and usage fields when the provider supplies them. Resume skips only already recorded request IDs. A fast contract test plus the complete archived-response regression is available as:

```
bash test_reproduction_contract.sh
```

The full regression executes every archived component, then checks the final prediction and ZIP identities. It does not turn a fresh provider rerun into an exact reproduction: new responses, validation score, and test ZIP may differ.

11.3 Artifact navigation

The repository at `https://github.com/UnGast/muselist-lmkbc2026` contains the reproduction root under `dataset2026/reproduction/`. Canonical request packs specify system and user messages, assistant prefixes, model/provider settings, token limits, request IDs, and dependencies. The core archive contains shared prompts and responses; the final-method bundle contains later-stage packs, responses, policy provenance, and independently stored stage outputs. The entry-point script resolves both automatically.

The released code and bundled task material retain their licenses and attributions in the repository's NOTICE. Model responses are distributed as experiment records. Credentials for optional fresh calls are supplied through environment variables, not stored in the archive.

The Instruction Tax on Factual Recall: Base Models Can Beat Their Instruction-Tuned Counterparts

Christophe Khalil*
American University of Beirut
cak29@mail.aub.edu

Aya Mourad*
Sorbonne Université, Paris, France
aya.mourad@sorbonne-universite.fr

Abstract

We present our solution to the AKBC LM-KBC 2026 Shared Task, where we achieved a score of 0.6836 on the official leaderboard, more than double the baseline of 0.2964. This task requires generating all possible objects for subject-relation pairs using a language model with up to 32 billion parameters, with no retrieval or fine-tuning involved. We show that the prompting methodology is as important as the model architecture: while the *instruction-tuned* model version achieves 0.5681 with its chat interface, the *base* model achieves 0.6836 when prompted using text completion. Thus, instruction tuning decreases factual recall; we refer to this effect as the *instruction tax*, which is caused by the model's weights rather than the prompting method.Our final solution is fairly straightforward: we prompt the base model with some examples from the training data, generate 64–256 outputs, and retain only the repeated ones. Two specialized methods are then applied for each relation: one improves the recall of large answer sets, and the other verifies a person's vitality before predicting their place of death.

1 Introduction

Knowledge graphs (KGs) serve as an essential framework for representing factual information in a structured format, enhancing semantic search and reasoning. Manual curation of such frameworks is expensive, and the resulting graphs always contain substantial information gaps (Schneider et al., 2022; Hofer et al., 2024). Although large language models (LLMs) have learned tremendous amounts of factual knowledge implicitly during their pre-training (Petroni et al., 2019), extracting this knowledge is challenging due to prompt sensitivity and hallucinations (Jiang et al., 2020; Zhang et al., 2025).

To evaluate progress on these extraction challenges, the AKBC 2026 Shared Task[1] assesses open-weight LLMs with fewer than 32 billion parameters under a strict *closed-book* setting, where retrieval, external knowledge sources, and fine-tuning are prohibited. The task requires models to predict all valid objects for a given subject–relation pair. For instance, the query *(Albert Einstein, city-of-death)* should produce the answer *Princeton*, whereas the query *(Barack Obama, city-of-death)* should return an empty set because no valid answer exists. This ability to abstain from producing unsupported facts is an essential part of the benchmark, as nearly 20% of the evaluation instances cannot be answered.

Our system is inspired by an empirical finding that we name the *instruction tax*. While *instruction-tuned* models (Ouyang et al., 2022) are generally favored for question answering tasks, our findings show that *instruction-tuned* models can sometimes significantly under-perform *base* models in retrieving facts. For instance, on the area relation, the *base* model queried by simple text completion outperforms the *instruction-tuned* model version by 0.29 through its chat interface, and by 0.30 under the same completion prompt. This gap persists despite prompting the *instruction-tuned* models using completion-style prompts.

Building on this finding, we developed a system based on five methodological principles: (1) choosing the *base* model through a fast probing process that takes about one GPU-hour; (2) prompting the *base* model using few-shot continuations rather than asking questions explicitly; (3) sampling 64–256 times for each row, and keeping only what recurs and returning empty if there is no recurrence; (4) adding a verification step for low-frequency examples in relations with hundreds of possible answers; and (5) breaking down more complicated

*Equal contribution.

[1] https://lm-kbc.github.io/challenge2026/

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 518–526
October 28, 2026 ©2026 Association for Computational Linguistics

tasks, such as checking whether the person is dead before predicting the *city-of-death*. Code, prompts, fitted thresholds, and the commands that reproduce each table are available at https://github.com/aiamourad/lm-kbc-2026/.

2 Related Work

Language models have been tested for factual recall in multiple studies. The LAMA benchmark was introduced by Petroni et al. (2019) and consists of factual questions answered using the fill-in-the-blank format. Roberts et al. (2020) showed that a fine-tuned model can answer factual questions from its parameters alone, and Brown et al. (2020) showed the same without weight updates, using a few in-context examples as prompts. Still, factual recall is notably less effective for long-tail entities (Sun et al., 2024), making long-tail entities the hard case for our task. The LM-KBC benchmark (Singhania et al., 2022) adds an evaluation setting for factual recall that requires predicting the entire set of answers and compares the predictions against accepted aliases.

Instruction tuning enhances the ability to follow instructions but can impair other abilities (Ouyang et al., 2022), with Zhou et al. (2023) arguing that it alters behavior more than knowledge. This paper quantifies these impacts in terms of factual recall. In addition, our method employs repeated sampling, akin to the self-consistency approach (Wang et al., 2023; Manakul et al., 2023), and examines the trade-off between efficiency and recall in mixture-of-experts models (Fedus et al., 2022).

3 AKBC Shared Task & Metrics

3.1 Dataset

This task contains six relationships and has 477, 475, and 475 examples in the train/val/test splits, respectively. There are two numerical relationships (`hasArea`, `hasCapacity`) for which the answer is a single number, and it is correct if it differs from the ground-truth answer by no more than ±5%. `zzawardWonBy` may have up to several hundred objects that the model must list to be considered correct. The remaining three relationships (countries (or comparable territories) that share a land border with the subject (`countryLandBordersCountry`), the stock exchange(s) on which the company's shares are publicly traded (`companyTradesAtStockExchange`), the city where the person died (`personHasCityOfDeath`)) have small answer sets that are empty for 18–42% of training rows.

3.2 Official Scorer

The official scorer normalizes strings, maps a prediction to an entity if it equals one of that entity's aliases, and maps predictions to entities to maximize the number of matches; row scoring is done independently, and the scores are then averaged. Let S denote the predicted set, G the reference set, and T the number of matches. The row score is:

$$F_1(S,G) = \begin{cases} 1 & S = G = \emptyset \\ 0 & \text{one of S,G empty} \\ 2T/(|S|+|G|) & \text{otherwise.} \end{cases} \quad (1)$$

The first two cases derive from the scoring conventions: abstention on an empty-answer row scores 1.0, whereas guessing on an empty-answer row scores 0. The baseline provided by the organizers never abstains and hence scores 0 on every empty-answer row. The leaderboard uses this scorer, and our validation scores come from running it locally.

4 Model Selection

The knowledge probe Which model should we select, and how should we interact with it? To this end, we provide the answers by means of an easy and cheap procedure called the *knowledge probe*. It consists of about 100 examples with known answers. For each of them, we query each of the models under consideration using several phrasings to ask for the corresponding fact and sample their answers 8–12 times. Next, we compute how often each particular answer was generated.

This results in two scores for each candidate model. The first one is called *voted* and stands for the proportion of examples whose most frequently returned answer is the correct one. The second one is the *ceiling*, i.e., the proportion of examples for which the correct answer appears at least once in the generated responses. This score reflects the upper limit of performance under such a probing procedure, since no selection strategy can return an answer that does not occur in the samples.

We probe the two numeric relations, as each sample yields a single number that can be parsed and checked automatically against the 5% tolerance threshold.

Which model version to ship Table 1 highlights the results that were derived from around one GPU-

	voted accuracy			tax	*ceiling*
	base	it-tuned			
family	compl.	chat	compl.	Δ	base →chat
Families publishing both model versions					
gemma-3-12b	.720	.370	.320	+.350	.84→.47
gemma-3-27b	**.880**	.590	.580	+.290	.95→.69
gemma-2-27b	.720	.640	.640	+.080	.82→.77
Mistral-Nemo-2407	.710	.630	.660	+.080	.90→.87
Instruction-tuned only					
Mistral-Small-3.2-24B	—	.720	.690	—	—
Magistral-Small-2506	—	.610	.570	—	—
Qwen3-30B-A3B (MoE)	—	.430	.430	—	—
Qwen3-30B-A3B-Think.	—	.430	.410	—	—

Table 1: Knowledge probe on `hasArea`, 100 validation rows, leading zeros omitted. Each row is one weights family, queried three ways: the base model by completion, and the it-tuned model by chat and by completion. *voted* is the accuracy of the best question form. Δ is the probe tax, base minus chat, on this relation alone; the full-system tax is Table 4. *ceiling* is the share of rows whose correct answer appears in at least one sample, base against chat.

hour of computing effort per model version, and this guided the selection of the model. In the probe we ask for the same fact in several different wordings, such as "The area of X is" or "X covers an area of". For each model version, we then report only the score of its best wording. We pick that wording using the same 100 rows we score on, so these numbers are a little too high. We treat every model version in the same way, but this helps a model more when its wordings give very different results. The table is therefore best used to rank the model versions, not to predict what one fixed prompt would score. Dense models outperformed the mixture-of-experts (MoE) model across all the experiments. While the evaluation counts an MoE model by its total number of parameters, not all of these parameters get activated per token, and the performance appears to depend on the amount of such computation. The reasoning-tuned model does not provide any noticeable improvement.

The instruction tax We observe a discrepancy between the scores of the base 0.880 and instruction-tuned 0.59 version of the gemma-3-27b model. To analyze whether the following is resulting from the nature of the prompt, we prompt the instruction-tuned model using the same completion prompt (compl. under it-tuned in Table 1) used for the base model version. Even with an identical query, it fell short at 0.580, and the sample ceiling falls from 0.950 to 0.690. This gives us a clearer image that the tuned model forgot facts it learned.

To check how consistent this pattern is, we test three additional pairs (gemma-3-12b, gemma-2-27b, and Mistral-Nemo-2407). In all eight tests, the base model version is at least as good as the instruction-tuned model version on both numerical relations. This appears to be a feature not inherent to instruction tuning per se, but dependent on the particular tuning run. Inside the Gemma family, the difference is 0.290 for gemma-3-27b and 0.350 for gemma-3-12b, but only 0.080 for gemma-2-27b, even though it is the same size as gemma-3-27b. In other words, with one generation less, the same tuning carries only a quarter of the cost. Mistral-Nemo-2407 also shows this small difference of 0.080.

The key difference between the two cases lies in the ceiling. With a high tax, performance on the matched completion prompt becomes very poor: dropping from 0.950 to 0.690 in gemma-3-27b, and from 0.840 to 0.470 in gemma-3-12b. With a low tax, the ceiling remains almost intact, dropping merely from 0.820 to 0.770 in gemma-2-27b, and from 0.900 to 0.870 in Mistral-Nemo-2407. It seems, therefore, that a high tax results in the loss of some answers from the sampling distribution, whereas a low tax results in poor answer selection from the remaining sample. Since both can be quantified using the same one-hour probe, the distinction can be made prior to committing to a model version.

As shown in Section 6, the probe order matches the order of the official submission runs, suggesting that the probing evaluation serves as an effective proxy for full submissions. The final system employs the base model version, gemma-3-27b-pt (Gemma Team, 2025), for all six relations.

5 Methodology

For inference, we deploy only one model (*gemma-3-27b-pt*) for the whole set of tasks. This model has 27.43 billion parameters, obtained precisely by counting the released weights rather than relying on the model's name, which may be incorrect. Within the task's 32-billion-parameter budget, it is impossible to deploy any other useful model in parallel with it. Inference is provided with the help of vLLM (Kwon et al., 2023) using two H100 GPUs and thus is performed for the whole test split in just a few minutes.

Relation	Completion format	k	n
hasArea	The total area of {s} is {v} square kilometres.	5	64
hasCapacity	The seating capacity of {s} is {v}.	5	64
countryLandBordersCountry	Countries sharing a land border with {s}: {v}	12	128
companyTradesAtStockExchange	{s} stock exchange listings: {v}	12	256
personHasCityOfDeath	City where {s} died (or none if alive as of 1 July 2026): {v}	12	128
awardWonBy	Recipients of {s}: {v}	4	64

Table 2: Completion formats and sampling budgets (k: exemplars, n: samples per row); {s} is the subject and {v} the gold value or none.

Never ask a question. For each relation, there is one completion template (see Table 2). A prompt is composed of a few examples selected from the training dataset and ends with an unfinished example that corresponds to the subject of the query:

```
The total area of Lake Victoria is 68800
square kilometres.
The total area of Bodensee is 536 square
kilometres.
The total area of Vygozero is
```

Such a formulation will naturally encourage the continuation of the observed pattern rather than an interpretation of the input as an instruction. There is no chat template and no task-specific instruction, and the model receives only the text in the exact form it should produce.

There are two crucial design decisions here. First of all, the numerical examples are chosen in such a way that they cover the whole distribution of numbers in the training dataset, including one sample from each quintile. This will prevent a bias towards a certain numerical range in the prompt. Second, examples with an empty response in the case of set-valued relations make up about one third of the exemplars and are marked as "none".

Ask many times, keep what repeats. Each row is sampled n times with different random seeds, and every output is truncated at the first line break. The temperature is 0.7 throughout. The exact value matters little: between 0.3 and 1.3 the differences stay within noise once the decision thresholds are refitted. Only very low temperatures do real damage, making the 64 samples nearly identical, collapsing the process to a single completion and removing the small benefit of aggregation.

We then parse the outputs into the required format for each relation type: a single numeric answer for numeric relations, and a comma-separated list for set-valued relations, where the output "none" denotes the empty set. Prior to aggregation, the responses are normalized in the same way as in the evaluation script: string-valued answers are merged according to the scorer's normalization rules, and numeric answers are grouped using the scorer's 5% tolerance.

For numerical relations, denote the parsed samples as $v_1, \ldots, v_m$, where $m \leq n$ because not every output parses into a number. For each candidate c, the cluster $C(c)$ is formed by all those samples lying within 5% of c, and the prediction is the median of the biggest cluster:

$$\hat{v} = \operatorname{median} C(c^*), \quad c^* = \arg\max_c |C(c)|, \quad (2)$$

where $C(c) = \{\, v_i : |v_i - c| \leq 0.05\, c \,\}$. This can be considered as majority voting modified to match the evaluation function: since the scorer allows any prediction that is within 5 percent of the reference value, votes are counted on tolerance clusters and not on numeric equality.

For relations with sets, we define $A_1, \ldots, A_m$ as the sets of answers after parsing. Support $s(a)$ for a candidate answer a is defined as its frequency of recurrence, that is, the fraction of the A_j that contain a:

$$s(a) = \tfrac{1}{m}\big|\{j : a \in A_j\}\big|, \quad S = \{a : s(a) \geq \tau_r\}. \quad (3)$$

The threshold τ_r is the sole parameter fitted for each relation independently. Instead of picking it manually, we pick the one that achieves the best official score on the training examples, and re-fit it each time either the number of samples n or the number of exemplars k is modified. These thresholds are more conservative for those relations where misprediction carries a high cost. In particular, for personHasCityOfDeath, the system will predict the city only if at least 40% of the 128 samples vote for it. For those relations having a bigger reference set, the fitted threshold is typically smaller. Abstention is a natural consequence of the same approach, since if none of the candidates achieves the needed recurrence rate, the predicted set is empty.

Verifying rare candidates for `awardWonBy`. The relation `awardWonBy` is especially challenging because its answer set can have hundreds of names. In such a situation, it is difficult to differentiate the genuine candidate, which only occurs three times among 64 samples, from the hallucinated candidate by recurrence alone. For this reason, we employ the model as a second-stage verifier using completion. We first present the model with ten verification examples from the training set that are evenly divided into positive and negative examples (e.g., `Did Barack Obama get the Nobel Peace Prize? yes` and `Did Marie Curie get the Turing Award? no`), and then pose the same question to the model regarding the candidate. We sample the verification question eight times, and let $q(a)$ be the ratio of samples answered with yes. We then add back the candidates whose support is at least $2/64$ and whose $q(a) \geq 0.75$. The two threshold values are chosen based on training examples. In the verification process, true positives are separated from false positives by means of the Area Under the Curve (AUC), which has a value of 0.86 on training candidates (0.75 on validation candidates). Lastly, we perform a non-neural clean-up procedure to combine duplicate names such as "Robert Caro" and "Robert A. Caro". Without performing this step, the scorer would consider both entries to be different predictions.

Is the person dead or alive? Approximately 40% of the death-relation rows refer to people who are still alive, and the correct answer in such cases is the empty set. We took a two-step approach in which we identify whether a given subject is dead or alive. If the person is alive, we ask the model to assign the empty set. On the other hand, if the model is confident that the person is indeed dead, we ask it to recall the location of their death.

Information about whether the person is alive or dead is far more prevalent than information about where they died. The location of the death is tail knowledge, while the person's living status is more widespread. We exploit this difference as an initial filter, which we call the *existence check*: it predicts whether an object exists for the row at all, rather than which object it is. For every query, we take 16 samples of a separate completion, built from ten exemplars evenly split between living and dead subjects of the training split, of the form `As of 1 July 2026, {s} is alive/dead`, followed by the query subject. If all 16 samples predict that the person is alive, then the answer must be the empty set.

6 Results

Table 3 gives the final scores and Table 4 the comparisons across models and interfaces, all from official submissions. Relations contribute 100 rows each except `awardWonBy` (10), `countryLandBordersCountry` (67) so "All relations" averages over 475 rows, not six scores; the last row scores the abstention decision, with the empty set as the positive class.

	val	test (official)		
Relation	F1	P	R	F1
`awardWonBy`	0.266	0.389	0.343	0.338
`companyTradesAtStockExchange`	0.765	0.893	0.932	0.872
`countryLandBordersCountry`	0.989	0.962	0.976	0.958
`hasArea`	0.900	0.820	0.820	0.820
`hasCapacity`	0.299	0.316	0.316	0.316
`personHasCityOfDeath`	0.550	0.900	0.630	0.570
All relations	**0.669**	**0.759**	**0.712**	**0.684**
Abstention decision	—	.722	0.892	0.798

Table 3: Final system. Test columns are official; validation is the same scorer run locally. Formats and thresholds are fitted on training rows only.

Firstly, there is an *instruction tax*, which is significant, especially when it comes to relations that rely heavily on exact recall (Table 4). Within one weights family, using the *base* model for completion instead of the instruction-tuned model via the chat interface increases the score by $+0.116$. The greatest improvement can be seen in the numeric relations, namely area ($+0.203$) and capacity ($+0.156$). In fact, the *base* model outperforms the instruction-tuned model on every single relation.

The comparison only makes sense if the instruction-based model is a strong one. To ensure this, we created a detailed prompt with a few-shot examples and challenged the model with yes-or-no questions for each query. More details are provided in Appendix B. Even after giving the instruction-tuned system all these additional advantages, the simpler completion-based system still performed better on every relation. Our findings imply a strong difference between the *base* and *instruction-tuned* models when it comes to factual knowledge recall: the facts recoverable by the *base* model become extremely rare in the output of the *instruction-tuned* model.

Secondly, the ranking method of Table 1 has predicted each ordering in the official submission runs

Relation	official test F1		
	it, chat	pt, compl.	Δ
`hasArea`	0.617	0.820	+0.203
`hasCapacity`	0.160	0.316	+0.156
`companyTradesAtStockExchange`	0.784	0.872	+0.088
`personHasCityOfDeath`	0.480	0.570	+0.090
`awardWonBy`	0.271	0.338	+0.067
`countryLandBordersCountry`	0.946	0.958	+0.012
All relations	0.568	**0.684**	+0.116
Mistral-Small-3.2-24B, chat		0.613	
Mistral-Small-3.2-24B, compl.		0.615	
Magistral-Small-2506, compl.		0.578	
Qwen3-30B-A3B (MoE), chat		0.534	
Qwen3-30B-A3B (MoE), compl.		0.551	
Qwen3-30B-A3B-Think., compl.		0.510	
task baseline		0.2964	

Table 4: The *instruction tax* at full-system scale, all scores official. Upper block: the gemma-3-27b family, gemma-3-27b-it by chat against gemma-3-27b-pt by completion, with Δ the second minus the first for each relation. Lower block: model versions with no base sibling, so there is nothing to subtract; each row is a single score, marked *chat* when the model was asked through its chat interface and *compl.* when it was given the completion prompt instead.

accurately, as the lower block of Table 4 shows. The dense models were ranked ahead of the MoE model (0.568 and 0.613 against 0.534) under the same instruction-tuned system, and the *base* checkpoint was ranked ahead of all of them at 0.681. The point is that the decisions which are key to achieving the highest score, model and interface, can be determined using only one GPU hour of computing time. Thirdly, sampling should be increased until little or nothing is gained, and this appears to be contingent on the relation. The official score increased from 0.6218 with n=8 samples to 0.6292 with n=24 samples, and again to 0.6604 with n=64 samples, the latter one being computed using the company format in Table 2. More samples were useful for some relations; for set relations, the gains continued until n=128, while for the company relation, the optimal number was found to be n=256. The verification rescue contributed an additional +0.027 on its relation, and the existence check contributed +0.030 on the death relation, which gave 0.6809. Doubling the samples for the numeric relations (n=128) did not help, so n=64 already saturates them.

Last, much of the gain comes from knowing when to say nothing (last two rows of Table 3). Empty exemplar lines plus the training-fitted threshold give an abstention F1 of 0.773; the existence check raises it to 0.798 and death precision to 0.900. The check catches an error recurrence cannot: the model naming a city, consistently, for someone still alive.

7 Conclusion

The design of our system rests on one insight: when it comes to factual recall, instruction tuning is not worth the expense. On the same parameters, gemma-3-27b-it scores 0.568 through its chat interface, while gemma-3-27b-pt scores 0.684 by completion. The rest follows: a one-GPU-hour probe selects the model version and the interface, each row is sampled between 64 and 256 times from the completion prompt, and only those responses above a fitted threshold are retained, making abstention the norm rather than the exception. The final score is 0.6836, against 0.2964 for the task's baseline. Two takeaways should apply elsewhere: the model version and the interface need to be tested, and where the best answer is often empty, knowing when not to answer helps almost as much as knowing more.

Limitations

The instruction tax is calculated on four (base, instruct) pairs. Officially, changing the weights alone costs 0.085 for gemma-3-27b and gemma-3-12b, whose sampling ceiling collapses, but 0.016 for Mistral-Nemo-2407 and 0.000 for gemma-2-27b, whose ceiling holds: the size of the tax depends on the tuning process, and four families are not enough to determine its applicability. The system is also quite costly, taking 64–256 generations to process each row, while the instruction-tuned baseline uses only one, and it is not zero-shot: the prompts are constructed from the training rows with labels, and the thresholds have to be recalibrated every time either n or k changes. The verification rescue and the existence check were each crafted by hand for a single relation.

Acknowledgments

This work was performed using HPC resources from GENCI-IDRIS (Grant 2025-AD011016170).

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33.

Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2011. An exact algorithm for f-measure maximization. In *Advances in Neural Information Processing Systems (NIPS)*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23.

Gemma Team. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Marvin Hofer, Daniel Obraczka, Alieh Saeedi, Hanna Köpcke, and Erhard Rahm. 2024. Construction of knowledge graphs: Current state and challenges. *Information*, 15(8):509.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Phillip Schneider, Tim Schopf, Juraj Vladika, Mikhail Galkin, Elena Simperl, and Florian Matthes. 2022. A decade of knowledge graphs in natural language processing: A survey. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 601–614.

Sneha Singhania, Tuan-Phong Nguyen, and Simon Razniewski. 2022. LM-KBC: Knowledge base construction from pre-trained language models. In *Proceedings of the Semantic Web Challenge on Knowledge Base Construction from Pre-trained Language Models (LM-KBC @ ISWC)*, CEUR Workshop Proceedings.

Kai Sun, Yifan Ethan Xu, Hanwen Zha, Yue Liu, and Xin Luna Dong. 2024. Head-to-tail: How knowledgeable are large language models (LLMs)? A.K.A. will LLMs replace knowledge graphs? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Nan Ye, Kian Ming A. Chai, Wee Sun Lee, and Hai Leong Chieu. 2012. Optimizing f-measures: A tale of two approaches. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2025. siren's song in the ai ocean: A survey on hallucination in large language models. *Computational Linguistics*, 51(4):1373–1418.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. LIMA: Less is more for alignment. In *Advances in Neural Information Processing Systems*, volume 36.

A Base and Instruction-Tuned Checkpoints, End to End

In Table 5, the whole system developed in Section 5 is run on each family releasing both checkpoints, with one call per checkpoint, and is evaluated on the official test set. Everything else in the pipeline, the sampling budget, and the training process is exactly the same for the eight runs; the only difference is in the weights. The probe columns reproduce Table 1. Eight runs cost 7 GPU-hours on two H100 GPUs.

The cost of each of these jobs is given in Table 6. All jobs use two H100 GPUs, and thus the number of GPU hours is double the wall time. A job

	official test			probe	
family	base	tuned	Δ	Δ	ceiling
gemma-3-27b	.6836	.5932	+.0904	.290	.95→.69
gemma-3-12b	.6018	.5166	+.085	.350	.84→.47
Mistral-Nemo	.5881	.5724	+.016	.080	.90→.87
gemma-2-27b	.6190	.6189	+.000	.080	.82→.77

Table 5: The complete system on four base/instruct pairs, official test, leading zeros omitted. Δ: cost of the tuned weights; *ceiling* from Table 1.

consists of both validation and testing, including the threshold fitting discussed in Section 5; only gemma-3-27b-pt reuses the thresholds already fitted for it and does not need to fit them again, which makes it the cheapest one. The most expensive is gemma-2-27b due to the inefficient attention implementation.

	probe		full system	
checkpoint	wall	GPU-h	wall	GPU-h
gemma-3-27b-pt	0:03	0.1	0:21	0.7
gemma-3-27b-it	0:09	0.3	0:29	1.0
gemma-2-27b	0:04	0.1	0:38	1.3
gemma-2-27b-it	0:05	0.2	0:23	0.8
gemma-3-12b-pt	0:08	0.3	0:29	1.0
gemma-3-12b-it	0:12	0.4	0:22	0.7
Mistral-Nemo-Base-2407	0:07	0.2	0:25	0.8
Mistral-Nemo-Instruct-2407	0:10	0.3	0:24	0.8
Mistral-Small-3.2-24B	0:06	0.2	0:33	1.1
Magistral-Small-2506	0:05	0.2	0:27	0.9
Qwen3-30B-A3B	0:06	0.2	0:31	1.0
Qwen3-30B-A3B-Think.	1:01	2.0	0:28	0.9

Table 6: Wall time and GPU-hours per checkpoint; two H100s per job. A full run covers validation and test with fitting. Below the rule: no base sibling.

B The Instruction-tuned Comparator in Full

This appendix outlines the instruction-tuned system deployed for the "it, chat" row in Table 4. Of all the instruction-tuned models we experimented with, this is our best system, which scored 0.5681 with gemma-3-27b-it and 0.6127 with Mistral-Small-3.2-24B. The second model achieved 0.6161 when the border rules discussed in Appendix B were applied. This system utilizes a very simple process that entails generating the potential answers, evaluating them separately, calibrating the scores, and choosing the highest-scoring combination.

Propose. The relation question is asked of the model for each row n times. Instead of using temperature to generate different outputs, we provide each sample with a new *approach instruction*, which is a one-sentence hint on how to solve the problem, e.g., "Walk around the country's borders clockwise and name its neighbours" or "First figure out whether the person was alive in 2026." In practice, it results in more productive disagreements among the samples compared to changing the temperature. Constrained decoding is used to make sure that the outputs remain in the same JSON format. Each of the prompts contains the definition of the relation given in the task and six example rows from the training set.

Group. Identical to the main system: the scorer's own normalisation for strings, 5% tolerance clusters for numbers.

Verification. Each candidate solution is then verified individually. The model receives a statement of the form "X is one of the correct answers to . . . ", and replies with either "Yes" or "No". Rather than sampling this reply, we take the log-probabilities of the first token: the score q is the probability mass on the tokens spelling "Yes", divided by the mass on "Yes" plus "No". The value of q obtained in this way is not always equally robust for different relations. For instance, it achieves an AUC of 0.931 for `awardWonBy` while reaching 0.450 for the border relation, a result inferior even to the random chance baseline. Owing to such variations, we do not fix any threshold for q; rather, we consider it to be one signal among others forwarded to the next step.

Calibrate. For each relation r, a logistic regression fitted using the training rows maps the signals to a probability. Let f_i be the recurrence probability for candidate i, q_i its verification probability, ρ_i its rank among the row's candidates in $(0, 1)$ space, and $\pi_i = 1/(1 + C)$ with C being the number of candidates; this is a five-parameter model per relation (four weights and a bias), fitted with an L_2 penalty.

Decode. The system outputs the set having the highest expected row score based on these probabilities. Rank the candidates by p_i and let S_k contain the top k; then only the $C+1$ sets $S_0, \ldots, S_C$ have to be compared (Dembczyński et al., 2011; Ye et al., 2012):

$$S^* = \underset{k\in\{0,\ldots,C\}}{\arg\max}\ \mathbb{E}[F_1(S_k, G)]\,. \qquad (4)$$

In the case when the real set can have at most one member, the candidates are mutually exclusive, and

the expectation admits a closed-form solution; otherwise, they are assumed to be independent, and the expectation is computed using dynamic programming on the p_i. The null set is the special case of $k{=}0$ and therefore does not require any threshold. In the case of `awardWonBy`, the model also infers the number of recipients and thus accounts for correct answers that are not in the candidate pool.

The Border Rules. The border relation is characterized by two features which a purely statistical pipeline cannot take advantage of: the borders are symmetrical, and each training/validation instance is a full list of its subject's neighbours. Representing by $\mathcal{G}(X)$ the reference list of neighbours of a country X that is a subject of the provided data, and by $\mathrm{dom}(\mathcal{G})$ the set of such subjects,

$$Y \in \mathcal{G}(X) \;\Rightarrow\; \mathit{border}(Y, X), \tag{5}$$

$$X \in \mathrm{dom}(\mathcal{G}) \wedge Y \notin \mathcal{G}(X) \;\Rightarrow\; \neg\, \mathit{border}(Y, X). \tag{6}$$

Rule 5 adds missing neighbours (checked against the validation gold, the rule proposed 47 neighbours and 46 were correct; during test, on the Mistral-Small-3.2-24B chat system, it added five cases like The Gambia for Senegal); Rule 6 deletes predicted neighbours which contradict the complete reference (during test, it deleted four cases like Sweden–Denmark, linked by a bridge not counted in the reference).

Both of them use reference data only, and predictions by the system itself are not considered evidence at all. In the case of the Mistral-Small-3.2-24B instruction-tuned system, the rules increased borders from 0.9487 to 0.9729 official and from 0.6127 to 0.6161 total. In the completion system, in which borders achieve 0.989 on validation with recall 1.000, the rules failed to pass the validation gate and were not used (Section 6).

One Workflow Does Not Fit All Relations: A System Description for the AKBC 2026 LM-KBC Shared Task

Andreea-Elena Cochintele and Anca Dobrovăț and Sergiu Nisioi
Faculty of Mathematics and Computer Science
University of Bucharest
andreea.cochintele@gmail.com
anca.dobrovat@unibuc.ro
sergiu.nisioi@unibuc.ro

Abstract

This paper presents a system developed for the fifth AKBC Shared Task on Knowledge Base Construction from Language Models (LM-KBC 2026). The task involves predicting sets of object-entity for subject-relation pairs across six relation types, ranging from single-valued relations to relations with potentially hundreds of recipients. All predictions are generated in a closed-book setting (Roberts et al., 2020), using up to 32 billion parameters across all neural components, without resorting to fine-tuning or retrieval augmented techniques. The development process consisted of two stages: validating the technical methodology using the officially specified baseline model (*Qwen3.5-9B*), followed by comparing eleven open-weight models using the same evaluation framework. This process led to the selection of Gemma 3 27B as the final system. The paper presents the main output-parsing errors identified and resolved, together with an evaluation of self-consistency and majority-voting aggregation for each relation type. The final system improved macro-F1 score from 0.361 to 0.531 (a +47% relative improvement) through workflow optimization and model selection.

1 Introduction

Pre-trained language models incorporate significant factual knowledge from their training data, a fact demonstrated by studies using single-answer cloze-questions (Petroni et al., 2019). The LM-KBC competition (LM-KBC Organizers, 2026) extends this line of research by analyzing whether a language model can function as a knowledge base. In this task, a subject-relation pair may have no correct object, a single correct object, or multiple correct objects, depending on the specific relation. The task does not assume a fixed number of correct answers. This distinction is important: for instance, the *awardWonBy* relation can have numerous correct answers, but *personHasCityOfDeath* typically has a single answer. A system that works well for one type of relation may not work as well for another. Furthermore, certain design choices that benefit one relation may have a negative impact on another.

This is a system description paper that reports engineering findings derived from a shared-task submission. Its main finding is that engineering choices—prompt generation, internal consistency, and aggregation—need to be tailored per relation rather than applied uniformly, since their effect depends on each relation's answer cardinality. The model comparison, the syntactic analysis taxonomy, and case studies shown in this paper are organized around establishing and illustrating this point.

Our contributions are:

- A controlled comparison of eleven open-weight models (Qwen, Mistral, Phi, Gemma, Llama) under the same evaluation harness.
- Identification of the 27B parameter model as the best available option within the 32B parameter budget of the competition.
- An empirically-derived taxonomy of output-parsing analysis errors in the results that occur when free-text is converted into lists of entities.
- An analysis showing that self-consistency and majority voting, based on the previous theory (Wang et al., 2022; Chen et al., 2023), help certain relations but have a negative impact on others.
- We document one case, on one relation, where an explicit negative instruction led to an increase in the frequency of the forbidden word, which is consistent with the known difficulties that medium-sized, open-weight models encounter regarding contextual negation.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 527–534
October 28, 2026 ©2026 Association for Computational Linguistics

Our code, prompts, and post-processing rules are publicly available.[1]

2 Related Work

Previous editions of the LM-KBC competition explored various approaches to constructing knowledge bases from language models. Li et al. (2022) won the first edition (ISWC 2022) with a BERT-based system that combined task-specific pre-training and prompt decomposition to progressively generate objects, achieving an F1 score of 55.0%. In the second edition, Nayak and Timmapathini (2023) fine-tuned the Llama 2 model using LoRA and context augmented by Wikipedia retrieval, achieving a mean F1 score of 0.6185 across 21 relations simultaneously. Zhang et al. (2023) reported an F1 score of 0.7007 using GPT-4 in a "closed prompting" scenario, while noting significant performance disparities across relations (e.g. 1.0 for *PersonHasNobelPrize* versus 0.328 for *PersonHasEmployer*), a pattern consistent with the cardinality-dependent behavior we analyze in subsection 6.2.

This paper is different from these prior systems in two ways. First, none of the systems mentioned combines an explicit “closed-book” constraint with a limit on the number of parameters. Based on competition rules, we restrict our comparison to open-weight models under 32 billion parameters, without fine-tuning or external retrieval. Second, we run a structured, per-technique ablation across eleven open-weight models and describe how prompting techniques, self-consistency, and majority-vote aggregation interact with each relation’s answer cardinality.

3 Task and Data

3.1 Task Definition

Given a subject entity and a relation, the task consists of predicting the complete and correct set of object entities, a set that may be empty, contain a single element, or comprise multiple elements. Predictions are evaluated as character strings: a predicted string is normalized and compared against the reference label (gold label) and its known aliases using a maximum-cardinality matching method; thus, the score is independent of the prediction order and each reference entity can be assigned at most one prediction. For the two numerical relations, a prediction is considered correct if it falls within a relative tolerance margin of 5% of the reference value.

[1] https://github.com/andreea-cochintele/lmkbc2026-submission

Relation	Train	Validation	Test
countryLandBordersCountry	67	68	67
personHasCityOfDeath	100	100	100
hasCapacity	100	100	100
awardWonBy	10	10	10
companyTradesAtStockExchange	100	100	100
hasArea	100	100	100

Table 1: Dataset statistics from the official task documentation.

3.2 Relations and Dataset Statistics

The dataset comprises six relations (see Table 1), among which three allow for empty sets, two are numerical, and the *awardWonBy* relation is documented as sometimes having hundreds of valid recipients for a small number of subjects. However, with only 10 subjects (an order of magnitude fewer than for the other relations), conclusions regarding this relation should be interpreted as purely illustrative.

3.3 Evaluation

Predictions are evaluated as normalized strings linked via aliases (see the taxonomy for cleaning the results in Table 7.)

The scorer identifies true positives in the bipartite match between predicted and gold entities, so the score is independent of the prediction order. Therefore, predicting the same entity using two surface forms reduces precision because it inflates the predictions without adding any true positives.

3.4 Competition Constraints

The competition imposed three rules that constrained the design of this system. First, the "closed-book" restriction prohibits web searches, RAG, or any consultation of factual information. Second, no additional training or fine-tuning is allowed. Third, the total number of parameters across all neural components must not exceed 32 billion. The largest model we evaluated, Gemma 3 27B, falls within this limit. Non-neural post-processing: normalization, deduplication, rule-based filtering, and vote aggregation are permitted and utilized.

4 System Description

4.1 Preliminary Ablation

Before adopting any model modifications, we checked the general engineering approach, including relation-specific prompts, per-relation token allocation, list cleaning, single-answer constraints, and majority voting on the officially designated model: *Qwen3.5-9B*, part of the Qwen family of models (Qwen Team, 2025). Each technique was implemented as a toggleable flag, tested first individually, then cumulatively, on the validation set (complete step-by-step results in Appendix C). Unless noted, all F1 values in this article are single run estimates, without testing for statistical significance; section 5 shows the only variance we measured (±0.005 macro-F1); *awardWonBy* figures reflect only 10 validation subjects.

Relation-specific prompts generated the greatest individual gain (0.361 to 0.415 in isolation), later confirmed on the Gemma 3 27B model. Per-relation token allocation, list cleaning, and single-answer constraints on the *personHasCityOfDeath* relation were adopted. Formatting results with one item per line was rejected. Instead of reducing the generated volume, it encouraged a higher number of output lines for relations with multiple answers.

Even if majority voting on the *awardWonBy* relation doubled precision (0.251 to 0.559) and halved the average number of predictions per subject (28.2 to 12.0), macro-F1 showed an insignificant change (0.082 $\rightarrow$ 0.083); therefore, we treat this as *an undetectable macro-F1 effect* rather than a null result. Re-evaluated on Gemma 3 27B, the same vote raised F1 from 0.147 to 0.161, attributable to the larger model's broader coverage of award winners, not the voting mechanism itself.

At this stage, prompting and post-processing alone brought the *Qwen3.5-9B* model to a macro-F1 score of 0.408, representing a relative improvement of 30% compared to the organizers' reported baseline.

4.2 Model Selection

We evaluated eleven open-weight models using the same testing framework: Qwen3.5-9B, Qwen2.5-7B-Instruct, Qwen3-8B, Qwen2.5-14B-Instruct, Qwen3-14B, Mistral-Nemo-Instruct-2407 (12B) and phi-4 (14 B), Gemma 3 (4B/12B/27B) and Llama-3.1-8B. The 14B-class models and the Mistral/phi models were 4-bit quantized with bitsandbytes library (Dettmers et al., 2023) to fit a 16 GB T4; Qwen3.5-9B, Qwen2.5-7B, and Qwen3-8B ran without quantization, while Gemma 3 and Llama ran through Ollama (GGUF). Full per-model scores are reported in Appendix B.

Two patterns emerged, though the unquantized and quantized models are not clean-cut (Appendix B): Qwen3.5-9B (0.408) and Qwen2.5-7B-Instruct (0.332), both unquantized, outperformed all the bitsandbytes-quantized models tested, but the third unquantized model, Qwen3-8B (0.256), did not (scoring lower than the Mistral-Nemo-Instruct-2407 model (0.321, 4 bits)). Quantization is therefore associated with lower scores here, but it is not sufficient, on its own, to explain the rankings. Second, within the Gemma 3 family, macro-F1 scaled with size (0.306 / 0.462 / 0.510 for 4B/12B/27B), and Gemma 3 27B outperformed all other models, despite being quantized using GGUF, which suggests that GGUF quantization and bitsand-bytes quantization are not interchangeable in this case, although we cannot isolate the quantization method from other confounds (the base model, the tuning recipe). We selected Gemma 3 27B as the final system and chose aggregation at the relation level (subsection 4.4) then raised the macro-F1 to 0.531.

4.3 Prompting

We use a fixed 5-shot prompt per relation, using examples drawn from the training set via a random number generator with a relation-specific seed, formatted as question-answer pairs, with a comma-separated list of answers (or the value "None"). This paper adds a relation-specific clarifying hint to both the query and each example in the "few-shot" set (Table 3). An initial run of the replicated baseline model exhausted its entire 64-token budget trying to determine the real-world entity being referenced, without ever generating a number. This explains the very low scores obtained by the baseline model for the *hasArea* and *hasCapacity* relations. The "one object per line" output format, tested first globally and then on *awardWonBy* alone, did not generate improvements (0.415 vs. 0.423) and was abandoned in favor of handling hedging and multi-entity lines at the parsing stage.

4.4 Output Parsing, Cleaning, Aggregation

Model output is free-form text, not structured data, and in early stages most errors traced back to parsing rather than to the model's knowledge. Table 7 lists the seven recurring failure types we identified.

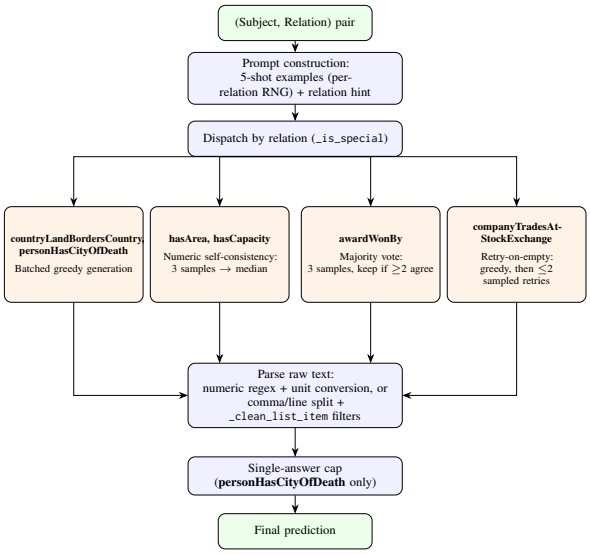


Figure 1: Per-item prediction pipeline. Each of the four branches applies to exactly the relations named on it; all other relations fall through to plain batched greedy generation.

Eliminating one type of failure altered the nature of the next.

Figure 1 illustrates the processing workflow for each item. In addition to "greedy" generation (immediately selecting the most probable option), which is the default setting, we implemented four aggregation strategies (Table 6), each of which was retained or strictly cancelled based on the measured "before/after" macro-F1 score values.

Majority voting for *awardWonBy* keeps a candidate only if it recurs, after normalization, in at least a fixed fraction of sampled generations, measurably improving precision every iteration. Numeric self-consistency (median of several samples) produced a clear improvement on *hasArea* (+0.070 F1) and an inconclusive change on *hasCapacity* (-0.010, within noise). Retry-on-empty helped *companyTradesAtStockExchange* but hurt *personHasCityOfDeath*, where an empty answer more often reflects a genuinely correct empty label than under-confidence. Full per-relation kept/discarded decisions are listed in Appendix D.

4.5 A Case Study in Prompt-Level Negation

In one instance, the *countryLandBordersCountry* relation suffered from the excessive generation of surface variants (translations, transliterations, outdated historical state names). Our initial solution (an explicit instruction such as *do not write 'West Germany' or 'East Germany', instead write 'Germany'*) made this measurably worse: predicted objects per subject rose from 1.691 to 4.441 in the very next run, and the forbidden strings appeared more than before. This isolated case is consistent with the known weaknesses of medium-sized open-weight models concerning negation handling: mentioning a token can increase its relevance (Kassner and Schütze, 2020). We reverted to a neutral prompt and moved the filtering out of the prompt and into the parsing stage: surface variants and explanatory text fragments are removed by the rules in Table 7. In the final system, *countryLandBordersCountry* is handled by batched greedy generation with no multi-sample aggregation (Figure 1). We do not claim that this generalizes beyond this single relation and phrasing. We report it because it was significant and contrary to the intent of the prompt.

	Validation		Test	
Relation	**Base**	**Ours**	**Base**	**Ours**
awardWonBy	0.055	0.168	0.151	0.259
companyTradesAtStockExchange	0.539	0.638	0.273	0.646
countryLandBordersCountry	0.784	0.975	0.732	0.921
hasArea	0.210	0.570	0.340	0.530
hasCapacity	0.090	0.210	0.122	0.112
personHasCityOfDeath	0.350	0.440	0.170	0.320
All	**0.361**	**0.531**	**0.296**	**0.473**

Table 2: Macro-F1 by relation, reproduced baseline vs. our system, on validation and test.

5 Setup

The final system is a subclass of the `AbstractModel` interface associated with the shared task. The documented baseline configuration specifies the Qwen3.5-9B model, a 64 token limit, 5-shot prompting (Brown et al., 2020), and no output format instructions. We reconstructed a faithful, minimal implementation without any pipeline enhancements, so that the comparison in Table 2 isolates our technical solution. We then ran a full validation set (478 pairs) using the final configuration—including multi-sample aggregation for four of the six relations.

Table 2 compares the reconstructed baseline model (Qwen3.5-9B) with the final system (Gemma 3 27B, full workflow) on the validation set and the hidden test set. Our system outperforms the baseline in both cases: +47% macro-F1 on the validation set (0.361 → 0.531) and +59.7% on the test set (0.296 → 0.473), indicating that the improvement generalizes beyond the training set. The biggest absolute gains are observed for hasArea (+0.360) and countryLandBordersCountry

(+0.191), while the smallest is for personHasCityOfDeath (subsection 6.2). Five of the six relations improve on the test set, mirroring the results from validation; the exception is hasCapacity (0.112 vs. 0.122), consistent with the noise-level effect of numerical self-consistency already mentioned for this relation (Table 6, −0.010 F1). As in the rest of the paper, these are point estimates obtained from a single run, without testing for significance.

Reproducibility

Three out of the seven workflow modifications evaluated were reverted after a negative "before/after" result (the *retry-on-empty* and *self-consistency* mechanisms for the *personHasCityOfDeath* relation, voting based on 5 samples and a threshold of 3 for the *awardWonBy* relation), highlighting that a technique beneficial for one relation may be detrimental to another with different cardinality characteristics. With deterministic seeds, repeated runs vary by at most ±0.005 macro-F1 for the two relations employing sampling-based aggregation; relations relying exclusively on *greedy* decoding reproduce identical results in every run.

6 Analysis

6.1 Long-Tail Knowledge

Well-documented entities (e.g., the area of a well-known country) were reproduced near-exactly by every model, including the smallest. "Long-tail" entities (e.g. a small Croatian island) resulted in large errors, presented with a high degree of certainty, across all models, including the final system (Kandpal et al., 2023). In one case, the estimated area of a country was exactly ten times the correct value, suggesting a specific figure that was memorized incorrectly yet confidently, rather than random noise.

Our final system measurably reduced the long-tail error for *hasArea*, where F1 increased from 0.210 (reproduced baseline, Table 2) to 0.570. The largest part of this gain comes from the relation-specific hint, which eliminated the token-budget failure described in subsection 4.3; numerical self-consistency added a +0.070 F1 improvement over that configuration. Neither these methods nor their combination eliminates errors for extremely rare entities. Since the closed-book rule prohibits consulting external sources, this limitation is intrinsic to the task as defined, rather than a flaw in the workflow employed. We present this aspect as a qualitative pattern derived from error analysis, rather than as a quantified statement regarding rare entities in general.

6.2 Self-Consistency Depends on Answer Cardinality

One factor is consistent with all five aggregation experiments we ran: either the independent samples converge toward a correct answer that greedy decoding misses, or greedy decoding already provides an answer close to the best individual answer, in which case sampling largely adds noise. The clearest evidence comes from the two largest relations (100 subjects each, and thus less prone to the noise specific to small samples than *awardWonBy*): for *hasArea*, greedy decoding of long-tail entities often yields an incorrect answer, so the median of the samples is helpful; for *personHasCityOfDeath*, greedy decoding is usually already the best single answer for this specific, single-value fact, and extra samples can overturn a correct greedy answer if two of them happen to agree on the same incorrect city. The relation *awardWonBy* (10 subjects) shows the same qualitative pattern — majority voting filters out isolated, hallucinated names while retaining those consistently reproduced.

6.3 Negation, Revisited

The prompt-level failure in subsection 4.5 is consistent with previous findings that pre-trained language models handle contextual negation unreliably (Kassner and Schütze, 2020). We recommend validating any instruction of the form “do not write X” through a before-and-after comparison.

6.4 Incompleteness of ground-truth data

Two of the ten *awardWonBy* validation subjects pertain to honors such as *doctor honoris causa* titles, supported by a documented but incomplete reference set, which limits accuracy, regardless of the system’s quality. We initially treated every high-frequency prediction for this relation as a probable error; a re-examination of the task documentation revealed that many of these were correct entries but underrepresented in the reference set.

6.5 Qualitative Case Studies

Successive analysis corrections applied to the same subject revealed, one by one, three types of errors: misspelled names, incomplete disambiguation suffixes, and conversational preambles—with each correction revealing the next error. A more

systematic error arose in the case of Russia's borders: our predicted list, identical across all three self-consistency samples, included only six of Russia's fourteen actual neighbors, plus seven non-neighboring countries, omitting two of the most important ones. Since the error was identical each time, majority voting could not correct it. The model maintained a false association with confidence. Self-consistency eliminates idiosyncratic noise, but does not eliminate a systematic error that is reproduced identically in every run.

6.6 Practical Trade-offs

Larger models generally achieve higher scores (Appendix B): Gemma 3 27B, quantized by GGUF, outperformed several unquantized models under 14B, indicating that the raw scale is not the only influencing factor within a fixed parameter budget. Self-consistency and majority voting involve additional inference calls (3 per element) for gains that depend on relations and are sometimes zero (subsection 6.2). Furthermore, majority voting trades recall for precision: it filters out isolated hallucinations, but a correct answer generated inconsistently across samples may be eliminated by the vote along with the noise.

7 Conclusions

We described a system for the AKBC 2026 LM-KBC shared task, developed using eleven open-weight models. A workflow designed for relation-level aggregation and an ablation-validated decision process produced gains beyond model selection alone (macro-F1 rising from 0.361 to 0.531 relative to a reproduced baseline). We also documented two case studies where explicit negative prompting and generalized self-consistency did not transfer well across relation cardinalities.

Acknowledgments

This research is supported by the project "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 351416.

Limitations

- We did not perform statistical significance tests, calculate confidence intervals, or estimate variance for most of the comparisons presented; nearly all results in this paper are point estimates obtained from a single run. In cases where we separately measured the variation from one run to another (±0.005 macro-F1 for sampling-based relations, section 5), we use this as an informal noise threshold, but we did not measure the variance for each configuration compared in tables 5-2, so some of the reported differences (particularly those below ∼0.01 F1) may not represent reliable effects.
- *awardWonBy* has only 10 subjects per division, an order of magnitude fewer than the other five relations. Many of our conclusions regarding majority voting and self-consistency rely heavily on this relation.
- The "closed-book" constraint excludes augmented generation through retrieval (the most direct solution for sparse numerical data, see subsection 6.1), so we cannot test it in accordance with the task rules.
- Residual variation from one run to another (±0.005 macro-F1) persists even with deterministic seeds, due to the non-deterministic order of floating-point number reduction in multi-threaded GPU execution.
- Available computing resources dictated the process: all experiments were run on Kaggle's free tier (weekly limit of 30 hours, two 16-GB T4 GPUs).

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2023. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818.

Tianyi Li, Wenyu Huang, Nikos Papasarantopoulos, Pavlos Vougiouklis, and Jeff Z. Pan. 2022. Task-specific pre-training and prompt decomposition for knowledge graph population with language models. In *Proceedings of the LM-KBC Challenge at ISWC 2022*.

LM-KBC Organizers. 2026. Akbc shared task 2026: Knowledge base construction from language models (5th edition). https://lm-kbc.github.io/challenge2026/. Accessed 2026.

Anmol Nayak and Hari Prasad Timmapathini. 2023. Llm2kb: Constructing knowledge bases using instruction tuned context aware large language models. In *Proceedings of the LM-KBC Challenge at ISWC 2023*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Bohui Zhang, Ioannis Reklos, Nitisha Jain, Albert Meroño Peñuela, and Elena Simperl. 2023. Using large language models for knowledge engineering (llmke): A case study on wikidata.

A Hints per Specific Relation

Table 3 lists the exact clarifying hint appended to every query and few-shot example for each relation, as discussed in subsection 4.3.

Relation	Appended hint
hasArea	"Answer with a single number in square kilometers, no units, no commas."
awardWonBy	"Name the recipient (person, group, or organization) who WON the award, not the work/project that was awarded. List as many confirmed winners as you know."
countryLandBordersCountry	"Only current, internationally recognised land borders. Exclude maritime-only neighbors. If none, answer None."
personHasCityOfDeath	"Name only the city (not the country or region). If still alive or unknown, answer None."
companyTradesAtStockExchange	"Name the stock exchange(s). If not publicly traded, answer None."
hasCapacity	"Answer with a single integer number of people, no units, no commas."

Table 3: Relation-specific prompt hints appended to every query and few-shot example.

B Full Model Comparison

Table 4 reports validation macro-F1 for all eleven models evaluated in subsection 4.2, plus the quantization method and parameter count of each.

Model	Params	Quant.	Macro-F1 (val.)
Qwen3.5-9B	9B	none	0.408
Qwen2.5-7B-Instruct	7B	none	0.332
Qwen3-8B	8B	none	0.256
Qwen2.5-14B-Instruct	14B	4-bit	0.291
Qwen3-14B	14B	4-bit	0.188
Mistral-Nemo-Instruct-2407	12B	4-bit	0.321
phi-4	14B	4-bit	0.234
Gemma 3 4B	4B	GGUF	0.306
Gemma 3 12B	12B	GGUF	0.462
Gemma 3 27B	27B	GGUF	0.510
Llama-3.1-8B	8B	GGUF	0.190

Table 4: Macro-F1 by model on the validation set.

C Full Preliminary Ablation

Table 5 shows every step of the incremental ablation on *Qwen3.5-9B* described in subsection 4.1, in the order each flag was tested.

Configuration	F1
Official external baseline (reference)	0.313
Own vanilla run, no improvements	0.382
All flags off (re-confirmation)	0.361
+ relation hints only	0.415
+ per-relation token budget only	0.377
+ list cleaning	0.406
All 4 base flags + fixed seed	0.423
+ one-per-line format (global)	0.415
+ isolated per-relation RNG + majority vote	0.408
Final: isolated RNG, majority vote off	0.408

Table 5: Ablation of engineering techniques on the reference model (*Qwen3.5-9B*), testing order.

D Full Aggregation Strategy Results

Table 6 lists, for every relation where a multi-sample aggregation strategy was tried (subsection 4.4), whether that strategy was ultimately kept in the final system.

Relation	Strategy	Kept?
awardWonBy	Majority vote, 3 samples, thr. 2	Yes
awardWonBy	5 samples, thr. 3	No
companyTradesAtStockExchange	Retry-on-empty	Yes
personHasCityOfDeath	Retry-on-empty	No
personHasCityOfDeath	Self-consistency, single answer	No
hasArea	Self-consistency, median of 3	Yes
hasCapacity	Self-consistency, median of 3	Yes(Neutral)

Table 6: Aggregation strategies tried per relation.

E Output-Parsing Failure Taxonomy

Table 7 lists the seven recurring failure types identified during development (subsection 4.4) and the rule-based fix applied to each.

Failure mode	Fix
Meta-commentary	Reject hedging lines and long lowercase-terminated sentences
Label-like residue	Reject items ending in a colon or with a comment marker
Punctuation residue	Combined edge-stripping in one pass
Unclosed suffix	Truncate at an unmatched opening parenthesis
Self-reference	Drop candidates equal to the subject entity
Conversational preamble	Exact-match filler-word and list-intro filters
Numeric drift	Strip thousand-separators; unit conversion for hasArea

Table 7: Taxonomy of output-parsing failure modes and fixes.

CoRe: Construct-versus-Recall Routing for Knowledge Base Construction Under a Parameter Budget

Fatih Naufal Habibillah
Universitas Indonesia
fatih.naufal51@ui.ac.id

Filbert Aurelian Tjiaranata
Universitas Indonesia
filbert.aurelian@ui.ac.id

Abstract

We report a system for the LM-KBC shared task under a 32B budget on the total parameters of all inference-time components, which rules out scaling to a larger single model and caps any ensemble at the same 32B total. Starting from a rationalized chain-of-thought (CoT) pipeline that reached macro-F1 0.522 with Gemma-4-31B-it, we show that swapping to a smaller, differently-trained model (Qwen3.5-27B) under an identical pipeline recovers 24 percentage points of accuracy on a purely numeric relation (surface area), while costing accuracy on relations requiring broad factual recall. This shows that parametric knowledge coverage, not reasoning capability, is the limiting factor at this model scale. We then present four negative or reversed results (self-consistency sampling on numeric relations, a confabulation-guard prompt, reasoning-budget effects, and non-comparable validation splits) as diagnostic evidence for how scaffolding interacts with relation type. We introduce CoRe routing (Construct versus Recall), which assigns each relation to CoT or direct answering according to whether its answer must be constructed or merely recalled. CoRe reaches macro-F1 0.546 on validation and 0.541 on test.

1 Introduction

The LM-KBC challenge asks systems to construct knowledge base triples directly from a language model's parametric knowledge, given a subject entity and a relation, without external retrieval (Singhania et al., 2023; Petroni et al., 2019). Prior systems have shown that prompt phrasing, few-shot construction, and ensembling can move scores by 5–15 percentage points independent of the underlying model (Biester et al., 2023; Alivanistos et al., 2022). We study this sensitivity under an additional constraint that prior LM-KBC systems did not face: a hard limit of 32B on the total parameters of all inference-time components. This constraint matters because capacity cannot simply be added: scaling to a larger single model (Li et al., 2022) is ruled out, and any ensemble of models (Biester et al., 2023) must fit within the same 32B rather than stacking on top of a strong single model.

Our task instance covers six relations spanning three qualitatively different extraction problems: multi-valued list construction (*award recipients*, *country borders*), single-valued numeric recall (*venue capacity*, *geographic area*), and calibrated abstention (*place of death*, *stock exchange listing*, where a substantial fraction of gold answers are the empty set). We show that these three problem types respond to scaffolding in opposite directions, so no single global prompting strategy is optimal for all of them.

Our contributions are: (1) CoRe routing, a per-relation policy that outperforms any single global configuration we tested; (2) evidence that a model swap within the parameter budget recovers more accuracy on knowledge-limited relations than any prompting intervention we found; (3) four diagnosed negative results, each explaining why a plausible intervention does not transfer to this setting.

2 Task and Data

The task provides six relations over a shared entity pool, split into train/validation/test with approximately 100 rows per relation (68 for *countryLandBordersCountry* and 10 for *awardWonBy*). Two relations (*hasArea*, *hasCapacity*) are strictly single-valued numeric and never empty in gold; two (*personHasCityOfDeath*, *companyTradesAtStockExchange*) permit the empty set at substantial rates (39% and 36% of validation rows respectively); the remaining two permit variable-length lists. Scoring uses per-row precision/recall against a multi-alias gold set, macro-averaged across relations, with numeric answers matched under 5% relative tolerance.

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 535–542
October 28, 2026 ©2026 Association for Computational Linguistics

3 System

3.1 Base pipeline

For each query we sample k=5 few-shot exemplars from the training split via a hash of (seed, subject, relation), which guarantees exemplar selection is independent of batch composition and processing order, a property we found necessary after observing that a shared random-number stream produced different exemplars for the same row across code revisions, confounding otherwise-controlled comparisons.

For relations assigned to the chain-of-thought (CoT) path (Wei et al., 2022) (§3.2), each exemplar's question is paired not with its gold answer directly but with a *rationalization*: the model is asked, separately and once per unique exemplar (cached across queries), to explain why the given answer follows from the question, treating the answer as already known. This produces a demonstration of reasoning style without requiring us to write relation-specific chains of thought by hand. The query itself is then answered with the same reasoning format, terminating in a `Final Answer:` line that we parse.

For relations assigned to the direct path, exemplars show the gold answer with no reasoning, and the query is answered directly under a low token budget.

3.2 CoRe routing

We evaluated CoT and direct answering as global choices before finding that neither dominates: routing every relation to CoT reached macro-F1 0.511 on validation; routing every relation direct reached 0.510; but choosing the better mode independently per relation reached 0.542. Table 1 shows the per-relation split. CoRe routing is built on this per-relation pattern. Relations whose answer must be *constructed*, such as award recipient lists or a two-step "is this company listed, and if so where" decision, benefit from explicit reasoning space. The numeric relations (*hasArea*, *hasCapacity*) also route to CoT, but for a narrower reason: a single reasoning pass helps the model lay out the arithmetic and commit to a well-formed value. What hurts these relations is not one reasoning pass but *additional* reasoning—self-consistency sampling and larger budgets (§4)—because their residual errors are memorization-bound (§6). Only *countryLandBordersCountry* and *personHasCityOfDeath*, where reasoning earns nothing over direct answering, take the direct path.

Step 1: rationalize an exemplar

system: You are a careful research assistant. You will be given a factual question together with its correct answer. In 2–4 concise sentences, explain the reasoning that leads to this answer, as if working it out from scratch. Do not mention that the answer was provided to you.

user: Question: What is the total surface area of West Island in square kilometres… **Correct answer:** 6.23

→ *cached as this exemplar's rationale*

Step 2: answer the query

system: You are a knowledge base. First reason step by step, then on its own final line write 'Final Answer: ' followed by only the answer entities. Always commit to a specific value, even if you are unsure. Never answer None.

user: What is the total surface area of **West Island**…

assistant: West Island is the main island of the Cocos (Keeling) group. Sources give its area as about 6.2 square kilometres…
Final Answer: 6.23

… 4 more exemplars…

user: What is the total surface area of **Bequia**…

Figure 1: The two-step prompt for a CoT relation, abbreviated. Step 1 runs once per unique exemplar and is cached; step 2 assembles the cached rationales into a few-shot context. The final sentence of the step-2 system prompt is the never-abstain policy, applied only to the two relations whose gold is never empty.

Relation	CoT	Direct
awardWonBy	**0.123**	0.078
hasArea	**0.610**	0.480
hasCapacity	**0.180**	0.160
countryLandBordersCountry	0.939	**0.978**
personHasCityOfDeath	0.380	**0.500**
companyTradesAtStockExchange	0.624	0.624

Table 1: Macro-F1 by relation under a global CoT vs. direct policy (Qwen3.5-27B, validation split). No single global choice is optimal.

3.3 A confounded but instructive case: reasoning space for calibration

The *companyTradesAtStockExchange* tie in Table 1 initially suggested the relation was insensitive to reasoning. Inspecting raw model output revealed otherwise: under the direct path, the model's median output length was 4 characters. It emitted `"None"` immediately, with no room to execute the two-step decision our prompt asked for ("is this company publicly listed; if so, which exchange"). Rewriting the prompt to state the two-step decision explicitly, while leaving the relation on the direct path, raised macro-F1 from 0.624 to 0.653. Moving it to the CoT path so that the instruction had

room to execute raised it further to 0.674, with the predicted-empty rate falling from 57 to 40 against a gold-empty rate of 36. The ordering matters: the prompt rewrite alone recovered less than half the gain, because a template's requirements and its assigned decoding path were mismatched. The template asked for reasoning the token budget could not support.

3.4 Never-abstain fallback

For the two strictly-numeric relations, an empty prediction always scores zero (gold is never empty), while any relation with non-empty gold rows scores identically whether the model abstains or guesses wrong. We therefore compute the median value of the training split at initialization and substitute it whenever an answer fails to parse, guaranteeing zero abstentions on these two relations.

4 Negative Results

We report four interventions that failed or reversed under direct measurement.

Self-consistency hurts single-valued numeric recall. Motivated by the intuition that sampling and aggregating should reduce variance, we applied self-consistency (Wang et al., 2023): we generated 5 completions per query at temperature 0.7 for *hasArea* on Gemma-4-31B-it (our earlier base model) and took the median. This broke 4 of the 37 rows that greedy decoding answered correctly while rescuing only 1, for a net loss (macro-F1 0.37 $\rightarrow$ 0.34). Manual inspection showed why: sampled completions for a given entity tend to *agree* with each other around a single misremembered value rather than scattering randomly around the truth. On Isle of Wight, greedy decoding correctly answered 384 km^2 (gold: 380.16); all 5 sampled completions clustered near 984. This indicates the model's errors on this relation are confident but incorrect recalled values rather than uncertain guesses, a failure mode that resampling cannot correct because it is not variance-driven.

Guarding against confabulation can regress accuracy. For *personHasCityOfDeath*, we observed the model conflating a person's city of death with their birthplace, their country's capital, or the city where they were famous. We rewrote the prompt to explicitly forbid these substitutions. Measured on the full validation run, macro-F1 fell from 0.500 to 0.470, driven by predicted-empty rate rising from 63 to 79 against a gold-empty rate of 39: the guard suppressed correct low-confidence answers along with incorrect ones. We revert to the original template.

Token budget effects are confined to the construct-type relations. We swept the CoT reasoning budget over 300, 500, and 2000 tokens on the test split, holding batch composition and every other setting fixed (Table 2). Two results stand out, and the second corrects an expectation we held during development.

First, the gains are confined to exactly the two construct-type relations on the CoT path. *awardWonBy* improves by 1.4pp and *companyTradesAtStockExchange* by 2.2pp from 300 to 2000 tokens, which is the direction Table 1 predicts: a longer budget helps when the answer has to be assembled rather than recalled. Almost all of the *companyTradesAtStockExchange* gain arrives between 500 and 2000 tokens, consistent with the median-output-length finding in §3.2 that this relation needs room to execute a two-step decision before it commits to an answer.

Second, the other four relations are numerically *unchanged* across all three budgets, identical to four decimal places in precision, recall, and F1 alike. This includes *hasArea* and *hasCapacity*, which CoRe assigns to the CoT path and which therefore did receive the larger allowance. We had expected the extra reasoning space to hurt them, by the same mechanism as the self-consistency failure above, and it does not. The explanation is a property of greedy decoding: the token at each step is the argmax over the prefix, so any query that already emitted its `Final Answer:` line within 300 tokens produces a byte-identical continuation at 500 or 2000. A larger budget can only change queries that were previously truncated mid-reasoning, of which these relations had few. Raising the reasoning budget under greedy decoding is therefore a one-sided operation: it can recover truncated answers but cannot disturb answers that already completed, which makes it safe to tune on the construct-type relations without re-measuring the rest.

The aggregate effect is correspondingly small: 0.5359 to 0.5408 macro-F1 across a nearly sevenfold increase in the token budget, at a proportional increase in decoding cost. We report the 2000-token configuration as our primary test result, but note that 300 tokens recovers 99.1% of its

	Max new tokens		
Relation	300	500	2000
CoT path			
awardWonBy	0.1524	0.1560	**0.1659**
companyTradesAtStockExchange	0.6833	0.6833	**0.7050**
hasArea	0.5600	0.5600	0.5600
hasCapacity	0.1327	0.1327	0.1327
Direct path			
countryLandBordersCountry	0.9657	0.9657	0.9657
personHasCityOfDeath	0.5100	0.5100	0.5100
All relations	0.5359	0.5360	**0.5408**
Zero-object cases*	0.7250	0.7250	**0.7273**

Table 2: Test-split macro-F1 by relation across CoT reasoning budgets under CoRe routing (Qwen3.5-27B). Only the two construct-type relations (*awardWonBy*, *companyTradesAtStockExchange*) respond to the budget. *hasArea* and *hasCapacity* are on the CoT path but already emit their answer within 300 tokens, so greedy decoding gives identical output at every budget; the two direct-path relations never receive the CoT allowance. *Zero-object cases are the subset of queries whose gold answer is the empty set, scored separately.

macro-F1, so the budget is not where the remaining headroom lies.

Small validation splits are not interchangeable with full-run scores. Twice during development we measured a relation in isolation (to reduce iteration cost) and compared the result against a baseline drawn from a full run over all six relations, concluding an intervention had helped or hurt when it had not. For *hasCapacity*, the full-run baseline (0.180) and the isolated-relation baseline (0.150) differ by 3 percentage points under an *identical* configuration; for *personHasCityOfDeath*, the gap was 5 points (0.500 full-run vs. 0.450 isolated). We traced this to batch composition: with greedy decoding, left-padding length depends on the other rows sharing a decoding batch, which depends on which relations are active in a given run. Moving one relation between reasoning paths shifted *other, untouched* relations' scores by up to 5 points purely through this channel. We recommend that KBC systems evaluated with batched greedy decoding treat cross-run relation-level comparisons as valid only when batch composition is held fixed, and report this as a source of measurement noise independent of the intervention under test.

5 Results

Table 3 compares our staged configurations against the model-swap baseline. The largest gain in our study came from changing the base model within the parameter budget rather than from any prompting intervention: under an otherwise-identical rationalized-CoT pipeline, Gemma-4-31B-it answered 37/100 *hasArea* queries within the 5% numeric tolerance, against 61/100 for Qwen3.5-27B. This 24-point swing exceeds the combined effect of every prompting intervention we measured on any single relation, and is consistent with our finding that errors on this relation are memorization failures rather than reasoning failures: a different model has different memorized facts, but no amount of reasoning scaffolding recovers a fact neither model stores.

Configuration	Macro-F1
Gemma-4-31B-it, all-CoT (baseline)	0.522
Qwen3.5-27B, all-CoT	0.511
Qwen3.5-27B, all-direct	0.510
Qwen3.5-27B, CoRe routing	0.542
+ rewritten *company* prompt	**0.548**

Table 3: Validation macro-F1 across pipeline configurations. CoRe routing outperforms either global policy under the same model, and the model swap alone is responsible for the majority of the gain over the original baseline on knowledge-bound relations. All figures here are validation. On the test split the same pipeline with *company* on the CoT path scores 0.5408 (Table 2); we did not measure the direct-*company* variant on test, so the two splits are not directly comparable row for row.

6 Error Analysis

Numeric relations degrade with entity obscurity Accuracy on *hasArea* is almost monotonic in the size of the entity being asked about (Table 4). Every country-scale entity is answered within tolerance; small islands and lakes are answered correctly less than half the time. Since the arithmetic involved is identical across buckets, and the prompt states the unit conversions explicitly, this gradient is not a reasoning or formatting effect. It tracks how often the entity is likely to have appeared in pretraining.

Inspecting the reasoning traces confirms this. On the failures the model does not hedge or refuse; it commits to a specific figure and then reasons fluently about its precision. For Hopen it answers 1130 km^2 against a gold value of 47, noting that the figure "satisfies the requirement for three significant figures"; for Gogland it answers 142.5 against 21, again arguing about significant figures rather than about the island. For Donauinsel it recalls an area in hectares, converts correctly using the factor

Gold area (km^2)	n	Within 5%
<10	13	38%
10–100	21	43%
100–1k	29	66%
1k–100k	33	73%
>100k	4	100%

Table 4: *hasArea* accuracy by the magnitude of the gold value (Qwen3.5-27B, validation). Larger entities are more prominent in pretraining and are answered far more reliably, though the extraction problem is identical.

supplied in the prompt, and arrives at 10.0 against a gold value of 3.9: the arithmetic is right and the remembered quantity is wrong. These are retrieval errors dressed in correct procedure, which is why the scaffolding interventions in §4 could not touch them, and why the model swap in §5 could.

hasCapacity shows the same structure more severely: the venues in this split are largely high-school, minor-league and small-college grounds, and the model routinely produces a confident, precisely-stated figure for a venue it has evidently confused with another. For "Melaleuca Field in Idaho", a minor-league baseball park seating 3400, it reports that the total maximum capacity "is officially published as 16,112".

Abstention is already near-optimal

Three relations permit an empty gold set, and the scoring rule makes the trade-off asymmetric: on a row whose gold is non-empty, abstaining and guessing wrong both score zero, whereas on a row whose gold is empty, any guess scores zero where abstaining would have scored one. Guessing is therefore free on non-empty rows and expensive on empty ones.

This asymmetry makes it tempting to push the model toward answering more often, and we tried it. The break-even accuracy required is higher than it appears. Converting the w rows where the model wrongly abstains yields $w \cdot a$ points at answering-accuracy a, but any prompt change general enough to do so also disturbs the c rows where it abstains correctly. Break-even requires $a > c/w$. On *personHasCityOfDeath* the model answers with 41% accuracy against $c/w = 36/27$, so an intervention would need 133% accuracy to pay for itself; on *companyTradesAtStockExchange* it answers with 87% accuracy against $c/w = 32/25$, requiring 128%. On both relations the operation cannot break even at any accuracy the model could attain. Our confabulation-guard experiment (§4) is the empirical version of this calculation: it moved abstention in the intended direction and lost three points.

We take this as evidence that the abstention behaviour of an instruction-tuned model on this task is already close to a local optimum, and that the remaining headroom on these relations is in *which* rows it abstains on rather than how often. Improving that requires better-calibrated knowledge, not a better-worded instruction.

7 Related Work

Biester et al. (2023) apply prompt ensembling to Llama2 models at 7B/13B/70B scale, measuring a consistent 4.7–6.1 point gain from ensembling over the single best prompt at every model size, comparable in magnitude to a roughly 10x increase in parameter count in their setting. Their relation-level results include *PersonHasPlaceOfDeath*, directly comparable to our *personHasCityOfDeath*: they report a 14-point F1 swing (0.444 vs. 0.303) between two phrasings of the same question, and describe a fact-probing step for death-related relations that first asks whether the subject is deceased before eliciting a location, a two-step decomposition we independently arrived at for the same relation via direct error analysis, which we take as convergent evidence that this decomposition is a property of the task rather than of either system. Alivanistos et al. (2022) similarly decompose relation extraction into multi-step prompting chains, and report that the empty-answer case is a distinct failure mode requiring its own treatment rather than falling out of the extraction step, which matches what we find in §6.

Prompting versus training. Li et al. (2022) took a different approach entirely, using task-specific pretraining rather than prompting, and won Track 1 of the original ISWC 2022 challenge. Zhang et al. (2023) likewise treat the problem as knowledge engineering over Wikidata rather than as prompt design. Both are reminders that our findings are specific to the prompting-only regime imposed by this task's parameter constraint, not a claim that prompting is the best available lever in general. Notably, Biester et al. (2023) report that fine-tuning *underperformed* their non-fine-tuned model in the same prompt-ensembling setup (0.562 vs. 0.589), so the ordering between the two approaches appears to depend on the model and the budget rather than being fixed.

Knowledge as the limiting factor. The framing of language models as knowledge bases originates with Petroni et al. (2019), whose LAMA probe assumed single-token, single-answer relations. The shared task relaxes both assumptions, and our results suggest that the harder setting exposes where the analogy breaks down: a knowledge base can be queried for an obscure entity as reliably as for a famous one, whereas the model's accuracy on *hasArea* falls from 100% to 38% as entity prominence decreases (Table 4) with no change in the extraction problem itself. This is consistent with reports that recall of long-tail facts scales with pretraining frequency and parameter count more than with inference-time technique (Kandpal et al., 2023; Mallen et al., 2023), and it is the reason we treat the 32B ceiling, rather than our prompting choices, as the binding constraint on this system.

8 Conclusion

We built a knowledge base construction system under a 32B budget on total inference-time parameters, which rules out a larger single model and caps any ensemble at the same 32B total. Within that constraint we found that no single prompting policy is optimal across relations: the better of chain-of-thought and direct answering is relation-specific, and routing on it (CoRe) outperforms either global policy by three points.

The more useful finding for future systems may be the negative one. Across a dozen interventions, everything that operated on the scaffolding around the model, including self-consistency sampling, exemplar selection, reasoning budget, and several prompt rewrites, moved the score by less than the run-to-run variation induced by batch composition. The one change that moved it substantially was replacing the base model with a different one of comparable size. On *hasArea* that swap was worth 24 points where no prompting change was worth three. We read this as evidence that at this scale the task is limited by which facts a model has memorised rather than by how effectively those facts are elicited, and that reporting practice should account for measurement noise at the few-point level before attributing gains to technique.

9 Limitations

Our routing decisions were tuned on a validation split of under 500 rows total (10–100 per relation), and §4 documents that even same-configuration reruns can differ by several points due to batch-composition effects; some of the per-relation deltas we report as evidence for routing decisions are close to this noise floor and should be treated as directional rather than precise. Our self-consistency finding is demonstrated on two numeric relations from one task instance; we do not verify it holds for numeric relations in general. The reasoning-budget sweep in Table 2 is a single run per budget with no seed variance, and its two non-zero deltas (1.4pp and 2.2pp) are within the range that §4 shows batch composition alone can produce, so the ranking of the three budgets should be read as directional rather than established. The 32B constraint that motivates this study is specific to our shared-task track and may not transfer to settings without it. Finally, our model-swap finding (different models recover different facts) suggests that an ensemble of two or more models sharing the 32B budget might outperform routing within a single model. The task rules permit such within-budget ensembles; we did not test one, and see it as the most direct follow-up to this work.

References

Dimitrios Alivanistos, Selene Báez Santamaría, Michael Cochez, Jan-Christoph Kalo, Emile van Krieken, and Thiviyan Thanapalasingam. 2022. Prompting as probing: Using language models for knowledge base construction.

Fabian Biester, Daniel Del Gaudio, and Mohamed Abdelaal. 2023. Enhancing knowledge base construction from pre-trained language models using prompt ensembles. In *Proceedings of the Knowledge Base Construction from Pre-trained Language Models Workshop (KBC-LM'23) at ISWC 2023*, volume 3577. CEUR-WS.org.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Tianyi Li, Wenyu Huang, Nikos Papasarantopoulos, Pavlos Vougiouklis, and Jeff Z. Pan. 2022. Task-specific pre-training and prompt decomposition for knowledge graph population with language models. Winner, Track 1, LM-KBC Challenge 2022.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases?

Sneha Singhania, Jan-Christoph Kalo, Simon Razniewski, and Jeff Z. Pan. 2023. Knowledge base construction from pre-trained language models (lm-kbc). In *Preface: LM-KBC Challenge 2023*. CEUR-WS.org.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Bohui Zhang, Ioannis Reklos, Nitisha Jain, Albert Meroño Peñuela, and Elena Simperl. 2023. Using large language models for knowledge engineering (llmke): A case study on wikidata.

A Full Score Tables

Table 5 gives per-relation macro-F1 for every configuration in the CoRe lineage on the validation split, and Table 6 places those numbers beside two earlier experimental lineages that we ran on the same task with different pipelines.

The cross-lineage comparison carries an important caveat. The three groups in Table 6 differ in more than their base model: the Gemma sampling lineage draws 20 (or 5) samples per numeric query and routes four string relations through a yes/no self-consistency gate, and the Qwen3-32B and Qwen3.5-9B lineages use a different codebase again, with relation-specific numeric prompting tiers. Differences across groups therefore confound model identity with pipeline design and should not be read as evidence about either one alone. Comparisons within a group are controlled and interpretable; comparisons across groups are reported only for completeness.

Two entries are worth isolating because they bear on the paper's central claim. Gemma-4-31B and Qwen3.5-27B under the *identical* CoRe pipeline (rows 1 and 2 of Table 5) differ by 24 points on *hasArea* (0.370 vs. 0.610) while moving in the opposite direction on *companyTradesAtStockExchange* (0.774 vs. 0.624). This is the controlled version of the model-swap result reported in §5. Separately, the Gemma sampling lineage reached 0.755 on *companyTradesAtStockExchange* with none of our prompt work, close to the 0.774 Gemma reaches under CoRe and well above every Qwen configuration, which suggests abstention behaviour on that relation is driven more by the base model than by the scaffolding around it.

	award	company	borders	area	capacity	death
Gemma CoT	.150	.774	.980	.370	.200	.470
Qwen all-CoT	.123	.624	.939	.610	.180	.380
Qwen all-direct	.078	.624	.978	.480	.160	.500
Qwen CoRe	.119	.624	.978	.610	.180	.500
+prompt	.119	.653	.978	.610	.180	.500
+prompt+CoT	.123	.674	.978	.610	.180	.470

Table 5: Per-relation macro-F1, CoRe lineage, validation split. Overall macro-F1: Gemma CoT 0.522, Qwen all-CoT 0.511, Qwen all-direct 0.510, Qwen CoRe 0.542, +*prompt* (rewritten *company* template, relation left on the direct path) 0.548, +*prompt+CoT* (same template, relation moved to the CoT path) 0.546. The last two differ only in which path *company* takes: moving it to CoT gains 2.1pp on that relation but loses 3.0pp on *personHasCityOfDeath*, which shares its batch group. Rows 1 and 2 share an identical pipeline and differ only in base model.

System	Lineage	Overall
Qwen3.5-27B +prompt	CoRe	**0.548**
Qwen3.5-27B +prompt+CoT	CoRe	0.546
Qwen3.5-27B CoRe	CoRe	0.542
Gemma-4-31B CoT	CoRe	0.522
Qwen3.5-27B all-CoT	CoRe	0.511
Qwen3.5-27B direct	CoRe	0.510
Gemma 4 31B n=5	Gemma sampling	0.507
Gemma 4 31B n=20	Gemma sampling	0.466
Tier-1 units	Qwen3-32B	0.454
Official baseline	Qwen3-32B	0.446
Agentic multi-step	Qwen3-32B	0.403
CoT (hasArea-only)	Qwen3.5-9B	0.420
Baseline	Qwen3.5-9B	0.407

Table 6: Overall macro-F1 across three experimental lineages on the validation split, best two or three configurations per lineage. Lineages differ in pipeline as well as model (see text), so cross-lineage rows are not a controlled comparison.

B Reproducibility

Our code, configurations, and prompt templates are available at `https://github.com/MightBeFonias/CoRe`. All results use greedy decoding with k=5 few-shot exemplars sampled by

a hash of (seed, subject, relation) with seed 42, so exemplar selection is reproducible and independent of batch composition. The CoRe configuration assigns *hasArea*, *hasCapacity*, *awardWonBy* and *companyTradesAtStockExchange* to the CoT path and the remaining two relations to the direct path, with a 2000-token reasoning budget on the CoT path and 64 tokens on the direct path. Numeric relations fall back to the training-split median when an answer fails to parse. Batch size is 4 throughout; §4 documents why this value must be held fixed across runs that are to be compared.

Ensembling Elicitation Strategies for Knowledge Base Construction

Vlad-Mădălin Ungureanu[1] **Matei Neacşu**[2*] **Ianis Belu**[2*]
[1]National Institute for Research and Development in Informatics – ICI Bucharest
[2]International Computer High School of Bucharest (ICHB)
madalin.ungureanu@ici.ro
neacsu.matei@ichb.ro
belu.ianis@ichb.ro

Abstract

Large language models represent a promising alternative for automating structured knowledge bases construction, but their reliance on parametric knowledge makes them susceptible to hallucinations and incomplete information. The Automated Knowledge Base Construction competition investigates the use of large language models to construct structured knowledge bases from their parametric knowledge alone, without fine-tuning or external retrieval. Given a subject-relation pair, participants must predict the complete set of correct objects across several relation types. Our approach proposes an ensemble of four prompting strategies: question-style, masked-style, few-shot chain-of-thought, and least-to-most prompting. It aggregates their predictions using relation-aware majority voting. The resulting system achieves a Macro-F1 of 0.6651 on the test set, substantially outperforming the provided baseline of 0.294. The system implementation can be found at the GitHub repository referenced below.[1]

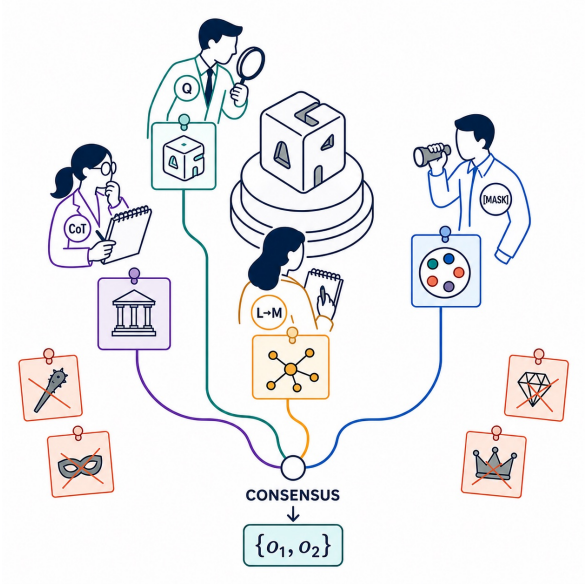


Figure 1: Multiple elicitation strategies examine the same fact from different perspectives. Their observations are combined through relation-aware aggregation to produce the final knowledge-base prediction.

1 Introduction

Structured knowledge bases store verified information, enabling LLMs to ground their outputs in factual data and thereby reduce hallucinations and improve reliability (Yang et al., 2025). Alternatively, some research explores constructing knowledge bases entirely from LLMs, though such systems still face challenges regarding precision (Hu et al., 2024).

Building on this line of work, the Automated Knowledge Base Construction shared task focuses specifically on leveraging Large Language Models to construct structured knowledge bases. The challenge lies in the requirement that predictions must be generated using the model's parametric memory alone, without fine-tuning or Retrieval-Augmented Generation (RAG) over external corpora. Additionally, although multiple LLMs may be combined within a single answer prediction system, the summed parameter count of the combined models must not exceed 32B. Given a subject s and a relation r, the task is to predict the complete set of correct object strings $\{o_1, o_2, \ldots, o_k\}$, where there may be zero, one or multiple correct answers.

The dataset includes six distinct relation types, each capturing specific entity attributes or associations with varying cardinalities. First, `awardWonBy` identifies the recipients of a specified award, a relation that often exhibits high cardinality because many awards are conferred upon dozens or hundreds of entities over time. In geographic contexts, `countryLandBordersCountry` links a country to the nations sharing its land borders (yielding

[*] Work conducted during an internship at ICI Bucharest.
[1] https://github.com/madalinmdv/akbc-2026

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 543–549
October 28, 2026 ©2026 Association for Computational Linguistics

an empty set for borderless countries or multiple values otherwise), while `hasArea` specifies a geographic entity's surface area in square kilometres. Commercial and venue-specific data are captured by `companyTradesAtStockExchange`, which maps a company to the stock exchange(s) where its shares are publicly traded (with unlisted subsidiaries yielding empty sets), and `hasCapacity`, which denotes the maximum spectator capacity of a venue. Finally, `personHasCityOfDeath` records the city where an individual passed away, remaining empty if the person is still living.

The proposed system employs multiple prompting strategies and aggregates the resulting predictions according to the relation type in order to generate a final output. Specifically, multiple prediction sets produced by the same model under different prompting strategies are combined to distinguish stable parametric knowledge from prompt-induced hallucinations. The chosen model for this task is Meta's 30B parameter Muse Glimmer.

This study aims to determine the extent to which different prompting strategies influence the quality and precision of model outputs, and to assess whether aggregating multiple responses generated by the same language model can help reduce hallucinations.

2 Related work

Considering the competition constraints in regards to fine-tuning the model or designing RAG implementations using external corpora, the workshop motivates a greater reliance on how effectively parametric knowledge can be elicited from the model. Under these circumstances, prompt-based strategies become one of the main pillars in the attempt to improve the reliability of the generated knowledge base. Prompting methods encompass a wide range of techniques, from foundational approaches such as few-shot and chain-of-thought prompting to more advanced methods (Sahoo et al., 2024).

Few-shot prompting refers to providing a language model with a small number of task-specific demonstrations at inference time, without updating the model's parameters (Brown et al., 2020). Demonstrations consist of paired inputs and their corresponding desired outputs, allowing the model to infer the task format. The study by Min et al., "Rethinking the Role of Demonstrations," shows that demonstrations derive their value not from teaching the model new input-output mappings, but from exposing it to the label space, the distribution of the input text, and the overall format of the sequence (Min et al., 2022).

Chain of thought prompting consists of including a series of intermediate reasoning steps in the input. This method has been proven to significantly improve the ability of large language models to perform complex reasoning (Wei et al., 2022). Automatic Chain-of-Thought prompting utilises LLMs to automatically generate reasoning chains for selected examples, with diversity in the sampled questions helping mitigate errors in the generated chains (Zhang et al., 2022). Participants in last year's knowledge base construction workshop, Edouard Albert-Roulhac and Amal Zouaq, utilized a large model to compose a comprehensive set of chain-of-thought style prompts (Albert-Roulhac and Zouaq, 2025). These examples were then used as few-shot demonstrations to produce multiple reasoning paths and answers during inference.

Self-refinement represents a method of improving LLM outputs through self-generated feedback. Madaan et al. showcased this method, drawing an analogy to how humans reflect on their own writing (Madaan et al., 2023). In their study, after an LLM produces an initial output, the same model evaluates it and, based on that feedback, refines its response. This process is repeated iteratively until the model's own feedback indicates that no further improvement is needed, or a fixed number of iterations is reached. This workflow assigns the LLM with multiple roles, acting as both generator and critic.

Least-to-most prompting solves complex problems by decomposing them into simpler subproblems (Zhou et al., 2022). Unlike standard chain-of-thought prompting, which reasons in a single pass, this approach decouples planning from execution via a two-stage process: first, it decomposes the initial query into subquestions, and second, it sequentially solves each subquestion using the accumulated history of previous answers.

3 System

The system ensembles the elicitation techniques described below using a 30B parameter model.

The results are then aggregated by majority vote, except for the `awardWonBy` relation, where the large number of correct objects per subject makes voting across strategies too restrictive.

Technique	Description
`Question style prompt`	Zero-shot, natural question.
`Masked style prompt`	Zero-shot; the same fact phrased as a sentence with a `[MASK]` token.
`Few-Shot CoT`	Few-shot chain-of-thought; hand-written worked examples per relation, provided in a single turn.
`Least-to-most prompting`	A chain of per-relation subquestions, each posed as its own conversational turn, with answers accumulating before the final query.

Table 1: Prompting techniques used in the ensemble, with a description of each.

3.1 Question and masked style prompting

We adopt question-style and masked-style prompting using templates provided by the task organizers.[2] These are used as a baseline against which the gains from more elaborate prompting strategies can be assessed.

Relation	Prompt
hasArea	What is the area of {subject_entity} in square kilometers?
hasCapacity	What is the total capacity of {subject_entity}?
countryLandBordersCountry	Which countries share a land border with {subject_entity}?
personHasCityOfDeath	In which city did {subject_entity} die?
awardWonBy	Who won the {subject_entity}?
companyTradesAtStockExchange	Where do shares of {subject_entity} trade?

Table 2: Zero-shot question prompt templates used for each relation.

Relation	Prompt
hasArea	The area of {subject_entity} in square kilometers is {mask_token}.
hasCapacity	The total capacity of {subject_entity} is {mask_token}.
countryLandBordersCountry	{subject_entity} shares a land border with {mask_token}.
personHasCityOfDeath	{subject_entity} died in the city of {mask_token}.
awardWonBy	{subject_entity} was won by {mask_token}.
companyTradesAtStockExchange	{subject_entity} shares can be traded at {mask_token}.

Table 3: Masked-style prompt templates.

3.2 Few-shot chain-of-thought prompting

The system presented in this article combines few-shot demonstrations with human-authored chain-of-thought reasoning traces derived from examples in the training data.

Unlike zero-shot strategies, which elicit an answer directly, this approach provides the model with manually constructed worked examples, in which each answer is preceded by an explicit reasoning trace, as illustrated in Figure 2.

```
Question: On which stock exchanges does Bangladesh
↪ Cement
Manufacturers Association trade?
Step 1 - The BCMA is a trade association
↪ representing cement producers in
Bangladesh, not a commercial corporation or
↪ operating company.
Step 2 - As a membership-based industry body, it
↪ does not issue equity shares
to the public and is therefore not publicly listed
↪ on any exchange.
Step 3 - Since it is not a publicly traded entity,
↪ it has no listings.
Answer: []

[...]
```

Figure 2: Chain-of-thought demonstration for the `companyTradesAtStockExchange` relation.

3.3 Least-to-most prompting

Least-to-most prompting decomposes each query into a sequence of simpler subquestions that are answered before the original question is posed. In contrast to the chain-of-thought strategy of Section 3.2, where the reasoning is produced in a single turn, each subquestion is here issued as its own conversational turn, and every subanswer is appended to the dialogue history before the next subquestion is asked.

The subquestion chains mirror the reasoning steps of the chain-of-thought schema. The chain for `personHasCityOfDeath`, shown in Figure 3, is representative, and is constructed so that the confusion between the city a person is famous for and the city in which they died is raised explicitly before the final answer is requested.

Each chain is preceded by worked dialogues demonstrating the expected form of the exchange. One such dialogue is shown, truncated, in Figure 4.

```
SUBQUESTIONS_DEATH = [
   "Who is or was {subject}, and has {subject} died
   ↪ ?",
   "Which city is {subject} most famous for, or most
   ↪  associated with through living or working
   ↪ there?",
   "If {subject} has died, in which city did the
   ↪ death actually take place, and is that the
   ↪ same city as the one just named or a different
   ↪  one?",
]
```

Figure 3: The least-to-most subquestion chain for the `personHasCityOfDeath` relation.

[2] https://github.com/lm-kbc/dataset2026

```
Problem: In which city did Sigmund Freud die?
Subquestion 1: Who is or was Sigmund Freud, and has
↪ Sigmund Freud died?
Subanswer 1: Sigmund Freud was the founder of
↪ psychoanalysis; he died in 1939
and is deceased.
Subquestion 2: Which city is Sigmund Freud most
↪ famous for [...]?
Subanswer 2: He is overwhelmingly associated with
↪ Vienna, where he lived and
worked for most of his life.
Subquestion 3: If Sigmund Freud has died, in which
↪ city did the death actually
take place, and is that the same city as the one
↪ just named or a different one?
Subanswer 3: He fled Vienna in 1938 after the Nazi
↪ annexation of Austria and
died the following year in London, so the death city
↪  differs from the famous
city.
Final question: In which city did Sigmund Freud die?
Answer: ["London"]

[...]
```

Figure 4: A worked least-to-most demonstration for the `personHasCityOfDeath` relation.

3.4 Self-refinement

Observing the results obtained for the `awardWonBy` relation under the other prompting strategies, it was concluded that a different approach is needed for this particular relation. In this version of the self-refinement approach, the model first iteratively expands its candidate list of award recipients across follow-up rounds to maximise recall, then self-reviews and prunes that list to remove low-confidence names, trading a modest drop in recall for a much larger gain in precision.

The first query asks the model to generate all the recipients it knows of for a given award. To improve recall, the generated answers are then given back to the model in the follow-up query shown in Figure 5, asking it to complete the list with any additional winners.

```
FOLLOW_UP_QUERY = (
   "Recipients of {award} identified so far:\n{
   ↪ listed}\n\n"
   "Who ELSE has received {award} that is NOT in the
   ↪  list above? "
   "Only add names you are genuinely confident about
   ↪ , and do not repeat any "
   "name already listed. If you cannot think of any
   ↪ more, reply with exactly: "
   "Answer: none"
)
```

Figure 5: Follow-up query used to expand the candidate list for the `awardWonBy` relation.

This process iterates for up to six additional rounds, continuing as long as the model generates more than two new names in the previous round. To improve precision, a subsequent pruning step was introduced, in which the model uses the system prompt shown in Figure 6 to review its own previously generated predictions and remove any it is confident are incorrect.

```
SYSTEM_PROMPT_PRUNE = (
   "You are a strict fact-checker for award
   ↪ recipients. You are shown a "
   "candidate list and you name only the ones you
   ↪ are EXTREMELY confident "
   "never received the award -- not the ones you
   ↪ merely cannot confirm or "
   "are unsure about, which you leave alone. Provide
   ↪  only the requested "
   "information without explanations, uncertainty
   ↪ statements, or additional "
   "context. "
)
```

Figure 6: System prompt used to prune low-confidence candidates for the `awardWonBy` relation.

3.5 Post-processing: answer extraction

A single answer-extraction module is applied uniformly across all elicitation methods, ensuring that score differences arise solely from the chosen prompting strategies rather than from differences in post-processing. Each strategy's final query includes explicit formatting instructions, determined by the relation type:

1. `Answer:` `<number>` for numeric relations, such as `hasArea` or `hasCapacity`;
2. `Answer:` `[...]` — a JSON array — for list relations;
3. `Answer:` followed by one name per line, specifically for the `awardWonBy` relation.

The parsing stage reads only the text following the final `Answer:` marker. For numeric relations, it first checks for a comma-grouped number and falls back to a plain number only if none is found, so that a value such as 51,286 is parsed in full rather than truncated at the first comma. For list relations, the segment is parsed as a JSON array, falling back to comma-splitting when the model's array is malformed. Replies indicating no objects — "none," "n/a," and similar — are mapped to an empty set, which for many subject–relation pairs is the correct answer rather than a failure to respond.

3.6 Answer aggregation

Result aggregation is performed through majority voting, subject to the following relation-specific rules:

1. **Numeric relations:** Responses within a 5% tolerance of one another are grouped into clusters, and the mean value of the largest cluster containing at least two items is selected as the final prediction. When responses are too scattered for a cluster to form, a fallback mechanism selects the prediction from one of the four prompting strategies.

2. **List relations:** A quorum of two is required for a value to be included in the final prediction. If no consensus is reached, the predicted set is left empty. This is a deliberate design choice, since an empty set is itself a valid answer for a considerable share of the dataset: approximately 20% of the validation triplets have an empty set as the correct answer.

3. **`awardWonBy`:** This relation is excluded from the ensemble entirely, as aggregating results across strategies would significantly reduce recall.

4 Results

The evaluations presented in this section exclude the `awardWonBy` relation. This is because the prompting strategies described for the rest of the system did not yield satisfactory results for this relation, and including it in the aggregated metrics led to a considerable drop in recall. The evaluation of `awardWonBy` is therefore presented separately.

Table 4 compares the two zero-shot strategies. Overall, question-style prompting yields a marginally higher Macro-F1 (0.667) than masked-style prompting (0.658). Therefore, in a zero-shot setting, eliciting facts through direct questions is slightly more effective.

Relation	P	R	F1	Empty
Question-style prompting				
companyTradesAtStockExchange	0.950	0.758	0.759	54
countryLandBordersCountry	0.998	0.987	0.991	18
hasArea	0.820	0.820	0.820	0
hasCapacity	0.289	0.289	0.289	0
personHasCityOfDeath	0.880	0.610	0.570	66
All	–	–	**0.667**	138
Masked-style prompting				
companyTradesAtStockExchange	0.910	0.732	0.712	53
countryLandBordersCountry	0.996	0.993	0.994	18
hasArea	0.800	0.800	0.800	0
hasCapacity	0.361	0.361	0.361	0
personHasCityOfDeath	0.840	0.590	0.520	64
All	–	–	**0.658**	135

Table 4: Evaluation results for the two zero-shot strategies on the validation set, over the five relations handled by the ensemble. `awardWonBy` is excluded, as it is generated by the dedicated strategy of Section 3.4.

Few-shot chain-of-thought prompting achieves an F1 score of 0.680, outperforming both zero-shot strategies. The improvement is most pronounced for `companyTradesAtStockExchange`, whose F1 score rises from 0.759 with question-style prompting to 0.847 with few-shot chain-of-thought prompting. This suggests that the reasoning traces provided in the demonstrations are particularly effective at guiding the model when the extraction task requires disambiguating between multiple candidate entities.

However, this improvement is not uniform across all relations. `hasCapacity` performs worse under chain-of-thought prompting than under either zero-shot strategy.

Relation	P	R	F1	Empty
companyTradesAtStockExchange	0.937	0.852	0.847	45
countryLandBordersCountry	1.000	0.992	0.995	18
hasArea	0.810	0.810	0.810	0
hasCapacity	0.278	0.278	0.278	0
personHasCityOfDeath	0.880	0.600	0.560	67
All	–	–	**0.680**	130

Table 5: Evaluation results for few-shot chain-of-thought prompting on the validation set.

Least-to-most prompting is the weakest of the four strategies, with a Macro-F1 of 0.647. The comparison with few-shot chain-of-thought prompting is controlled, in that the two strategies employ similar reasoning decomposition and differ only in whether it is delivered in a single turn or across several. Least-to-most is inferior on every one of the five relations; for example, it reaches 0.812 against 0.847 on `companyTradesAtStockExchange`.

Relation	P	R	F1	Empty
companyTradesAtStockExchange	0.913	0.835	0.812	45
countryLandBordersCountry	1.000	0.988	0.993	18
hasArea	0.800	0.800	0.800	0
hasCapacity	0.237	0.237	0.237	0
personHasCityOfDeath	0.720	0.600	0.490	51
All	–	–	**0.647**	114

Table 6: Evaluation results for least-to-most prompting.

The aggregated results achieve a Macro-F1 of 0.689, marginally outperforming the best individual strategy observed previously: 0.680 with few-shot chain-of-thought prompting. This suggests that the overall gain in Macro-F1 stems from more balanced performance across relations rather than from consistently matching the best individual result for each relation.

Relation	P	R	F1	Empty
companyTradesAtStockExchange	0.950	0.838	0.839	46
countryLandBordersCountry	1.000	0.992	0.995	18
hasArea	0.810	0.810	0.810	0
hasCapacity	0.330	0.330	0.330	0
personHasCityOfDeath	0.850	0.620	0.560	62
All relations	0.776	0.701	**0.689**	126

Table 7: Validation results after relation-aware aggregation.

Turning to the `awardWonBy` relation, the self-refinement approach achieves a Macro-F1 of 0.209 on the validation set, more than double the 0.094 F1, which was the score obtained by the best-performing of the four other prompting strategies described earlier.

Relation	P	R	F1	Empty
awardWonBy	0.329	0.196	0.209	–

Table 8: Validation results for the separately evaluated `awardWonBy` relation.

On the hidden test set, the proposed implementation obtained a Macro-F1 score of 0.6651, as shown in Table 9. Among the evaluated relations, `hasCapacity` achieved the lowest F1 score; this limitation may arise because the relevant information was not included in the model's training data. In contrast, the highest score was obtained for `countryLandBordersCountry`.

Relation	P	R	F1
awardWonBy	0.5179	0.2850	0.3357
companyTradesAtStockExchange	0.9433	0.8327	0.8105
countryLandBordersCountry	0.9861	0.9702	0.9680
hasArea	0.8200	0.8200	0.8200
hasCapacity	0.2857	0.2857	0.2857
personHasCityOfDeath	0.9150	0.6000	0.5667
All relations	0.7728	0.6760	**0.6651**

Table 9: Per-relation results on the hidden test set.

4.1 Observations

The ensemble strategy was also applied to prediction sets generated by a combination of smaller models: Mistral-Nemo:12b, Llama3.1:8b, and Gemma2:9B. Although the single-model system presented in this paper achieved a higher overall Macro-F1, aggregating predictions across these smaller, heterogeneous models yielded a proportionally larger improvement in Macro-F1 than aggregating predictions from a single model alone.

This leads to a second observation: given the constraints against adjusting model parameters or drawing on external information, the correctness of the generated knowledge base is fundamentally bounded by whether the sought-after fact was present in the large language model's training data.

Additionally, it should be noted that the competition baseline was produced using Qwen3.5-9B while our results were obtained with the considerably larger Muse Glimmer 30B, which limits the extent to which the results can be interpreted as a direct comparison of the proposed approach against the baseline.

5 Conclusions

This paper investigated how different elicitation strategies affect parametric-only knowledge-base predictions and whether aggregating single-model responses reduces hallucinations. Across the four evaluated strategies, few-shot chain-of-thought achieved the highest overall Macro-F1, narrowly outperforming two zero-shot approaches, whereas multi-turn least-to-most prompting consistently scored lowest. Finally, the high-cardinality `awardWonBy` relation proved unsuited to majority voting, requiring a dedicated expansion-and-verification procedure instead.

Acknowledgements

The authors acknowledge the support of project PN 23 38 01 01, "Contributions to the consolidation of emerging technologies specific to the Internet of Things and complex systems," which provided technical resources essential to this research.

References

Wenli Yang, Lilian Some, Michael Bain, and Byeong Kang. 2025. A Comprehensive Survey on Integrating Large Language Models with Knowledge-Based Methods. *arXiv preprint arXiv:2501.13947*.

Yujia Hu, Tuan-Phong Nguyen, Shrestha Ghosh, and Simon Razniewski. 2024. Enabling LLM Knowledge Analysis via Extensive Materialization. *arXiv preprint arXiv:2411.04920*.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. *arXiv preprint arXiv:2402.07927*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu,

Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? *arXiv preprint arXiv:2202.12837*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic Chain of Thought Prompting in Large Language Models. *arXiv preprint arXiv:2210.03493*.

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-Wise Self-consistency for LLM Probing. In *Proceedings of the KBC-LM Workshop and the LM-KBC Challenge 2025*, CEUR Workshop Proceedings, Vol-4041.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv preprint arXiv:2303.17651*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. *arXiv preprint arXiv:2205.10625*.

DataMind at AKBC Shared Task 2026: Relation-Specific Multiple Prompting for Closed-Book Knowledge Base Completion

Tanaree Patcharatrakul[1], **Peerapon Vateekul**[2]*
[1]International School of Engineering, Faculty of Engineering,
Chulalongkorn University, Thailand
[2]Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Thailand
6738108421@student.chula.ac.th, peerapon.v@chula.ac.th

Abstract

We describe a system for the AKBC Shared Task 2026. The task is to predict complete sets of knowledge base objects from parametric memory alone, with no retrieval and no fine-tuning, across six relations and within a 32B parameter budget. Our system combines relation-specific multiple prompting, consistency-based aggregation, and error-driven prompt refinement. For each relation, multiple prompt variants and samples elicit predictions, with cross-prompt agreement reducing wording-specific errors and cross-sample agreement filtering unstable guesses. Aggregation is tailored to each relation, including cluster-median aggregation for numeric values and voting for categorical and set-valued outputs. The system achieves a validation macro-F1 of 0.5577 and a test macro-F1 of 0.5699, compared to organizer baselines of 0.3131 (validation) and 0.2964 (test).

1 Introduction

Large language models store a considerable amount of factual knowledge from pre-training. We ask how reliably that knowledge supports prediction without retrieval augmentation or task-specific fine-tuning. Closed-book knowledge base construction (KBC) tests this. When given a (subject entity, relation) pair, the system must list all correct objects from parametric memory, with no external database. AKBC Shared Task 2026 (AKBC 2026 Shared Task Organizing Committee, 2026) turns this into a benchmark. It requires complete and precise predictions across six relations within a 32B total-parameter budget.

Our system is built on `qwen/qwen3.6-27b` (Qwen Team, 2026), with a `gemma-3-4b-it` (Gemma Team et al., 2025) auxiliary for language identification, and applies four techniques: (1) **relation-specific few-shot prompting**; (2) **self-consistency decoding** (Wang et al., 2023) with confidence-escalated sampling for numeric relations; (3) **cross-prompt majority voting**, where several prompt variants predict independently and a prediction is kept only when at least two agree; and (4) **error-driven prompt refinement**, where the model reflects on its own errors and proposes better prompts. The system achieves a validation macro-F1 of **0.5577**, an improvement over the 0.3131 baseline. Our submission appears on the Codabench leaderboard under the team name **DataMind**.[1]

2 Task and Setup

Relations. AKBC Shared Task 2026 (AKBC 2026 Shared Task Organizing Committee, 2026) covers six relations in three answer types. First, **numeric relations** return a single numeric value as a string, scored with 5% relative tolerance:

- `hasArea` asks for the total area of a geographic entity in square kilometers.
- `hasCapacity` asks for the maximum spectator capacity of a venue.

Next, a **single-object relation** returns exactly one string or an empty prediction:

- `personHasCityOfDeath` asks for the city where a person died; living persons or unknown cases return `[]`.

Finally, **set-valued relations** return zero or more objects:

- `countryLandBordersCountry` asks for all countries sharing a land border with a given country.
- `companyTradesAtStockExchange` asks for all stock exchanges where a company is independently listed.

* Corresponding author.

[1]Code and prompts: https://github.com/tanaree-pat/akbc-2026

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 550–559
October 28, 2026 ©2026 Association for Computational Linguistics

- `awardWonBy` asks for all recipients of a given award by name.

Evaluation. Performance is measured by F1 averaged over all test instances. For each (subject, relation) pair, F1 is computed against the gold set, then average across all pairs. Numeric predictions count as correct within a 5% relative tolerance. Gold answers are lists of canonical names and aliases, and the evaluator accepts any alias.

Constraints. The task sets three strict constraints. First, systems are **closed-book**, which means no retrieval augmentation, web search, or access to external knowledge bases at inference. Second, **no fine-tuning** is allowed. All methods run at inference time. Third, the **total parameter count** across all models must stay under 32B. Non-neural post-processing, such as normalization, deduplication, voting, and numeric clustering, is allowed, as is multi-step agentic inference.

Data and model. The dataset has training, validation, and test splits. We use the training split only as a source of few-shot examples. `qwen/qwen3.6-27b` (Qwen Team, 2026) (27B) is the main model. In addition, a `gemma-3-4b-it` (Gemma Team et al., 2025) model (4B) is used together with Qwen for selecting a prompt language when working on the `hasArea` relation. The total parameter count for both models is 31B, staying within the 32B limit.

Baseline. The organizers provide a reference system using a Qwen 3.5 9B model with simple prompting. It achieves an overall macro-F1 of 0.3131 on validation and 0.2964 on test. Per-relation baseline scores appear alongside our results in Table 2.

3 Methods

Table 1 lists the technique used for each relation. The subsections below describe each one, while the entire set of prompts can be found in Appendix A.

3.1 Numeric Relations

`hasArea`. Our prompt asks the model to return the total area of a geographic entity in square kilometers as a single numeric string, and to always give a number, never an empty answer. Few-shot examples cover countries, medium-sized islands, and small islands with decimal values, so the model is calibrated across the full output range.

To predict an area, we designed a nested ensemble technique where the model answers the same question many times and those answers are calculated into a single number. This happens in two stages. First, within a single run, the model is sampled repeatedly and combined into one area estimate. Second, six independent runs each produce an area estimate, and the 6 estimates are combined into the final area estimate.

For a single run, we use confidence-escalated self-consistency (Wang et al., 2023), where the model is sampled 5 times at temperature 0.7 and the samples are pooled by a *cluster-median*. To compute the cluster-median, we group the samples within 5% of one another and return the median of the cluster with the most members. A run's confidence is the size of its largest cluster divided by the number of samples, which measures how strongly the samples agree. The sampling escalates to nine draws if the run's confidence is below 80%.

For the six independent runs, the six numbers are pooled by the same cluster-median to give the final area estimate. Runs 1–5 use the *same* base prompt and differ only by random sampling at temperature 0.7. Run 6 uses an **anchored native-language** variant as we believe that the model may have seen area figures in local-language sources that are absent from English text, so querying in the entity's native language may return a more precise value. The prompt first names the entity in English (Step 1) to fix its identity, then asks for its area in the target language (Step 2). This stops the model from mapping the name to a different extent in the target language (for example, a sub-national territory read as the whole country when queried in Chinese). To pick the language, `qwen/qwen3.6-27b` first names the entity's primary local language together with a confidence in $[0, 1]$. If that confidence is below 0.7, `gemma-3-4b-it` is asked the same question, and its answer replaces qwen's when gemma is more confident. The prompt is issued in the chosen language only if this confidence reaches 0.7.

`hasCapacity`. The prompt asks the model to name the venue's country and venue type, then estimate its capacity using typical size based on the region and type given in the prompt. The prompt separates national stadiums, top professional leagues, and community grounds by typical size ranges. As with `hasArea`, we apply confidence-escalated self-consistency, drawing 5 samples and escalating to 9 when they disagree as the model may hold the

Relation	Technique summary
hasArea	Nested ensemble: per-run SC (5→9, cluster-median), then cluster-median over 6 runs (5 base + 1 native-lang)
hasCapacity	Single prompt with confidence-escalated SC (5→9)
personHasCityOfDeath	Ensemble using agreement from all 4 prompts, else abstain
countryLandBordersCountry	Single exhaustive prompt, single-sample
companyTradesAtStockExchange	Ensemble vote, keep if ≥3 of 5 prompts agree, else abstain
awardWonBy	Union of alphabetical + year-by-year sweeps, each with per-name confidence vote (SC×5, ≥2/5)

Table 1: A summary of the relation-specific techniques. Note that SC refers to self-consistency.

capacity value in parametric memory but report it with variance, and sampling lets the cluster-median recover the stable value.

3.2 Single-Object Relation

`personHasCityOfDeath`. Four prompt variants run independently, and a city is output only if all the four variants return the same city. If any variant disagrees or abstains, the system returns `[]`. Each prompt variant is as follows.

The **recent_aware** variant drafts then revises. The model first identifies who the person is and when they were most active, then considers whether someone prominent in the 1980s–2010s and likely in their 70s–90s may have died recently. This makes the model name a death city for older figures it would otherwise assume are still alive. It commits to a city if reasonably confident, and returns `[]` only when sure the person is alive or when it has no information. The **recent_precise** variant uses the same recency reasoning plus explicit city-name rules: US sub-city districts such as Brooklyn or West Hollywood map to their enclosing city (New York City, Los Angeles), and UK deaths use the specific named town rather than the nearest major city. The **city_precision** variant uses the same city-name rules without the recency framing. It decides whether the person is living or dead, applies the naming rules if dead, and returns `[]` when it has no reliable information rather than guessing. The **meta_antidefault** variant comes from error-driven prompt refinement, similar to (Wu et al., 2024). In particular, the model was shown its past errors with gold labels and asked to find failure patterns. The resulting prompt tells the model to check whether its recalled city is the actual place of death or just a biographical link such as the career city, birthplace, or national capital, and to output a city only when it has a direct, specific memory of the death there.

Using four differently-framed prompts is the key to this relation. Each variant reasons in its own way so each makes different thought process. Requiring all four to agree cancels out the errors any single framing would make on its own.

3.3 Set-Valued Relations

`countryLandBordersCountry`. A single prompt with a directional-exhaustiveness instruction ("check all directions: north, south, east, west; do not stop early") and an explicit land-border-only constraint is used. Island nations with no land borders return `[]`. Few-shot examples include a two-neighbor case (Sierra Leone), an island nation (New Zealand), and a nine-neighbor case (Germany). We use a single model sample without ensembling because our observation shows that the same errors tended to recur across prompt variants, suggesting that majority voting may offer limited benefit.

`companyTradesAtStockExchange`. Five prompt variants run independently, and an exchange is kept only when at least three agree. Each prompt variant is as follows.

The **simple** variant is a baseline few-shot prompt with listed and unlisted companies. The **listed_check** variant runs two steps: Step 1 decides whether the company is independently listed under its own name, rejecting subsidiaries, private firms, cooperatives, and delisted companies; Step 2 names the exchange only if Step 1 passes. The **meta_precision** and **meta_verify** variants come from the same error-driven refinement and target over-prediction. Both prohibit inferring a listing from a company's name, headquarters, or sector and instead require the model to recall an actual listing. The **anti_confusion** variant targets two error patterns: NYSE–Nasdaq confusion, which it addresses through ticker-length heuristics, and incorrect country-based defaults, which it addresses by requiring a recalled listing rather than one inferred from the company's headquarters.

awardWonBy. Two runs are combined for this relation. The **alphabetical** run tells the model to recall recipients by first letter (A through Z). Meanwhile, the **year-by-year** run enumerates recipients from the award's first year through 2026 (decade by decade for cohort-elected fellowships), which reaches deeper on long-history awards. Both prompts set a recall target: "aim for 50 or more names; a partial list of 100 names scores higher than a tidy list of 10."

Each of the two runs is filtered by a per-name confidence vote where the model is sampled five times, and a name is kept only if it appears in at least two of the five samples. Here the output is a list of names rather than one answer, so we vote on each name on its own. When average per-name confidence is low, sampling escalates to nine draws. The two filtered runs are then unioned.

For this relation, we observed that some responses exhausted their generation budget while still reasoning, and never produced the output array. Therefore, to recover an answer in this case, we use a second call that asks the model to extract every recipient name from its own reasoning text. If that call also fails, a regex pass look for quoted, bulleted, or numbered names from the reasoning text.

4 Results

Table 2 reports macro-F1 per relation on validation and test sets, with the organizer baseline.

Overall. The system achieves a test macro-F1 of 0.5699 and a validation macro-F1 of 0.5577, compared to baseline scores of 0.2964 on test and 0.3131 on validation—a test gain of 0.2735 points. Every relation improves over its organizer baseline on both splits, with test-set gains ranging from +0.1021 (`hasCapacity`) to +0.4941 (`companyTradesAtStockExchange`).

Performance range. Validation F1 runs from 0.1802 to 0.9892 across the six relations, and the test F1 presents in a similar range, from 0.2245 to 0.9440.

Per-relation characterization. `countryLandBordersCountry` is the easiest, sitting near-ceiling on both splits. Three relations show moderate performance. `companyTradesAtStockExchange` and `hasArea` both gain from prompt ensembling, and `personHasCityOfDeath` gains from a deliberate precision-recall tradeoff (below). The two hardest are `awardWonBy`, which is recall-bound because the model enumerates only part of the recipient set, and `hasCapacity`, where errors come from entity confusion on obscure venues.

Precision-recall tradeoff in `personHasCityOfDeath`. Unanimity voting outputs a city only when all four variants agree. On the test set, this achieves 0.8700 precision and 0.5500 recall. By abstaining when the variants disagree, the method reduces incorrect predictions but also lowers recall. This precision-recall tradeoff leads to higher F1.

Validation vs. test shifts. `companyTradesAtStockExchange` and `awardWonBy` improve a lot on test (0.6752→0.7668 and 0.1802→0.2749). Differences in split composition may contribute to these changes.

5 Analysis

From our experiments, we analyze why the remaining errors occur.

Entity prominence and identification errors. From our inspection, `hasArea` and `companyTradesAtStockExchange` errors suggests difficulties with some less prominent entities. `hasCapacity` adds a related pattern, where the model anchors on a more famous venue that shares a name or city with the target and returns a plausible value for the wrong venue.

Incomplete enumeration in `awardWonBy`. Some awards have hundreds of recipients. The model recalls some reliably but cannot list the full set. We observe that missed recipients are not less well-known than recalled ones. Hence, the limit is possibly how many names the model can produce.

`personHasCityOfDeath`. We inspected that errors include failure to produce a death location, associated location substitution, such as a career city or birthplace, and other incorrect city predictions. Some missed deaths occurred well before the model's reported training cutoff, indicating that recency alone cannot explain these failures.

6 Related Works

Language models store factual knowledge from pre-training text (Petroni et al., 2019), and that knowledge is uneven. Entities that appear more

Relation	Base (val)	Ours (val)	Base (test)	Ours (test)
hasArea	0.2900	0.5800	0.3400	0.5700
hasCapacity	0.1800	0.2200	0.1224	0.2245
personHasCityOfDeath	0.2100	0.5000	0.1700	0.4900
countryLandBordersCountry	0.6650	0.9892	0.7315	0.9440
companyTradesAtStockExchange	0.3543	0.6752	0.2727	0.7668
awardWonBy	0.1008	0.1802	0.1510	0.2749
Overall	0.3131	0.5577	0.2964	0.5699

Table 2: Macro-F1 per relation on validation and test splits, with organizer baseline (Base) and our system (Ours).

often are recalled more reliably (Mallen et al., 2023; Kandpal et al., 2023). This literature motivates entity prominence as a possible explanatory factor in our setting.

Prompt engineering and output aggregation matter for reliable factual recall. Jiang et al. (2020) show that prompt phrasing strongly affects what a model recalls. Self-consistency (Wang et al., 2023) improves accuracy by sampling several outputs and choosing by majority. We apply this technique to numeric clustering and per-name voting. Also, automatic prompt optimization (Li et al., 2025) improves the prompts based on erroneous cases. We apply it to build the meta_antidefault and meta_precision prompt variants.

Selective prediction, i.e., abstaining on uncertain inputs to raise precision, has been studied in QA (Kamath et al., 2020). We show the same idea works for KBC, where requiring unanimity across four prompt variants for `personHasCityOfDeath` raises test precision to 0.8700 at the cost of recall.

7 Conclusion

We presented a closed-book KBC system for the AKBC Shared Task 2026 that reaches a test macro-F1 of 0.5699, within a 32B budget and without retrieval or fine-tuning. Its main strength is agreement. By asking each question with several prompts and several samples and keeping only the answers that recur, the system draws out reliable knowledge and filters errors. We also tailor the techniques to each relation. The techniques improve performance over the organizer baseline, but errors remain in entity identification, enumeration, and factual prediction. These results reflect the limitations of our current configuration, which calls for future research to address them.

Acknowledgments

We would like to thank Piyawat Lertvittayakumjorn for his advice throughout this work.

References

AKBC 2026 Shared Task Organizing Committee. 2026. AKBC shared task 2026: Predicting complete knowledge base entries from language models. https://lm-kbc.github.io/challenge2026/. AKBC Workshop at EMNLP 2026, Budapest.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 192 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Amita Kamath, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5684–5696. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. 2025. A survey of automatic prompt engineering: An optimization perspective. *arXiv preprint arXiv:2502.11560*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating

effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Qwen Team. 2026. Qwen3.6-27B. https://huggingface.co/Qwen/Qwen3.6-27B. Official model card.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yurong Wu, Yan Gao, Bin Benjamin Zhu, Zineng Zhou, Xiaodi Sun, Sheng Yang, Jian-Guang Lou, Zhiming Ding, and Linjun Yang. 2024. StraGo: Harnessing strategic guidance for prompt optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10043–10061, Miami, Florida, USA. Association for Computational Linguistics.

A Prompts

This appendix gives the exact prompts used in the final system. `{subject}` is replaced with the subject entity at inference time. The full source, including all prompt variants, is in the code repository.

A.1 hasArea

Runs 1–5 use the base prompt. Run 6 uses the anchored native-language prompt.

`base`: direct area query, with few-shot examples spanning countries to small islands (runs 1–5).

```
What is the total area of {subject} in
    square kilometers? For countries use
     total area (land + inland water).
    Return a JSON array with a single
    number as a string, no units.
Examples:
- Israel: ["20770"]
- Mangareva: ["15.4"]
- Wellington Island: ["5556"]
Always give your best numeric estimate
    -- never return an empty answer. No
    explanation.
```

`native_lang_anchored`: anchors the entity in English, then asks for its area in the local language (run 6).

```
What is the total area of {subject} in
    square kilometers?
Step 1: Identify what {subject} is --
    its type (island, lake, country,
    province, etc.) and location. If it
    is an island or sub-national
    territory, confirm it refers to that
     specific geographic feature ONLY,
    not any larger administrative region
     or country it belongs to.
Step 2: Identify the primary local
    language of where {subject} is
    located.
Step 3: Recall what local-language
    sources say about this specific
    entity's area -- they often have
    more precise figures for lesser-
    known places than English sources.
Step 4: Apply the correct rule:
    countries → land + inland water;
    islands → the island's land area
    only; lakes → water surface area.
Always give your best numeric estimate
    -- never return [].
Return a JSON array with a single number
     as a string on the last line. No
    units.
```

Language identification (run 6, Step 1): asked of `qwen/qwen3.6-27b`, and of `google/gemma-3-4b-it` as a fallback when qwen's confidence is below 0.70. The higher-confidence answer is used.

```
What is the primary local language of
    the place, country, or origin
    associated with "{subject}"? Respond
     ONLY as JSON: {"language": "<
    language name>", "confidence": <0 to
     1>}
```

Translation (run 6, Step 2): when the language confidence is at least 0.70, `qwen/qwen3.6-27b` translates the filled-in anchored prompt (`{prompt}`) into the identified language (`{language}`), keeping the entity name in English so its identity stays anchored.

```
Translate the following text into {
    language}. Keep every number, JSON
    array, and the entity name EXACTLY
    as written. Output ONLY the
    translation, with no preamble or
    commentary:

{prompt}
```

A.2 hasCapacity

`country_tier`: identify the venue's country and type, then estimate from region-specific capacity norms.

```
What is the maximum spectator capacity
    of {subject}?

Step 1: What country and city is {
    subject} in, and what type and level
     of venue is it?
Step 2: Estimate capacity using regional
     norms for that venue type:
  - South America (Brazil, Argentina,
    Colombia): often large, 30,000-
    100,000
  - Western Europe top division: 20,000-
    80,000; lower divisions: 5,000-
    20,000
  - North America (NFL/MLB/NBA): 17,000-
    82,500 depending on sport
  - East Asia (Japan, South Korea, China
    ): 15,000-70,000
  - Africa, South/Southeast Asia,
    smaller nations: 5,000-40,000
Step 3: Give your best specific number
    within that range.
Always give your best estimate -- never
    return [].
Return a JSON array with a single
    integer as a string on the last line.
```

A.3 personHasCityOfDeath

We use four prompt variants, and a city is output only if all four agree.

`recent_aware`: reasons about whether a long-active person may have died recently before committing to a city.

```
In which city did {subject} die?

Step 1: Who is {subject}? What are they
    known for, and when were they most
    active?
Step 2: As of 2026, consider deaths
    through 2025. If {subject} was
    prominent in the 1980s-2010s and was
     likely in their 70s-90s, they may
    have passed away recently -- commit
    to a city if you have reasonable
    confidence.
Step 3: If deceased, name the city.
    Return [] only if you are confident
    the person is still alive, or you
    have no information at all about
    their death.

City granularity only -- not country or
    region.
Examples:
- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []
Return only the JSON array on the last
    line.
```

`meta_antidefault`: separates a real death-place memory from a biographical default (birthplace, career city, capital).

```
In which city did {subject} die?

Step 1: Identify {subject} and their
    vital status. If alive or status
    unknown, go to Step 3.
Step 2: Recall the exact city of death.
    Separate confirmed fact from
    inference: is the recalled city the
    ACTUAL place of death, or merely
    where they were born, worked, or are
     culturally associated? Geographic
    defaults (national capitals, major
    hubs, career bases) are not
    substitutes for the real death
    location.
Step 3: Output the city ONLY if you have
     a direct, specific memory of where
    they died. If you know they died but
     not where, or you are inferring
    from association, return [].
City granularity only.
Examples:
- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []
Return only the JSON array on the last
    line.
```

`city_precision`: applies obituary-style city-naming rules (e.g. US boroughs map to the enclosing city).

```
In which city did {subject} die?

Step 1: Is this person living or
    deceased?
Step 2: If deceased, which city? Use the
     specific city name as it appears in
     obituaries.
Precision rules:
  - US: Brooklyn, Queens, Manhattan,
    Bronx, Staten Island → 'New York
    City'
  - US: West Hollywood, Beverly Hills,
    Santa Monica → 'Los Angeles'
  - US: Cambridge MA is 'Cambridge', not
     'Boston'
  - UK: use the specific town or city,
    not the nearest major city
  - Elsewhere: use the municipality
    where the death occurred
Step 3: Return [] if still living, or if
     you have no reliable information.

Examples:
- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []
Return only the JSON array on the last
    line.
```

`recent_precise`: combines the recency reasoning with the city-naming rules.

```
In which city did {subject} die?

Step 1: Who is {subject}? What are they
    known for, and when were they most
    active?
```

```
Step 2: Consider deaths through 2025. If
    {subject} was prominent in the 1980
    s-2010s and was likely in their 70s-
    90s, they may have passed away
    recently -- commit to a city if you
    have reasonable confidence.
Step 3: If deceased, name the city using
    obituary-standard precision:
  - US: Brooklyn, Queens, Manhattan → '
    New York City'; West Hollywood,
    Beverly Hills, Santa Monica → 'Los
    Angeles'
  - UK: use the specific town (not
    nearest major city)
  - Elsewhere: use the municipality
Return [] only if you are confident the
    person is still alive, or you have
    genuinely no information. Do not
    return [] just because you are
    uncertain -- a best guess is better
    than nothing.

Examples:
- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []
Return only the JSON array on the last
    line.
```

A.4 countryLandBordersCountry

`base`: one exhaustive directional sweep, land borders only, single-shot.

```
List every single country that shares a
    land border with {subject}. Be
    exhaustive -- do not stop early.
    Check all directions: north, south,
    east, west. Only land borders count,
    not maritime. Island countries with
    no land borders return [].
Examples:
- Sierra Leone: ["Guinea", "Liberia"]
- New Zealand: []
- Germany: ["Austria", "Belgium", "Czech
    Republic", "Denmark", "France", "
    Luxembourg", "Netherlands", "Poland
    ", "Switzerland"]
Return only a JSON array of country
    names. No explanation.
```

A.5 companyTradesAtStockExchange

Five variants; an exchange is kept only when at least three predict it.

`simple`: baseline few-shot query for the listing exchange(s).

```
At which stock exchange does {subject}
    trade? A company can be listed on
    multiple exchanges.
Examples:
- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo
    Stock Exchange", "Nagoya Stock
    Exchange", "Fukuoka Stock Exchange"]
- Bangladesh Cement Manufacturers
    Association: []
If not publicly traded or unknown,
    return []. Return only a JSON array
    of exchange names. No explanation.
```

`listed_check`: first checks the company is independently listed, then names the exchange.

```
Does {subject} have its own shares that
    trade on a stock exchange, and if so
    where?

Step 1 -- Eligibility. Decide whether {
    subject} ITSELF has publicly traded
    shares under its own name RIGHT NOW.
    Answer 'no' (→ []) if any of these
    apply:
  - It is a subsidiary whose shares are
    not separately listed (only its
    parent trades). e.g. a brand or unit
    owned by a larger listed group.
  - It is privately held (family/founder
    /private-equity owned, no public
    shares).
  - It is a cooperative, mutual,
    association, government body, or non
    -profit.
  - It was taken private, acquired, or
    delisted and no longer trades on its
    own.
Only answer 'yes' if the company files
    as an independent public issuer
    under its own name.
Step 2 -- Exchange. If and only if
    eligible, name the exchange(s) it is
    listed on (primary listing plus any
    cross-listings).

Be strict: if you are not confident the
    company is independently listed,
    return [].
Examples:
- Farfetch (independent public company):
    ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo
    Stock Exchange", "Nagoya Stock
    Exchange", "Fukuoka Stock Exchange"]
- Kintetsu Real Estate (subsidiary of
    listed Kintetsu Group): []
- Bangladesh Cement Manufacturers
    Association (industry body): []
Return only a JSON array of exchange
    names on the last line. If not
    independently listed, return [].
```

`meta_precision`: high-precision recall; forbids inferring a listing from name, HQ, or sector.

```
Return a JSON array of stock exchanges
    where {subject} is listed.
Strictly enforce precision: only include
    an exchange for which you have
    definitive knowledge of an actual
    listing. Do not infer a listing from
    the company's name, headquarters,
    or sector. Do not include private
    companies, or subsidiaries/acquired
    companies whose shares are not
    separately traded. If the company is
    not publicly traded or the listing
    is unknown, return [].
```

```
Examples:
- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo
    Stock Exchange", "Nagoya Stock
    Exchange", "Fukuoka Stock Exchange"]
- Bangladesh Cement Manufacturers
    Association: []
Return only a JSON array of exchange
    names. No explanation.
```

meta_verify: verifies public-trading status and actual recall before answering.

```
Identify the stock exchange(s) for {
    subject}.
Before outputting, verify two things:
    (1) the company is publicly traded (
    not private or an unlisted
    subsidiary), and (2) the exchange is
     a listing you actually recall --
    not a guess based on the company's
    name or region. If either check
    fails, return [].
Examples:
- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo
    Stock Exchange", "Nagoya Stock
    Exchange", "Fukuoka Stock Exchange"]
- Bangladesh Cement Manufacturers
    Association: []
Return only a JSON array of exchange
    names on the last line.
```

anti_confusion: targets NYSE–Nasdaq confusion and wrong-country defaults.

```
At which stock exchange does {subject}
    trade?

Two common mistakes to avoid:
1. NYSE vs Nasdaq confusion. These are
    DIFFERENT exchanges. Use the ticker
    symbol to identify which one -- NYSE
     tickers are usually 1-3 letters (e.
    g. DLB, EIX, IBM, AXP); Nasdaq
    tickers are usually 4+ letters (e.g.
     MSFT, AAPL, CSCO, AMZN). Do NOT
    guess -- if you are unsure which US
    exchange, return [].
2. Wrong-country default. A company may
    be listed in a country DIFFERENT
    from its headquarters. Do not return
     an exchange solely because the
    company is based there. Only name an
     exchange you actually recall the
    company trading on.

Rules:
- Only include exchanges where the
    company has its OWN shares listed
    independently (not a subsidiary or
    private company).
- Return [] if not publicly traded, or
    if you are not confident of the
    exchange.

Examples:
- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo
    Stock Exchange", "Nagoya Stock
    Exchange", "Fukuoka Stock Exchange"]
- Bangladesh Cement Manufacturers
    Association: []
Return only a JSON array of exchange
    names. No explanation.
```

A.6 awardWonBy

Two runs, alphabetical and year-by-year, each filtered by the per-name confidence vote, then unioned.

alphabetical: sweeps recipients by first letter to avoid year-by-year loops.

```
List every person or organisation that
    has received the {subject}.

Recall by NAME, not by year. Going year-
    by-year makes you loop and second-
    guess a single year forever. Instead
    , sweep your memory by first letter:
     think of recipients whose surname
    starts with A, then B, then C, and
    so on to Z. Write each name down as
    soon as it comes to mind and move on
     -- do not stop to verify the exact
    year.

Rules:
- Include ONLY the recipient's name --
    no years, no winning works, no
    categories.
- Aim for 50 or more names. A partial
    list of 100 names scores far higher
    than 10.
- Do not deliberate about any single
    name -- if you are fairly sure
    someone received it, include them
    and keep going.
- Only return [] if you are certain you
    know no recipients whatsoever.

Write the full JSON array of names as
    your FIRST output, before any
    reasoning.
Examples:
- Nobel Peace Prize: ["Martin Luther
    King Jr.", "Desmond Tutu", "Nelson
    Mandela", "Malala Yousafzai", "Elie
    Wiesel", "Mother Teresa", "Jimmy
    Carter", "Barack Obama", "Wangari
    Maathai", "Henry Kissinger", "
    Rigoberta Menchú", "Albert
    Schweitzer"]

Return a JSON array of names. No
    explanation.
```

year_sweep: enumerates recipients year by year for long-history awards.

```
List every recipient of the {subject},
    going year by year from when it was
    first awarded through 2026.

Format each year you are confident about
     as: YEAR: Name
```

```
Skip years you are genuinely uncertain
    about -- do not guess a name just to
     fill a gap.
For fellowships or memberships elected
    by cohort, go decade by decade
    instead.

Rules:
- Confirmed recipients only -- not
    nominees, shortlisted candidates, or
     people merely associated with the
    award.
- Names only -- no categories, no
    winning works, no extra commentary.
- Annual prizes with one winner: one
    name per year. Multiple-winner
    prizes: list all you know.

After the year-by-year section, compile
    every name into a single JSON array
    on the last line.
Return only the JSON array if you cannot
     recall any specific year-recipient
    pairs.

Example (Turing Award excerpt):
1966: Alan Perlis
1967: Maurice Wilkes
1968: Richard Hamming
["Alan Perlis", "Maurice Wilkes", "
    Richard Hamming", ...]
```

Relation-Specific Elicitation of a Base Model under a 32B Closed-Book Budget for LM-KBC 2026

Ruggero Marino Lazzaroni **Tim Booker** **Pau Soler Valadés**
Mathias Angermaier **Kirill Solovev**
University of Graz, Austria
{ruggero.lazzaroni,kirill.solovev}@uni-graz.at

Abstract

We describe a system for the LM-KBC 2026 shared task on closed-book knowledge base construction. Given a subject and relation, the system tries to predict all corresponding objects without retrieval or additional training and remains within the 32-billion-parameter limit. The system uses the base model gemma-4-31B with separate prompting and aggregation rules for six relations; few-shot demonstrations and selection thresholds come from the train set and are checked on the validation set. It obtains 0.6843 global F1 on the official validation set and 0.6961 on the official test set, compared with 0.308 for our run of the organisers' baseline on validation. The largest per-relation gains over our earlier baseline configuration come from a company-listing prompt with explicit abstention and a log-stratified area prompt, plus a vote-share abstention threshold for empty-heavy relations (like `personHasCityOfDeath`) that recovers most of the empty-case precision. Across more than 30 prompting, verification, and aggregation configurations, further inference rarely recovers facts that are absent from the generated candidates. These results indicate that relation-specific elicitation improves closed-book extraction, while candidate coverage and selection remain the main limitations. Code, prompts, and configuration files are available at https://github.com/ruggsea/lmkbc2026-submission.

1 Introduction

Research shows that graphs generation methods appear to still be unable to produce complete knowledge graphs, and pretrained language models provide one possible source of missing facts (Petroni et al., 2019; Kalo et al., 2023, 2025). The LM-KBC shared task evaluates this use directly: given a subject and relation, a system must return every corresponding object using only knowledge stored in the model's weights. Unlike single-fact probing benchmarks such as LAMA, this year's LM-KBC permits zero, one, or several objects. The 2026 task matches normalised strings through alias expansion and bipartite matching, then computes item-level macro-F1 with a ±5 % tolerance for numeric answers.

We set out to implement a system respecting the three constraints that the track imposes. First, inference is closed-book: the system cannot use retrieval, internet access, external factual corpora, or knowledge-base lookups, while non-neural filtering, normalisation, deduplication, and aggregation are permitted (section 3.3). We therefore use an external data source (Wikidata) only offline, and never at query or inference time, for two purposes: to construct the auxiliary pseudoval set described in appendix B, and to assemble the fixed demonstration bank used by the `hasArea` prompt (section 4.3). Assembling a demonstration bank offline is the scaled-up equivalent of hand-writing few-shot examples: the bank is fixed before any prediction is made, it is identical for every query, and nothing is looked up per subject. We further verify that it cannot supply an answer, excluding every subject that occurs in official train, validation, or test, and removing the query entity itself at prompt-construction time; both checks match on Wikidata entity identity rather than surface string, so aliases of the same entity are caught as well. Second, all neural components in the inference path must contain at most 32B parameters in total; multi-step and multi-model systems are permitted within this combined limit, while our system uses one model. Unlike the 2025 task, which prescribed Qwen3-8B, the 2026 task permits any open-weight model within the parameter budget (section 2). Third, the system cannot use fine-tuning, LoRA, probes, or soft prompts. The six relations include null-prone string sets (`countryLandBordersCountry`, `companyTradesAtStockExchange`, and `personHasCityOfDeath`), long recipient

Proceedings of the 11th Workshop on Automated Knowledge Base Construction (AKBC 2026), pages 560–571
October 28, 2026 ©2026 Association for Computational Linguistics

lists (awardWonBy), and numeric values (hasArea and hasCapacity).

We use one permitted base model as the sole parametric source and assign a separate elicitation method to each relation. The choice of base model is guided by the intuition that base models are most suited to few shot prompting and aggregation, plus they do not suffer from partial catastrophic forgetting and sycophancy that post training may introduce. Our one model design accommodates the different cardinalities, null rates, and output types without imposing one prompt and aggregation rule on all relations. The relation-specific elicitation methods are designed to address the different characteristics of the relations. Across the tested models, prompts, aggregators, and decoding strategies, facts absent from the sampled outputs were rarely recovered through further prompting. This pattern is consistent with the knowledge gaps reported by the 2025 systems and with the distinction between encoded and expressed knowledge in Inside-Out (Albert-Roulhac and Zouaq, 2025; He and Razniewski, 2025; Gekhman et al., 2025).

Our system obtains 0.6843 global F1 on the 475-item official validation set and 0.6961 on the official test set. Going deeper into null-item analysis, we examine the abstention rule for personHasCityOfDeath and show that its vote-share threshold determines every abstention decision, while a separate yes/no condition changes no prediction (section 4.4). We also compare more than 30 selection, verification, and generation configurations across the six relations. The comparison identifies candidate generation as the main limit for awardWonBy and candidate selection as a major limit for hasCapacity (section 5.3). Code, prompts, and configuration files are available at https://github.com/ruggsea/lmkbc2026-submission.

2 Related Work

LM-KBC. The LM-KBC series has moved from disambiguated single-object probing towards open-cardinality construction (Kalo et al., 2023, 2025). The 2025 task prescribed Qwen3-8B, whereas the 2026 task permits model and method selection within the parameter budget. Of the five systems submitted in 2025, the winning ReWiSe combined synthetic and human chain-of-thought demonstrations with relation-specific self-consistency (Albert-Roulhac and Zouaq, 2025). The runner-up used a Self-RAG-style describe-then-extract pipeline and divided awardWonBy into smaller subqueries (He and Razniewski, 2025). He and Razniewski (2025) attribute the persistent awardWonBy failures to missing knowledge rather than retrieval difficulty, reporting a "complete knowledge void" for post-1990 recipients in a Max Planck Medal case study, while Albert-Roulhac and Zouaq (2025) leave open whether the required awardWonBy and hasCapacity facts are present in the model weights. The best 2025 F1 values were 0.32 for hasArea, 0.15 for hasCapacity, and 0.18 for awardWonBy, the three relations the organisers identify as remaining particularly challenging; because 2025 prescribed an 8B model, these values contextualise rather than compare with 2026 scores. For personHasCityOfDeath, the 2025 results separate the baseline from the best system almost entirely by precision (0.08 vs 0.85) at nearly identical recall (Kalo et al., 2025), consistent with false-positive cities on empty-gold subjects dominating the baseline errors. This asymmetry also motivates our abstention analysis in section 4.4. In the 2022 task, a BERT baseline with a 0.5 likelihood threshold scored 31.08 % test F1, while predicting an empty set for every subject already produced 18.00 % (Singhania et al., 2022). Thus, abstention can account for a substantial part of the attainable score on null-prone relations.

Parametric knowledge and elicitation. Self-consistency has been shown to stabilise answers that the model already produces, while relation-specific vote thresholds adapt aggregation to different cardinalities (Wang et al., 2023). The Inside-Out framework has found that models encode more factual knowledge than they express, with an average relative gap near 40 % (Gekhman et al., 2025). This result is consistent with our finding that additional samples rarely recover city-of-death answers that never enter the draw pool, which we discuss in section 4.4. Two studies using masked single-token BERT-large probes over multi-valued Wikidata relations provide related evidence on cardinality estimation and self-verification. Singhania et al. (2023) report 49.5 F1 for top-k thresholding, 35.1 for a separate count probe, and 8.0 for a yes/no verification probe. The count probe almost always predicts more than ten objects, while the verification probe accepts nearly every candidate. Among missed objects, 26.9 % are generated below the cutoff and 27.75 % are never generated, which

parallels the two sources of awardWonBy recall loss discussed in sections 4.6 and 5.3. In a separate analysis of 41 relations, Veseli et al. (2023) report near-zero recall at 90 % precision for deathPlace and sharesBorders from BERT-scale probes. Both relations are far more accessible to our 31B base model (table 1), although model scale and elicitation format make this comparison unfair.

3 Task and Data

3.1 Official task

Each item contains a (SubjectEntity, Relation) pair, and the system returns a possibly empty list of ObjectEntities. The evaluator normalises strings, expands aliases, and applies bipartite matching; numeric predictions are correct when they fall within ±5 % of gold. We run the organisers' released Qwen3.5-9B baseline[1] ourselves and obtain 0.308 macro-F1 on validation, against the 0.313 reported in the task repository; the small difference reflects sampling and a later validation revision. We report our own run throughout so that every column of table 1 is scored on the same labels. Validation contains 68 border items, 100 company items, 100 cityDeath items, 100 hasArea items, 97 hasCapacity items, and 10 award items. Company and cityDeath contain 38 and 39 empty-gold items, respectively, while the award gold lists are long and sparse.

3.2 Evaluation surfaces and split discipline

We use official train to select methods and thresholds, official validation for reporting and acceptance checks, and a pseudoval set for early threshold and transfer checks. We constructed the pseudoval set by scraping approximately 2400 Wikidata rows with empty rates and entity types matched to the official distribution (appendix B). All numeric thresholds in the submitted system are selected on official train. However, some development decisions retained a component only when it did not reduce validation performance, as stated in the Limitations section. Therefore, our research approach was to go through different combinations of prompts, thresholds, and aggregation rules over train, to then probe on pseudoval and on the validation set in order to select the best configuration possible at that time of the development. When a change improved validation but reduced pseudoval performance, we treated the result as evidence of overfitting (e.g. section 4.1).

[1] https://github.com/lm-kbc/dataset2026

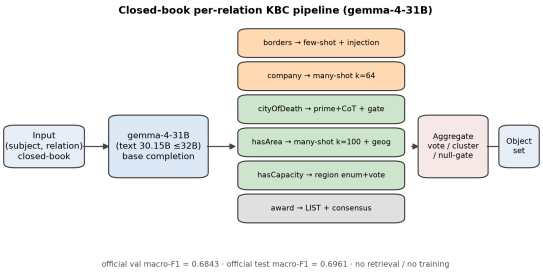


Figure 1: System overview with one permitted base model, six relation-specific prompt prefixes, stochastic sampling, and non-neural aggregation by voting, numeric clustering, or abstention.

3.3 Aggregation primitives

For string sets, our overall approach was to sample n completions ("draws"), parse and normalise their candidates, and keep an object (e.g. a country's name in the case of countryLandBordersCountry) when it appears in at least a fraction τ of draws. Empty and abstaining draws count against the object under this agreement threshold. For numeric relations, we parse the generated numbers and group values within a relative width. For areas, clustering operates in $\log_{10}$ space with width w=0.05, and the system returns the median of the largest cluster.

4 Method

After sweeping through different model, prompts combinations using some baseline prompts and thresholds, we opted to serve gemma-4-31B (Gemma Team, Google DeepMind, 2026) as a base completer that continues a plain-text prefix without chat roles. Direct counting of the safetensors gives 31.273B parameters: 30.697B in the language model plus 0.570B in the vision tower and 0.006B in the vision embedding (that we never invoke at inference), giving a final effective parameter count of 31.273B. The complete multimodal checkpoint therefore remains within the 32B limit, as the system never necessitates the multimodal capabilities of the model. For each item, the model continues a relation-specific prefix through n stochastic draws at a fixed temperature. A set of per-relation non-neural rules then parses the candidates and aggregates them into the final object set.

Prompting, aggregation, and thresholding. The final prompts and aggregation or thresholding methods used in the submitted system were selected through independent rounds of iteration for each relation. Selection and tuning for prompting, aggregation, and thresholding were performed independently per relation, according to the selection criteria discussed above. The full set of prompts and selected methods used in the final system are provided in Appendix A.

In addition, the relation-specific methods address distinct output structures. The capacity prompt enumerates venues with the same name before selecting the venue in the stated place, while the award prompt continues a year-by-year recipient list. Figure 1 summarises the system, while table 1 reports validation and test F1: in the table we compare the scores of the submitted system with a previous baseline we instantiated using the gemma-2-27B model.

4.1 `countryLandBordersCountry`

A short few-shot prefix of three hand-picked demonstrations shows semicolon-separated land neighbours and uses `none` for islands (appendix A.1). We sample 30 draws per subject at temperature 0.7. After tuning on the train set, we opt to keep a neighbour when it appears in at least τ=0.35 of draws and merge surface variants with the official alias table. Because land borders are well represented in pretraining, the main elicitation problem is over-inclusion. An island gating approach, larger demonstration banks, agreement across prompt variants, and reverse-border filtering did not improve train or/nor pseudoval performance over this recipe. Only one additional filter was found to be potentially beneficial: reverse-border filtering drops A→B unless the system also predicts B→A; it improved validation by 0.008 but reduced pseudoval performance by 0.014, therefore we did not include it in the submitted system.

4.2 `companyTradesAtStockExchange`

The many-shot company prompt uses an "issuer listing review" prefix with 64 sentences about listed and explicitly unlisted issuers, followed by `{Company} is…` (appendix A.2). We sample 100 draws per subject at temperature 0.7. Abstention phrases produce an empty vote; otherwise, we extract the named exchanges and retain those with agreement τ=0.3. Because approximately one third of subjects in the datasets have empty gold sets, the prompt places "not listed" outcomes in the same format as positive listings. The 64 demonstrations also match the prose used in listing disclosures for listed firms. A separate yes/no listing gate, binary probes for individual exchanges, and cross-method agreement did not seem to improve this configuration.

4.3 `hasArea`

The area prompt elicits a value in a Wikipedia-style lead sentence rather than through a bare numeric fill-in. From a bank of 331 entities with known areas, collected offline from Wikidata and filtered to exclude every subject appearing in the official splits, we remove the query, order the remaining entities by $\log_{10}(\text{area})$, and select 100 evenly spaced demonstrations (appendix A.3). Each demonstration names the entity type and gives the area in a short lead sentence. We sample 100 draws per subject at temperature 0.8, parse the first area-like number, and return the median of the largest cluster with width w=0.05.

The submitted prediction combines the lead-sentence draws with an independently sampled second channel, which we call the geographic-entry pool. This channel imitates the "Geography of X" section of an encyclopedia entry rather than a lead sentence: a prefix of three hand-written entries, none of which occurs in any official split, each open with the heading `Geography of {Entity}`, describe the entity in one sentence, and close with an explicit `Total land area:` figure in km^2 (appendix A.9). The query continues the same pattern, and we parse and cluster its 50 draws, sampled at temperature 0.8, exactly as for the lead-sentence pool. The two channels therefore differ in the surface form that carries the number, an encyclopedia section heading against a lead sentence, while sharing the model, the parser, and the clustering rule. When the two clustered medians differ by more than w, the system returns the geographic-entry value. This override fires for 28 of 200 pooled train and validation rows (0.140) and for 15 of 100 validation rows, of which nine are decisive. Its win rate is not statistically distinguishable from chance. The geographic-entry value wins 55.6 % of decisive validation rows, with a bootstrap 95 % CI of [0.22, 0.89]. On an independent larger split, it wins 34.2 % of 38 decisive rows, with a CI of [0.18, 0.50]. The resulting validation predictions score 0.7800 F1 for `hasArea` in table 1.

Relation	Baseline Validation F1	gemma-2 reference Validation P/R	gemma-2 reference Validation F1	gemma-4 system Validation P/R	gemma-4 system Validation F1	Test F1
borders	0.662	0.984 / 0.986	0.982	0.995 / 1.000	0.9970	0.9753
company	0.366	0.810 / 0.863	0.7519	0.878 / 0.903	0.8386	0.8470
hasArea	0.290	0.740 / 0.670	0.6700	0.850 / 0.780	0.7800	0.8500
cityDeath	0.180	0.910 / 0.580	0.5633	0.870 / 0.600	0.5800	0.6000
hasCapacity	0.180	0.402 / 0.402	0.4021	0.361 / 0.361	0.3608	0.3265
award	0.087	0.295 / 0.297	0.244	0.450 / 0.212	0.2374	0.3609
All	0.308	0.747 / 0.674	0.6458	0.773 / 0.702	0.6843	0.6961

Table 1: Per-relation scores. Precision and recall are available for validation; the final column reports official test F1. The gemma-2 reference and gemma-4 system use the same official validation set.

Log-stratified lead prompting recovers magnitudes more reliably than bare numeric recitation. The strongest alternative continues a Wikipedia-infobox stub ending at `| area_km2 =` (appendix A.10). Alternative units, Fermi-style estimates, density vetoes, and pooling the two prompts did not change train or pseudoval scores.

4.4 `personHasCityOfDeath`

Three prompt pools contribute to the decision (appendices A.4 to A.6). The prime pool is a few-shot biographical fact file ending at `City of death:`, while the status pool generates `Status: Alive|Deceased` and `City of Death:` fields. The yes/no pool asks whether the person is deceased with a recorded city and yields the yes-vote fraction y. We sample 30 prime draws, 50 status draws, and 20 yes/no draws per subject, all at temperature 0.7. The system pools the prime and status draws and identifies the modal city. It emits this city only when $y \geq \theta$ and $v/N \geq \phi$, where v is the modal-city vote count and N is the number of pooled city draws. We set $\theta = 0.1$ and $\phi = 0.3625$. The value of ϕ equals $29/(30+50)$, which is the large-n limit of an absolute threshold of 29 votes with 30 prime draws and 50 status draws. Among validation rows that satisfy the vote-share condition, the minimum y is 0.325, which is well above $\theta = 0.1$. Consequently, no row that passes the vote-share condition fails the yes/no condition, and setting $\theta = 0$ changes none of the 100 predictions. Abstention therefore depends entirely on the vote-share threshold. On a fixed validation draw pool, removing this threshold reduces F1 from 0.5600 to 0.420, whereas removing the yes/no condition changes nothing.

Approximately 39 % of validation subjects have empty gold sets, and any predicted city for these subjects scores zero. The vote-share threshold therefore reduces false-positive cities without a separate verification step. The gemma-2 reference configuration (section 5.2) reaches 0.563 F1, compared with 0.5800 for our validation predictions. We interpret this small difference as evidence that the rule primarily improves empty-side precision, rather than as a large relation-level gain.

4.5 `hasCapacity`

Many subjects follow the form `"VenueName in Place"`, although several venues can share the same bare name. The capacity prompt uses few-shot examples to enumerate venues with that name, select the venue in the stated place, and emit its capacity (appendix A.7). We sample 30 draws per subject at temperature 0.8, parse the final capacity number, cluster values within a 2.5 % relative width, and return the median of the largest cluster. Two simpler prefixes perform worse on venue homonyms: a direct prompt omits enumeration, while a description prompt adds a `Description:` field before `Capacity:`. The submitted system uses the cluster median on both splits, and the validation figures we report throughout are for that configuration. Rank-one voting within the same draw pool scores 0.3918 F1 on the 97 validation items, but it is not the submitted rule and we do not report it in table 1.

Controlled probes indicate that the generated samples contain relevant capacities that the aggregation rule does not select. On official train, a shortlist oracle marks an item correct when its gold capacity occurs in one of the three densest clusters from a large description-prompt pool. This oracle scores 0.470 F1, compared with 0.206 ± 0.038 for a magnitude-matched permutation null, giving a chance-corrected $\Delta \approx 0.26$. Without gold labels, digit probes, scientific notation, logit-lens and function-vector read-outs, and grammar-constrained decoding all perform worse than the capacity prompt. Tighter clustering also does not

address most errors. Only 16.7 % of incorrect train rows (11/66) and 20.7 % of incorrect validation rows (12/58) fall within 5 % to 20 % of gold. The remaining errors exceed 20 %, with an approximately $2\times$ band forming the largest interpretable group (27.3 % on train and 24.1 % on validation) and no stable unit-scale pattern.

4.6 `awardWonBy`

The Wikipedia-style award prefix demonstrates year-by-year recipients for two well-known awards and continues the same schema for the query award (appendix A.8). We sample 20 draws per award at temperature 0.8 and retain names with draw support $\geq \tau = 0.46$. The recipe scores 0.2374 F1 on the ten validation awards. These awards are specialised and their gold lists are sparse, so a real winner omitted from gold becomes a false positive when predicted. Relative to the gemma-2 reference configuration (section 5.2), the recipient-list prompt increases precision but reduces recall, leaving F1 similar in table 1. Generate-then-verify cascades reduced precision without a corresponding recall gain.

5 Experiments

5.1 Setup

We run vLLM offline on four H100 or four L40S GPUs with tensor parallelism of four. Subject-bootstrap intervals for global F1 are approximately ±0.04. A complete regeneration with unchanged prompts and thresholds but new stochastic draws changes official-validation global F1 by an estimated standard deviation of 0.0072, obtained by combining per-relation regeneration estimates in quadrature. We therefore treat single-run global differences below 0.014 as unresolved unless they also occur on train or pseudoval. Unless stated otherwise, every prompt comparison in this section is run with the same number of draws and the same temperature as the corresponding submitted prompt in section 4, so that alternatives differ only in the prompt itself.

5.2 Main results

Table 1 reports official validation and test results for our run of the organisers' baseline, our submitted gemma-4 system, and a gemma-2 reference configuration: the strongest system from our earlier development, which pairs `gemma-2-27b-it` (Gemma Team, Google DeepMind, 2024) with its own optimized (via iteratives sweeps) per-relation prompts and thresholds. Both model configurations use the same official validation set. Relative to the gemma-2 reference, our system gains 0.087 F1 for `companyTradesAtStockExchange` and 0.110 for `hasArea`, while `hasCapacity` decreases from 0.4021 to 0.3608. For `awardWonBy`, the reference configuration scores 0.244 and our system scores 0.2374. Because validation contains only ten awards, we interpret this 0.007 difference as a tie within subject-level noise. These comparisons are reference points rather than controlled ablations, because the base model and the relation-specific recipes differ together.

5.3 Error analysis

The relations exhibit different failure modes: cityDeath trades filled-case recall for empty-case precision, company is well calibrated on empty cases, the capacity method often predicts the wrong magnitude, and awardWonBy returns short lists relative to some gold sets. Appendix C reports the corresponding counts, per-award results, and negative results.

6 Discussion and Conclusion

Our closed-book LM-KBC 2026 system uses one permitted `gemma-4-31B` model with relation-specific elicitation and scores 0.6843 global F1 on official validation, compared with the 0.308 baseline. The official test score is 0.6961 F1. The company prompt, area prompt, and cityDeath vote-share threshold improve relations when the required facts occur in the generated candidates. However, missing candidates continue to limit award-recipient recall, while capacity errors remain dominated by selecting among generated magnitudes. The cityDeath threshold remains in the submitted system because it removes false positives for empty-gold items, while its separate yes/no condition changes no predictions (section 4.4). The 2025 systems reported the same difficult relations under an 8B mandate, and these relations remain the hardest for our 32B-budget system; verification, multi-agent methods, date anchors, and representation read-outs provide no consistent improvement. Two controls limit overfitting in this analysis: pseudoval matches official empty rates and entity types (appendix B), and the split discipline in section 3.2 rejects several validation-only gains that reverse on pseudoval.

Limitations

The awardWonBy validation set contains only ten subjects, so its relation-level differences are imprecise. While building a representative set for tuning, we found that matching pseudoval empty rates reduces but does not remove distribution shift.

Regarding model selection, our initial sweeps did not cover every model below 32B. Retrieval, models above 32B, and training on the target distribution fall outside the task rules and could change the attainable performance.

The reported validation score is one stochastic realisation rather than an expectation given the non-deterministic aspect of some of our prompting approaches. A complete regeneration changes official-validation global F1 by an estimated standard deviation of 0.0072. This estimate comes from a separate stochastic regeneration and was not recomputed for every reported configuration. Furthermore, although threshold values were not fitted to official-validation gold, several development decisions retained a component only when it did not reduce validation performance. This process makes the validation result an optimistic estimate of held-out performance, and we do not have a sufficiently precise train-only floor to quantify the optimism. The official test score therefore provides the primary independent confirmation of the submitted system.

Declaration on Generative AI

During the preparation of this work, the authors used large language models for refining and polishing the draft and for help with the literature organisation. The authors reviewed and edited all content and take full responsibility for the publication. The authors' evaluation pipeline, rather than generative tools, produced the experimental results, prompts, and scores.

Acknowledgements

This work was supported by VALPOP (EU Horizon Europe grant 101177310). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Executive Agency (REA). Neither the European Union nor the granting authority can be held responsible for them.

References

Edouard Albert-Roulhac and Amal Zouaq. 2025. ReWiSe: Relation-wise self-consistency for LLM probing. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025)*, volume 4041 of *CEUR Workshop Proceedings*.

Zorik Gekhman, Eyal Ben-David, Hadas Orgad, Eran Ofek, Yonatan Belinkov, Idan Szpektor, Jonathan Herzig, and Roi Reichart. 2025. Inside-out: Hidden factual knowledge in LLMs. In *Conference on Language Modeling (COLM)*.

Gemma Team, Google DeepMind. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Gemma Team, Google DeepMind. 2026. Gemma 4 technical report. *Preprint*, arXiv:2607.02770.

Jingbo He and Simon Razniewski. 2025. Combining self-retrieval-augmented generation with divide-and-conquer for language model-based knowledge base construction. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025)*, volume 4041 of *CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Simon Razniewski, Bohui Zhang, and Tuan-Phong Nguyen. 2025. LM-KBC 2025: 4th challenge on knowledge base construction from pre-trained language models. In *Joint Proceedings of the 3rd Workshop on Knowledge Base Construction from Pre-Trained Language Models and the 4th Challenge on Language Models for Knowledge Base Construction (KBC-LM+LM-KBC 2025)*, volume 4041 of *CEUR Workshop Proceedings*.

Jan-Christoph Kalo, Sneha Singhania, Simon Razniewski, and Jeff Z. Pan. 2023. LM-KBC 2023: 2nd challenge on knowledge base construction from pre-trained language models. In *Joint Proceedings of the KBC-LM Workshop and the LM-KBC Challenge @ ISWC 2023*, volume 3577 of *CEUR Workshop Proceedings*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Sneha Singhania, Tuan-Phong Nguyen, and Simon Razniewski. 2022. LM-KBC: Knowledge base construction from pre-trained language models. In *Knowledge Base Construction from Pre-trained Language*

Models 2022 (LM-KBC 2022), volume 3274 of *CEUR Workshop Proceedings*, pages 1–10.

Sneha Singhania, Simon Razniewski, and Gerhard Weikum. 2023. Extracting multi-valued relations from language models. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 139–154, Toronto, Canada. Association for Computational Linguistics.

Blerta Veseli, Sneha Singhania, Simon Razniewski, and Gerhard Weikum. 2023. Evaluating language models for knowledge base completion. In *The Semantic Web: 20th International Conference, ESWC 2023*, volume 13870 of *Lecture Notes in Computer Science*, pages 227–243. Springer.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

A Prompts for submitted methods

Unless stated otherwise, each prompt is a base-completion prefix that the model continues. The tokens `{name}`, `{s}`, `{query}`, and `{award}` mark the subject slot. Ellipses written as `[...]` indicate truncated demonstration lists, while the code release contains the complete banks.

A.1 Borders — set few-shot

```
Country: France
Land border countries: Belgium;
Luxembourg; Germany; Switzerland;
Italy; Monaco; Spain; Andorra

Country: Japan
Land border countries: none

Country: Bolivia
Land border countries: Brazil;
Paraguay; Argentina; Chile; Peru

Country: {name}
Land border countries:
```

Generation stops at a blank line or a new `Country:` header. The parser treats `none`, `n/a`, `no land borders`, and `island nation` as empty.

A.2 Company prompt with 64 demonstrations

The complete template contains 64 demonstration lines under this header and schema:

```
MSCI Global Investable Market Indices
-- issuer listing review.
For each issuer, the review records
the stock exchange or exchanges on
which its shares are publicly listed,
or notes that the issuer is not
publicly listed.

Shield On Service is publicly listed
on the Indonesia Stock Exchange.
JYP Entertainment is publicly listed
on the Korea Exchange (Kosdaq).
General Mills is publicly listed on
the New York Stock Exchange.
Orange County Choppers is not
publicly listed.
Vrtaky is not publicly listed.
[... 54 additional lines ...]
{s} is
```

If the first line contains *not publicly listed*, *not listed*, *not admitted*, *not traded*, *does not trade*, or bare *none*, matched without case sensitivity, the parser returns `[]`. Otherwise, it extracts the span after *(listed|registered|quoted|trades?|traded|trading) on (the)?*, splits the span on commas, semicolons, or *and*, and removes parts matching the abstention list. The final vote threshold is τ=0.3.

A.3 Area prompt with 100 demonstrations

Each demonstration comes from a 331-entity area bank that is stratified by log area at inference:

```
Monaco is a country. It has an area
of 2.02 km2.

Manhattan is a city. It has an area
of 59.1 km2.

Andorra is a country. It has an area
of 468 km2.

Belgium is a country. It has an area
of 30528 km2.

{query} is
```

The type phrase is *country*, *island*, *lake*, *city*, *mountain range*, or *geographic entity*, as appropriate. The parser extracts the first area-like number in km^2, clusters values at relative width w=0.05, and returns the median of the largest cluster.

A.4 City-of-death prime pool

Demonstrations are complete fact blocks ending at `City of death:`. The query prefix stops after `Nationality:`, and the model continues the remaining fields through `City of death:` in the same schema.

```
Person: Albert Einstein
Nationality: German-born, later
  Swiss and American
Field: theoretical physics
Years: 1879-1955
Final-years residence: Princeton,
  New Jersey
Buried: cremated, ashes scattered
  near Princeton
City of death: Princeton

Person: Keanu Reeves
Nationality: Canadian
Field: acting
```

```
Years: born 1964, still alive
Final-years residence: Los Angeles
Buried: N/A
City of death: N/A

Person: {name}
Nationality:
```

The parser reads the completed `City of death:` field and treats `N/A`, `none`, and `alive` as empty votes.

A.5 City-of-death status pool

```
Fact: Albert Einstein
Status: Deceased
City of Death: Princeton

Fact: Taylor Swift
Status: Alive
City of Death: N/A

Fact: {name}
Status:
```

The parser reads the completed `City of Death:` field and treats `N/A`, `none`, and `Alive` without a city as empty votes. Status-pool and prime-pool votes are combined before applying the emission rule in section 4.4.

A.6 City-of-death yes/no pool

```
Q: Is Albert Einstein deceased, and is
his specific city of death publicly
recorded as an established fact?
A: Yes

Q: Is Keanu Reeves deceased, and
is his specific city of death publicly
recorded as an established fact?
A: No

Q: Is {name} deceased, and is their
specific city of death publicly
recorded as an established fact?
A:
```

We compute $y = n_{\text{yes}}/(n_{\text{yes}} + n_{\text{no}})$.

A.7 Capacity prompt and alternatives

The submitted prompt uses three hand-written demonstrations in this schema:

```
Venue: Wembley Stadium in London
Venues named Wembley Stadium: Wembley
Stadium in London, England (national
stadium); Wembley Stadium in
Melbourne, Australia (minor arena).
In region: Wembley Stadium in London,
England.
Capacity: 90,000

[... two further demonstrations:
Olympic Stadium in Berlin (74,475)
and Goodwin Field in California
(3,500) ...]

Venue: {full_subject}
Venues named {bare_name}:
```

A weaker direct prompt uses a bare few-shot `Venue: ... / Capacity:` format without enumerating venues that share a name. A weaker description prompt inserts a one-line `Description:` field before `Capacity:`, and the query stops after `Description:` so that the model emits a description followed by a number.

A.8 Award-recipient list completion

```
List of recipients of the Pritzker
Architecture Prize:
1979 - Philip Johnson
1980 - Luis Barragan
[... additional years ...]

List of recipients of the {award}:
```

The parser reads `year -- name` lines and retains names with draw support $\geq \tau = 0.46$.

A.8.1 Per-award error detail

Two "honorary doctor of ⟨university⟩" awards score zero against gold lists of 40 and 53 names. The Presidential Medal of Freedom has the largest precision-recall difference, with precision 0.533 and recall 0.013 against a 610-name gold list. Its 24-name prediction cannot cover this gold set.

A.9 Area geographic-entry channel

The second `hasArea` channel described in section 4.3 continues an encyclopedia section rather than a lead sentence. The prefix holds three hand-written entries, none of which occurs in official train, validation, or test:

```
Geography of Tasmania
Tasmania is an island state of
Australia south of the mainland.
Total land area: 68,401 km2.
```

```
Geography of Sardinia
Sardinia is a large island in the
Mediterranean west of Italy.
Total land area: 24,090 km2.

Geography of Hokkaido
Hokkaido is the northernmost main
island of Japan.
Total land area: 83,424 km2.

Geography of {query}
{query} is
```

Draws stop at a blank line or the next heading. We parse the first area-like number and cluster the draws with the same $\log_{10}$ rule and width w=0.05 used for the lead-sentence pool, so the two channels differ only in the surface form that carries the number.

A.10 Area source-anchor alternative

The source-anchor prompt used for comparison in section 4.3 continues this infobox wikitext:

```
{{Infobox island
| name = Manhattan
| area_km2 = 59.1
| country = United States
}}

{{Infobox
| name = {query}
| area_km2 =
```

The prompt uses the same $\log_{10}$ clustering rule as the lead-sentence prompt, with w=0.05.

B Pseudoval composition

Pseudoval is an offline Wikidata-derived gold file of approximately 2400 rows constructed through SPARQL. We remove every `(SubjectEntity, Relation)` pair that also occurs in official train or validation. Table 2 reports its relation-specific construction.

Matching the official empty rates and entity types is necessary for useful transfer estimates. An unmatched construction with almost no empty company cases and too many protected areas produced a cityDeath threshold-ranking correlation of approximately $r = -0.02$ with official validation. With the matching rules in table 2, simple-baseline rankings correlate with official validation at $r > 0.6$ for all six relations.

Relation	N	Empty%	Rule
borders	84	∼13 %	land neighbours via P47∩country
company	293	∼44 %	P414 present vs. absent
cityDeath	250	∼46 %	humans w/wo place-of-death P20
hasArea	842	0 %	islands/lakes with P2046
hasCapacity	800	0 %	stadium/arena subclass, P1083
award	158	0 %	award subclass, winner list P1346

Table 2: Pseudoval composition by relation.

C Additional error and mechanism analyses

Negative results. None of the unsuccessful mechanism families exceeds the submitted relation-specific recipe on its development surface. Self-consistency saturates, verification and multi-agent configurations provide no consistent improvement, and the tested cross-relation gates fail to improve their baselines. Date anchors produce only variation within noise, two-model systems within 32B lose 0.06 to 0.10 global F1, and representation read-outs perform worse than the capacity and award methods. Following He and Razniewski (2025), these results distinguish facts that are generated but poorly assembled from facts that never enter the candidate pool. The company and area prompts improve the first case. Missing award recipients limit candidate coverage, while capacity errors also reflect selection among generated magnitudes. The paragraphs below summarise more than 30 tested strategies and the corresponding sample-count sweeps.

Error distributions. On official validation, cityDeath abstains correctly for 37 of 39 empty-gold subjects, producing two false positives, but it also abstains incorrectly for 29 of 61 filled-gold subjects. The company method abstains correctly for 32 of 38 empty-gold subjects and incorrectly for 3 of 62 non-empty subjects. For a single numeric prediction, precision, recall, and F1 are identical, so these errors reflect incorrect magnitudes rather than output format. For the ten awardWonBy items, the system never returns an empty list. The predicted-list length ranges from 7 to 70 names around a mean of 24.3. Performance varies by award; the Turing Award reaches precision 0.986 and recall 0.852 against 81 gold recipients. The number of gold recipients relative to the fixed shortlist length therefore constrains performance more than one

uniform precision-recall trade-off. Appendix A.8.1 reports the two items with zero F1 and the largest precision-recall difference.

Mechanism comparison. Set-relation self-consistency stabilises near $n{=}40$, while award recall remains nearly constant across broad sample-count sweeps. Judge, debate, panel, self-refinement, and generate-then-verify configurations provide no consistent improvement over the strongest relation-specific recipe. The separate company and border gates also fail to improve their baselines; we do not assume that this result transfers to the distinct cityDeath vote-share rule. Date anchors across 132 months produce only variation within noise. A larger model proposing candidates for a smaller filter model within the combined 32B budget more often removes correct predictions than recovers missing objects, with typical global losses of 0.06 to 0.10. Logit-lens, function-vector, and related hidden-state read-outs perform worse than the submitted methods for hasCapacity and awardWonBy. In total, we test more than 30 aggregation and selection strategies across all six relations by varying draw counts, thresholds, numeric-cluster widths, candidate filters, and cross-relation gates. For hasCapacity and awardWonBy, every tested selector performs below or within noise of the submitted recipe. No tested selection or aggregation change improves the reported development results under our acceptance criterion. The border island gate reproduces empty cases that the few-shot system already handles, while the company side question is not easier than predicting the listing itself. Taken together, these comparisons support the narrower conclusion that further prompting and selection do not reliably recover facts absent from the generated candidates.

Author Index

Abzianidze, Lasha, 316
Agadi, Akshaj, 139
Ahmed, Tanvir, 417
Alemany, Sheila, 343, 359
Angermaier, Mathias, 560

Bansal, Aditya, 56
Barcklow, Daniel, 393
Belu, Ianis, 543
Bonald, Thomas, 151
Booker, Tim, 560
Bossy, Robert, 247
Both, Andreas, 488
Boufouss, Younes, 234
Bruss, C. Bayan, 393
Bui, Thanh-Dan, 434

Cesari, Ettore, 316
Chang, Pei-Yun, 293
Chen, Chih-Hsi, 293
Chen, Xuan Ni, 293
Chen, Ze, 456
Chung, I-Ling, 293
Chung, Yeounoh, 94
Cochintele, Andreea-Elena, 527
Cutura, Jannic Alexander, 441

Daghero, Francesco, 19
de Jong, Jorryt, 316
Dhyani, Dushyanta, 169
Dieing, Thilo, 26
Dixit, Vibhu, 185
Do, Thanh-Trung, 434
Dobrovăț, Anca, 527

El Hamdani, Rajaa, 151

Fatsi, Noah, 393
Foussat, Anne-Sophie, 247
Friedrich, Annemarie, 45

Gao, Max, 456
Gastinger, Julia, 26
Gerald, Thomas, 234
Ghosh, Shrestha, 1, 8
Giordano, Luca, 1, 8
Guigue, Vincent, 247
Gupta, Vivek, 185

Habibillah, Fatih Naufal, 535
He, Jiarong, 456
Hsieh, Lawrence B., 427
Hu, Yujia, 1, 8
Huang, Hsin-Yu, 293
Hulsebos, Madelon, 94

Ji, Xingyu, 94

Kalaev, Dmitri, 393
Kalo, Jan-Christoph, 284, 409
Khalil, Christophe, 518
Kosiuk, Wojciech, 94
Kovalchuk, Sergey, 261
Kowalzik, Josua, 1
Kubryak, Oleg, 261
Kumar, Senthil, 393
Kunze, Jonas, 488

Lange, Lukas, 45
Lazzaroni, Ruggero Marino, 560
Lee, Darian, 56
Lee, Dokyun, 224
Lin, Chienyi, 293
Lu, Ke, 88

Malliaros, Fragkiskos, 151
Malmasi, Shervin, 169
Marripalapu, Neel, 139
Meilicke, Christian, 26
Melnyk, Igor, 393
Minot, Joshua R., 224
Moghaddam, Mahyar Tourchi, 19
Morača, Stefan, 316
Mourad, Aya, 518

Neacșu, Matei, 543
Nédellec, Claire, 247
Nguyen, Tuan-Phong, 1, 8, 409, 434
Nisioi, Sergiu, 527
Njor, Emil, 19

Özcan, Fatma, 94

Pappacoda, Gianmarco, 334
Paroubek, Patrick, 234
Patcharatrakul, Tanaree, 550

Pommeret, Luc, 234

Razniewski, Simon, 1, 8, 45, 409
Reinert, Owen, 393
Rosengren, William, 108
Rosset, Sophie, 234
Rumsey, Shannon, 56

Sasaki, Takashi, 473
Sauvion, Nicolas, 247
Schrader, Timo Pierre, 45
Shankarampeta, Abhilash, 185
Shi, Yuanming, 56
Silchenko, Maksim, 447
Simek, Olga, 343, 359
Sneiderman, Robert, 178
Soler Valadés, Pau, 560
Solovev, Kirill, 560
St. Clair, Jack, 56
Stuckenschmidt, Heiner, 26
Sullivan-Pao, Danielle, 343, 359
Sun, Yiqun, 427

Tang, Xinyi, 56
Thanga, Ashish, 185
Tjiaranata, Filbert Aurelian, 535
Torroni, Paolo, 334
Tym, Mikołaj, 462

Ugleva, Anastasia, 261
Ungureanu, Vlad-Mădălin, 543

Varde, Aparna S., 19
Vateekul, Peerapon, 550
Vedula, Nikhita, 169
Vo, Quoc-Linh, 480

Wang, Bryan, 169
Wang, Heng, 293
Wang, Zhe, 379
Wang, Zhigang, 379
Wei, Chengcheng, 456
Wei, Pengfei, 427

Yang, Edison, 139
Yang, Xiaolin, 379
Yi, John Seon Keun, 224
Yuen, Ian, 128

Zhang, Bohui, 409
Zhang, Wenjia, 379
Zheng, Chenjun, 379
Zheng, Jiewen, 456
Zheng, Wenqing, 393
Zimmermann, Adrian, 496