NORA 2026

NORA: The 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay

Proceedings of the Workshop

November 10, 2026

©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 979-8-89176-457-6

Preface by the General Chair

NORA 2026, the 3rd Workshop on KNOwledge GRaphs and Agentic Systems Interplay, held in conjunction with the 5th Asia-Pacific Chapter of the Association for Computational Linguistics (AACL) and the 15th International Joint Conference on Natural Language Processing (IJCNLP), continues to explore the intersection of AI Agentic Systems and Knowledge Graphs (KGs), building on the foundations laid in previous editions.

In recent years, autonomous AI agents have experienced an unprecedented surge in development, catalyzed by the rapid technological advancements of Large Language Models (LLMs). While modern agents benefit immensely from the linguistic proficiency and generation capabilities of these foundation models, they face persistent bottlenecks. Chief among these are the challenges of catastrophic forgetting during continuous deployment and the strict constraints of context window limits, both of which restrict an agent's ability to maintain a long-term, coherent understanding of contextual information.

Concurrently, KGs have solidified their role as a powerful paradigm for structuring, managing, and reasoning over interconnected data. KGs excel at providing the rigorous grounding, integration, and interlinking necessary for complex reasoning tasks. However, leveraging KGs typically demands significant upfront and ongoing investments in construction, curation, and domain expertise. The intersection of these two fields presents a natural and compelling synergy: LLMs offer fluid generation and adaptability, while KGs provide structured, persistent memory and deterministic factual grounding.

The NORA workshop is established to explore, analyze, and discuss this emerging frontier. Our program for the AACL-IJCNLP edition captures novel practices, ongoing research efforts, and deployed industry solutions that bridge the gap between connectionist AI agents and symbolic knowledge management. The call for papers attracted high-quality submissions from intradisciplinary, interdisciplinary, and multidisciplinary researchers, alongside industry practitioners and independent researchers.

The accepted contributions span a rich tapestry of topics, including personal memory and KG, architectures for persistent and shared agent memory, (graph) retrieval-augmented generation, semantic parsing, information extraction and the transformation of unstructured agent experiences into structured knowledge. Furthermore, these proceedings showcase grounded applications demonstrating how the synergy of agents and KGs transforms real-world domains such as enterprise systems, smart personal assistants, legal applications, agriculture, growth marketing and industrial engineering.

We welcomed four types of submissions through four tracks: Research papers, Position & Demo papers, Industry & Use Case presentations, and Short papers. We have also invited ARR commitment from previous ARR cycles. We received a total of 19 submissions, which were reviewed by 43 members of the Programme Committee. The review process was double-blind. Each submission received three reviews. In total, 11 papers were accepted for publication in this volume, including 8 Research papers, 1 Position & Demo paper, and 2 Industry & Use Case papers.

We would like to express our deepest gratitude to the program committee members for their rigorous and insightful reviews, ensuring the high academic quality of the workshop. We also thank the authors for their innovative contributions, our invited speakers for sharing their visionary perspectives, and the AACL-IJCNLP organizing committee for their steadfast logistical support. We hope these proceedings inspire continued collaboration and spark new research trajectories at the intersection of GenAI, NLP, Agentic Systems and Knowledge Engineering.

The NORA Workshop Chairs

Btissam Er-Rahmadi Sébastien Montella Damien Graux André Melo Hajira Jabeen

Organizing Committee

General Chair

Btissam Er-Rahmadi, Independent Researcher, UK

General Chair

Sébastien Montella, Huawei Technologies R&D UK Ltd, UK

General Chair

Damien Graux, EcoVadis, UK

General Chair

André Melo, Huawei Technologies R&D UK Ltd, UK

General Chair

Hajira Jabeen, University Hospital Cologne, Germany

Program Committee

Program Committee

Hanna Abi Akl, INRIA and Data ScienceTech Institute
Georgios Alexandridis, University of Athens
Ankit Arun, Facebook
Leslie Barrett, Bloomberg, LP
Debmalya Biswas, UBS Group AG
Davide Buscaldi, Ecole polytechnique and Université Paris 13
Won Ik Cho, Samsung Advanced Institute of Technology
Bonaventura Coppola, SAP Security Research
Deborah A. Dahl, LF AI & Data Foundation and Conversational Technologies
Sucheta Ghosh, Heidelberg University, Ruprecht-Karls-Universität Heidelberg and Heilbronn University of Applied Sciences
Tong Guo, Alibaba Group
Wei Hu, Nanjing University
John S Hudzina, Thomson Reuters
Srideepika Jayaraman, IBM TJ Watson Research Center
Jiyue Jiang, Carnegie Mellon University
Tracy Holloway King, Adobe Systems
Marek Kubis, Adam Mickiewicz University of Poznan
Sanjeev Kumar, Blazel Inc
Sebastián Ferrada, Universidad de Chile
Chong Li, Nanyang Technological University
Mingchen Li, University of Massachusetts at Amherst
Gilbert Lim, EyRIS
Aman Mehta, Snowflake
Hideya Mino, NHK
Lori Moon, MoonWorks, Inc.
Marcin Namysl, Ringler Informatik AG
Finn Årup Nielsen, Technical University of Denmark
Lucas Pavanelli, aiXplain
Vera Pavlova, burevestnik.ai and rttl labs
Pradyot Prakash, Meta
Shihao Ran, Dataminr
Alsu Sagirova, DeepPavlov
Munira Syed, The Procter & Gamble Company
Sudarshan R. Thitte, International Business Machines
Giuliano Tortoreto, University of Trento
Cassia Trojahn, Université Grenoble Alpes
Masaaki Tsuchida, Tokyo University of Science and MetaMoJi Corporation
Brian Ulicny, RTX BBN Technologies
Kaige Xie, Georgia Institute of Technology
Jinan Xu, Beijing Jiaotong University
Tianlin Zhang, University of Chinese Academy of Sciences, CHN Energy Digital Intelligence Technology Development (Beijing) Co. LTD and Independent
Dong Zhou, Guangdong University of Foreign Studies
Xiliang Zhu, Dialpad Inc.

Table of Contents

AMU: Admission and Memory Update for Personalized Conversations —Structured Memory with SLM Guided Control
Tao Hwang and Yishi Diao 1

Remember, Verify, or Ask? Evaluating Memory Decisions at Reuse Time
Baichuan Li, Junyi Yao and Zihao Zheng 15

Failure Modes of Imperfect Relational Context in LLM Document Extraction
Boris Kundu and Pooja Mehta 26

Asterism: A Hebbian-Weighted Knowledge Graph as a Priority Index over Long-Term LLM Memory
Bidit Das 39

Multi-View Contract Graph: A Knowledge Graph-Based Retrieval Framework for Contract Information Extraction
Momojit Biswas and Preethu Rose Anish 53

BD-AgriKG: A Knowledge Graph and Bilingual Text2Cypher Benchmark for Crop Suitability and Productivity in Bangladesh
Md. Aukerul Moin Shuvo and Md Nazrul Islam Mondal 63

Source-Aware Memory Retrieval across Engineering Standards
Sung Tae Yoo and Heyoung Yang 77

Beyond Chunking: Node Integrity and Field Structure in Bengali Retrieval-Augmented Agent Memory
Khan Raiyan Ibne Reza, Sanjana Aktar Maria and Sumaiya Tabassum Nimi 84

KG-SFT Contracts: Auditing Evidence Lineage and Loader Retention in Enterprise RAG Adaptation
Bogdan Roshchupkin, Alexander Litvintsev, Alina Avanesyan, Maria Khodorchenko and Andrei Zhukov 98

Tracing False Speaker Links Through Retrieval and Answering
Dev D. Goyal 109

Tracing Failures through Plant Topology: Propagation Reasoning by a Knowledge-Graph-Grounded LLM Agent
Muhamad Imannulhakim 131

AMU: Admission and Memory Update for Personalized Conversations — Structured Memory with SLM Guided Control

Tao Hwang[1,*], **Yishi Diao**[2]
[1]Independent Researcher, [2]Nanchang University
Correspondence: taohwang@ieee.org

Abstract

Large language models (LLMs) have become the foundation of personalized assistants, but maintaining persistent user memory across long-term interactions remains challenging. Existing memory systems often focus on storage, retrieval, or consolidation, while memory writing remains less controlled: transient requests, duplicate statements, and outdated user states may enter memory and later be retrieved for personalization. In this paper, we present **AMU**: **A**dmission and **M**emory **U**pdate for Personalized Conversations, an SLM-guided (Small language model guided) structured framework for writing-time memory control. AMU uses structured memory filtering to decide what should enter memory and SLM-guided storage management to determine whether an admitted record should be stored separately, discarded as a duplicate, or fused as an update. We evaluate AMU in a controlled memory writing and retrieval setting. Experimental results show that AMU maintains cleaner and more retrievable personalized memories. Our code is available at https://github.com/UnicusT11/AMU-memory.

1 Introduction

Large language models (LLMs) are now widely used in conversational agents and personalized assistants. Recent models have shown strong abilities in instruction following, reasoning, tool use, multilingual interaction, and multimodal understanding (Ouyang et al., 2022; OpenAI et al., 2024; Wang et al., 2024; Team et al., 2025; Grattafiori et al., 2024). However, many LLM-based interactions are still organized around a single session. The model can respond to the current prompt, but it does not naturally maintain a stable and evolving understanding of the user across long-horizon interactions (Zhong et al., 2023; Westhäußer et al., 2025). For personalized assistants, this makes persistent memory necessary, especially for reusable user information such as preferences, background, plans, and changing personal states.

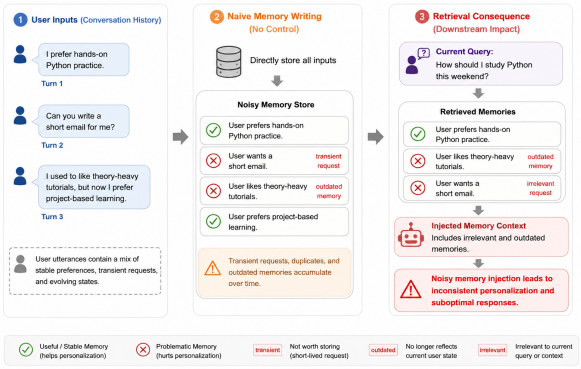


Figure 1: Motivation of writing-time memory control. Directly storing user utterances can preserve transient, redundant, or outdated memories, while AMU keeps stable and reusable memories for cleaner personalization.

Existing work has supported such continuity through several directions. Long-context methods place more dialogue history into the prompt or train models to handle longer contexts (Bai et al., 2024; Liu et al., 2024; Chen et al., 2024), but they can be costly and still do not ensure that the model uses the right evidence. Model-internal memory methods introduce memory tiers or read-write memory units inside the agent or model architecture (Packer et al., 2024; Modarressi et al., 2024), but these designs are often tied to particular implementations. External memory modules take a different route: they extract, store, retrieve, and inject user information outside the response model. This makes them practical for deployment, since the memory component can be updated without retraining and can be reused across different model backends (Zhou et al., 2026; Rafique and Bindschaedler, 2026). Prior systems have explored a range of external-memory designs

[*]Corresponding author.

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 1–14
November 10, 2026 ©2026 Association for Computational Linguistics

for storing, retrieving, and updating user information (Izacard and Grave, 2021; Park et al., 2023; Li et al., 2025; Xu et al., 2025; Chhikara et al., 2025).

Despite these developments, fine-grained control at the memory writing stage remains underexplored. User conversations contain mixed signals. Some inputs describe stable preferences or long-term facts, while others are temporary requests, repeated statements, outdated states, emotional expressions, or details that are only weakly relevant to future personalization. If such information is admitted into memory without sufficient control, the memory store can gradually accumulate noisy, redundant, or outdated entries. Figure 1 illustrates this problem: transient requests and outdated preferences may be stored together with useful memories and later retrieved, which can lead to noisy memory injection and inconsistent personalization. A memory system therefore needs to decide not only which stored memories should be retrieved, but also which information should be written into memory in the first place and how it should interact with existing records.

Retrieval filtering or post-hoc consolidation only partly addresses this problem. Retrieval can select stored memories for a particular query, but it cannot remove the effect of low-value records that have already been written into the memory store. Consolidation can merge or revise stored memories, but when admission is weak, it may still operate over a noisy candidate pool. A robust personalized memory system therefore needs two coupled writing-time capabilities. The first is admission control, which filters low-value or non-personal inputs before storage. The second is relation-aware memory update, which determines whether an admitted record should be stored as a new memory, discarded as a duplicate, or fused into an updated user state. These capabilities are particularly important in long-term personalized settings, where preferences, plans, and personal states may change over repeated interactions.

In this paper, we propose **AMU**: **A**dmission and **M**emory **U**pdate for Personalized Conversations, a structured framework for writing-time memory control in personalized conversations. AMU focuses on the memory writing stage and targets two problems caused by passive memory accumulation: low-value or non-personal information may enter long-term memory, and repeated or evolving user states may create redundant or outdated records. In AMU, structured admission first determines which user inputs are eligible for memory storage. A small language model (SLM) is then used as a lightweight controller to decide how each admitted record should interact with existing memories. Specifically, the SLM judges whether a candidate memory should be stored separately, discarded as a duplicate, or fused as an update to the current reusable user state. The resulting memories are kept as structured records and serialized into plain-text memory context for downstream RAG-based response generation.

Our contributions are summarized as follows:

- We introduce **AMU**, a structured framework for writing-time memory control in personalized conversations. AMU manages user memories before downstream retrieval by treating memory construction as a controlled writing process rather than passive accumulation.
- We propose structured admission for personalized memory writing. By parsing user inputs into typed memory records, AMU filters low-value or non-personal information before storage and uses structured fields to guide candidate selection and retrieval-time filtering.
- We propose SLM-guided memory update for maintaining reusable and evolving user memories. A lightweight SLM determines whether an admitted memory is duplicate, update, or separate from existing memories, and update-related records are fused into the current reusable user state.

2 Related Work

Large language models (LLMs) have enabled strong conversational and reasoning abilities, but persistent personalization remains difficult when user information must be maintained across long-term interactions. Existing personalized dialogue systems commonly condition responses on personas, user profiles, or dialogue histories to improve consistency and user adaptation (Shuster et al., 2022). Recent personalization benchmarks further evaluate whether LLMs can use user profiles, preferences, and historical interactions when generating responses (Salemi et al., 2024). However, these studies mainly focus on how user information is used during generation, while the process by which user utterances are admitted, filtered, updated, or discarded before retrieval is less explicitly studied.

Personalized Memory. Early personalized dialogue work usually relies on static persona descriptions or profile sentences. Personal Chat (Zhang et al., 2018) introduced persona-grounded conversations to evaluate whether models can produce responses consistent with assigned personas. Dinan et al. (2020) further studied controllable and persona-aware dialogue modeling in open-domain conversations. More recent work explores longer-term personalization and persistent memory management, where agents accumulate and maintain user-specific information from interaction histories rather than relying only on predefined persona statements. Mem0 maintains persistent memories through explicit memory operations, A-MAC focuses on adaptive memory admission and conflict-aware maintenance, and A-MEM organizes memories through structured linking and dynamic evolution (Chhikara et al., 2025; Zhang et al., 2026a; Xu et al., 2025). These studies show that memory construction and maintenance are increasingly important for long-term personalized interaction. AMU focuses specifically on writing-time control for personalized conversational memory, coupling structured admission with relation-aware decisions over whether an admitted memory should be treated as duplicate, update, or separate.

Retrieval-Augmented Memory. Retrieval-augmented language models use external knowledge or non-parametric memory to improve factuality and task performance. REALM retrieves external knowledge during pre-training and downstream prediction (Guu et al., 2020), and RETRO conditions generation on retrieved chunks from a large-scale retrieval database (Borgeaud et al., 2022). Retrieval augmentation has also been applied to black-box and in-context language models, showing that retrieved evidence can improve model outputs without changing model parameters (Ram et al., 2023; Shi et al., 2023). In dialogue and question answering, retrieval helps ground responses and reduce unsupported generation (Shuster et al., 2021; Komeili et al., 2022; Nakano et al., 2022). More recent methods study adaptive or self-reflective retrieval, deciding when retrieval is needed and how evidence should be incorporated (Asai et al., 2024; Jiang et al., 2023; Mallen et al., 2023). However, these works mainly address how to retrieve and use stored information at inference time. They do not fully address how a personalized memory store should be kept clean before retrieval. AMU complements retrieval-augmented methods by applying structured filtering during memory writing and retrieval.

Memory Construction and Management. Agent-oriented methods have shown that intermediate reasoning traces, feedback, and memory-like states can improve long-horizon behavior. ReAct combines reasoning with external actions in an agent loop (Yao et al., 2023), Reflexion uses verbal feedback as an episodic memory signal for future decisions (Shinn et al., 2023), and Self-Refine iteratively improves model outputs through self-generated feedback (Madaan et al., 2023). These methods demonstrate the usefulness of model-generated judgments and feedback. Nevertheless, personalized memory management requires a different form of decision-making: a system must decide whether a user utterance should be stored, discarded as redundant, or used to update an existing memory. AMU addresses this issue through structured memory filtering and SLM-guided storage management, where candidate memories are explicitly judged as duplicate, update, or separate before being written back to the memory store.

3 Admission and Memory Update for Personalized Conversations

This section describes AMU, an SLM-guided structured framework for personalized memory management. §3.1 presents the overall framework. §3.2 describes structured memory filtering, §3.3 introduces SLM-guided storage management, and §3.4 describes structured storage and RAG-based retrieval.

3.1 AMU Framework

AMU maintains personalized memories as structured and updateable records rather than an append-only log of user utterances. As shown in Figure 2, AMU consists of two coupled mechanisms: structured memory filtering and SLM-guided storage management. Given a user input, AMU first converts it into a typed memory record. The structured fields are then used to determine whether the input should enter memory processing, which existing memories should be considered as candidates, and which memory subset should be exposed during retrieval. For admitted records, the SLM judges whether the new record is duplicate, update, or separate with respect to existing memories. Sep-

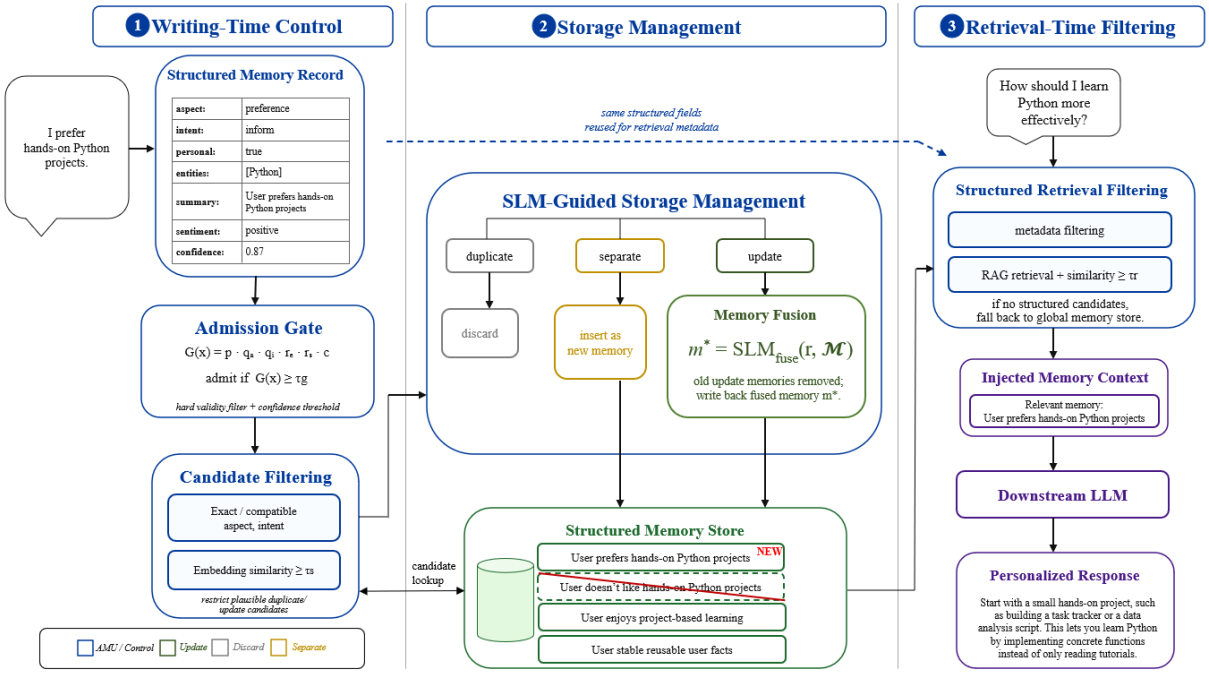


Figure 2: Overview of AMU. Structured fields guide memory admission and retrieval, while the SLM decides whether an admitted record is duplicate, update, or separate before storage.

arate records are inserted, duplicate records are discarded, and update-related records are fused into the current reusable memory state. By decoupling memory management from the downstream response LLM, AMU supports early exits and enables the maintained memory store to be reused across different LLM backends.

3.2 Structured Memory Filtering

Structured memory record. Structured memory filtering begins with a typed memory record. Given a user input, we use an SLM to produce

$$m = \{a, i, p, e, s, v, c\} \tag{1}$$

where a denotes the memory aspect, i denotes the communicative intent, p indicates reusable personal information, e contains salient entities, s is an affect-neutral memory summary, v is sentiment, and c is confidence. The parsing process first extracts structural fields, including aspect, intent, reusable-personal flag, and entities, and then formulates the memory summary, sentiment, and confidence score. This step constructs the representation used by subsequent filtering and storage management.

Each user input is structured with two complementary labels: aspect for the type of user information and intent for the communicative function of the utterance. Aspect labels correspond to common facets in user modeling and personal information understanding, such as profile, relationship, activity, health, and preference (Purificato et al., 2024; Tan et al., 2025), while intent labels correspond to dialogue-act-style abstraction of utterance function (Vasselli et al., 2025). Accordingly, we define the aspect set as {`personal_profile`, `social_relationship`, `activity`, `health`, `preference`, `other`}, and the intent set as {`inform`, `request`, `express`, `plan`, `reflect`, `other`}. These labels provide complementary structural views: aspect captures what the memory is about, while intent captures how the user presents it. The entity field anchors memory formulation but is not used as a hard matching key.

The memory summary s is stored as an affect-neutral statement that preserves one explicit, stable, and reusable user fact, while sentiment v is stored separately as {`positive`, `neutral`, `negative`}. This separation keeps factual memory content distinct from affective tone. We use the confidence score c as a lightweight reliability signal for the generated memory record, corresponding to calibration and self-evaluation in long-form generation (Huang et al., 2024). Since model-generated confidence is not necessarily calibrated, c is elicited with a rubric over explicitness, stability, and ambiguity, rather than treated as an unconstrained probability

estimate.

Admission gate. Based on the structured record, an admission gate determines whether the record enters the memory writing pipeline. We formulate memory admission as a writing-time control decision based on content type and persistence signals in memory admission control (Zhang et al., 2026a). Given $m = \{a, i, p, e, s, v, c\}$, the gate computes

$$G(x) = p \cdot q_a \cdot q_i \cdot r_e \cdot r_s \cdot c \tag{2}$$

Here, p indicates reusable personal information, q_a filters fallback aspects, q_i filters fallback intents and purely task-oriented requests without reusable personal content, and r_e and r_s indicate whether entities and memory summary are non-empty. Since all terms except c are binary, the gate implements a hard validity filter combined with a confidence threshold. A record is admitted only when $G(x) \geq \tau_g$.

Candidate filtering. For admitted records, structured labels restrict the candidate memory pool. Existing memories with the same aspect-intent pair are first retrieved. Because these labels are predicted fields, the lookup expands to compatible intents under the same aspect when no exact-pair candidates are found. The resulting candidate set is further filtered by embedding similarity:

$$\text{sim}_j = \text{sim}(u, u_j), \quad \text{sim}_j \geq \tau_s \tag{3}$$

where u and u_j are embeddings of the current record and candidate memory m_j. The threshold τ_s is used only during writing for duplicate/update candidate filtering. This step limits later SLM judgment to plausible duplicate or update candidates.

3.3 SLM-Guided Storage Management

After structured filtering, we perform SLM-guided storage management over the retained candidates. For each candidate, a compact comparison view is constructed using entities, memory summary, and sentiment. The SLM predicts the storage relation between the current record and each candidate as one of three labels: `duplicate`, `update`, or `separate`.

A `duplicate` record adds no reusable information beyond an existing memory and is discarded. An `update` record revises, narrows, contradicts, or supersedes an existing memory and should replace the old state. A `separate` record describes another stable user fact and can coexist with existing memories. This relational judgment turns memory writing from a binary store-or-drop decision into a storage management process that supports insertion, deletion, and update.

Deterministic actions are applied according to the predicted relations. If all candidates are separate, the current record is inserted as a new memory. If no update is found but at least one duplicate exists, the current record is discarded as redundant. If one or more update candidates are found, memory fusion is performed. We treat update as a storage-management relation for evolving user states, where outdated preferences or states should be revised rather than kept active (Sun et al., 2025). Let d denote the current structured record and $\mathcal{M}$ denote the list of candidate memories predicted as `update`. The SLM fuses d and $\mathcal{M}$ into a new memory record:

$$m^* = \text{SLM}_{\text{fuse}}(d, \mathcal{M}).$$

Only update candidates are used for fusion; separate candidates remain unchanged, and duplicates are excluded. The fused memory m^* represents the current reusable state. Old update memories are removed, and only m^* is written back to the memory store.

3.4 Structured Storage and RAG Retrieval

Accepted memories are stored as document-style structured records, with `memory_summary` used as the main retrieval text for vector indexing. Structured fields are also used as retrieval metadata to filter candidate memories before vector retrieval, so retrieval is performed over a more relevant subset of the memory store. If no structured candidates are available, retrieval falls back to the global memory store.

During retrieval, the RAG backend uses the current user query to retrieve relevant memories and injects them as plain-text memory context. We further filter the retriever's top-k results with a retrieval-time threshold τ_r, which suppresses weakly related memories during inference. This threshold differs from τ_s, which is used during writing for duplicate/update detection. Thus, structured filtering is used at both writing time and retrieval time: writing-time filtering controls which records enter and update the memory store, while retrieval-time filtering controls which maintained memories are exposed for downstream use.

4 Experiment

4.1 Dataset and Experiment Setup

Dataset. We use GPT-4o to construct a controlled benchmark for memory writing and retrieval by adapting LoCoMo-style long-term conversational contexts into single-topic sessions with explicit action labels and retrieval targets (Maharana et al., 2024; Kim et al., 2025). This setup is motivated by the privacy sensitivity and annotation difficulty of real long-term personal memory logs. The benchmark simulates one to two months of theme-based interactions, including stable preferences, evolving attitudes, repeated information, and preference or plan updates. Each session is generated under a specific theme with a guideline-constrained prompt and a fixed JSON schema. We generate 20 sessions with different themes, and each session contains 50 memory-writing turns followed by 10 evaluation queries. The benchmark contains 1,000 memory-writing turns and 200 evaluation queries in total. Each memory-writing turn is annotated with a construction-time action label, including `store`, `discard`, `update`, or `filter`, and each memory-bearing turn is assigned a slot-level `memory_id`. Each evaluation query is associated with expected memory ids for retrieval evaluation.

Experimental setting. Experiments are conducted on the 20 sessions. For each session, the first 50 turns are processed sequentially as memory-writing inputs, and the last 10 queries evaluate retrieval against the expected slot-level memory ids. All memory-based systems use the same Haystack-based RAG backend and Qwen3-Embedding-0.6B embedding model. The compared systems differ in how memories are constructed, admitted, updated, and maintained, while the downstream retrieval protocol is kept fixed for controlled evaluation. Qwen3.5-2B is used as the default SLM for memory management unless otherwise specified. All experiments were conducted on a single NVIDIA RTX 4070 GPU.

4.2 Manual Validation of Silver Labels

To assess the reliability of GPT-4o-generated silver annotations, the same subset of writing turns and evaluation queries was independently reviewed by multiple annotators. The subset was selected through programmatic random sampling of item IDs and contained 30 writing turns and 5 queries from each session, corresponding to 600 of 1,000 writing turns (60%) and 100 of 200 evaluation queries (50%). For writing turns, the annotators judged whether the assigned action matched the definitions of `store`, `discard`, `update`, and `filter`. For evaluation queries, they verified whether the expected `memory_ids` were traceable to the corresponding writing turns and appropriate for the query.

Table 1 reports the mean validity rates across the independent annotations: 89.3% for action-label validity and 94.1% for query-memory alignment. These values measure manual annotation validity rather than inter-annotator agreement.

Metric	Validity (%)
Action label validity	89.3
Query-memory alignment	94.1

Table 1: Manual validation of silver annotations. Reported values are mean validity rates across independent annotations.

4.3 Experimental Results

Compared baselines. We compare AMU with five baselines covering both simple memory-retention strategies and existing long-term memory-management methods. Full Compressed Memory compresses and stores all memory-writing inputs, representing summary- or compression-based memory construction in long-term memory systems (Chen et al., 2025; Zhang et al., 2026b). Sliding Window retains only recent memories and discards older ones, providing a recency-based reference under limited memory capacity. Mem0 (Chhikara et al., 2025) extracts salient information from conversational history and maintains persistent memories through explicit memory operations, including addition, update, deletion, and no-op. A-MAC (Zhang et al., 2026a) performs adaptive memory admission using signals such as utility, confidence, novelty, recency, and content type, together with conflict-aware memory maintenance, providing a direct comparison for selective memory writing and maintenance. A-MEM (Xu et al., 2025) constructs structured memory notes and dynamically links and evolves stored memories as new information is incorporated. All methods are evaluated under the same slot-level retrieval protocol to compare the quality of the maintained memory stores.

Method	P@3	R@3	F1@3	HIT@3	Red.@3 ↓
Full Comp.	35.42	55.50	41.30	57.00	12.08
Sliding Window	24.42	29.50	25.87	29.50	11.75
Mem0	34.08	57.50	40.88	56.00	3.63
A-MAC	30.36	58.76	37.96	54.50	3.18
A-MEM	35.13	56.90	40.36	56.50	2.88
AMU(Ours)	**37.33**	**63.50**	**44.47**	**57.50**	**2.83**

Table 2: Main retrieval results. Red.@3 denotes Redundancy@3, where lower values are better.

Evaluation metrics. We evaluate top-3 retrieval on memory-required queries. Retrieved memories are mapped to slot-level `memory_id`s and compared with expected memory ids. We report P@3, R@3, F1@3, and HIT@3, measuring precision, recall, precision-recall balance, and whether at least one expected memory slot is retrieved. Since AMU targets cleaner memory maintenance rather than retrieval coverage alone, we also report Redundancy@3, defined as the percentage of top-3 retrieved memories that repeat an already retrieved slot-level `memory_id`. Lower values indicate cleaner retrieval. No-memory queries are used only as a sanity check.

Results. As shown in Table 2, AMU achieves the best overall retrieval performance while maintaining the lowest redundancy. It obtains 37.33 P@3, 63.50 R@3, 44.47 F1@3, and 57.50 HIT@3, with a Red.@3 of 2.83. Compared with Full Compressed Memory, AMU improves both precision and recall while reducing Red.@3 from 12.08 to 2.83. This indicates that retaining all compressed memories preserves useful information but also introduces substantial redundant memory traces. Sliding Window performs substantially worse across retrieval metrics, showing that recency-only retention can discard information required for long-term personalization. Among the existing memory-management methods, different approaches exhibit different retrieval–redundancy trade-offs. Mem0 achieves 57.50 R@3 and 40.88 F1@3 while substantially reducing redundancy relative to Full Compressed Memory. AMU further improves its P@3, R@3, and F1@3, while reducing Red.@3 from 3.63 to 2.83. A-MAC obtains the highest recall among the compared existing methods at 58.76, but its lower precision of 30.36 results in an F1@3 of 37.96. AMU improves both precision and recall over A-MAC, indicating a stronger precision–recall balance under our controlled memory-writing setting. A-MEM achieves a Red.@3 of 2.88, which is already close to AMU, but AMU increases R@3 from 56.90 to 63.50 and F1@3 from 40.36 to 44.47. This result is particularly important because AMU's improvement cannot be explained simply by producing a less redundant memory store. Overall, the results show that AMU provides a stronger balance between retrieval effectiveness and memory cleanliness. Existing methods respectively provide explicit memory operations, selective admission, or structured memory evolution, yet AMU achieves the highest P@3, R@3, F1@3, and HIT@3 together with the lowest Red.@3 in our controlled setting. These results support combining structured admission with relation-aware duplicate, update, and separate decisions for maintaining reusable personalized memory before downstream retrieval.

4.4 Component Verification

To verify AMU's two main mechanisms, we conduct ablations with two variants. **AMU w/o Structured Filtering** removes structured filtering from memory writing and retrieval. **AMU w/o SLM Storage Management** keeps structured filtering but removes SLM-guided relation judgment, so `duplicate`, `update`, and `separate` relations are no longer explicitly handled. We report F1@3, Redundancy@3, and **Action Acc.**, defined as the percentage of memory-writing turns whose predicted action matches the gold action label.

As shown in Table 3, removing either mechanism weakens AMU. Without structured filtering, both F1@3 and Action Acc. decrease, indicating weaker noise filtering and retrieval alignment. Without SLM storage management, F1@3 and Action Acc. drop more substantially, showing that explicit duplicate/update/separate judgment is important for memory maintenance. Although this variant has lower Redundancy@3, the low redundancy is accompanied by poor retrieval coverage

Variant	F1@3	Red.@3 ↓	Act. Acc.
w/o Struct. Filtering	34.30	2.25	65.40
w/o SLM Storage Mgmt.	25.62	0.25	34.50
AMU(Ours)	44.47	2.83	67.70

Table 3: Component verification of AMU. Red.@3 denotes Redundancy@3, where lower values are better.

Size	F1@3	Red.@3 ↓	Act. Acc.
0.8B	13.90	0.31	26.40
2B	44.47	2.83	67.70
4B	41.78	3.08	81.00
9B	40.00	3.33	84.20

Table 4: Effect of SLM controller size.

and unreliable writing decisions. These results show that structured memory filtering and SLM-guided storage management are complementary: the former selects records and candidates, while the latter decides how admitted memories are stored, discarded, or updated.

5 Analysis

We analyze how the size of the SLM controller affects memory maintenance, focusing on the relation between writing-action accuracy and retrieval-side behavior. As shown in Table 4, larger controllers improve Action Acc., especially from 0.8B to larger models, indicating that overly small controllers are not stable enough for structured parsing and storage decisions. However, after 4B, the accuracy gain becomes limited, while F1@3 and Red.@3 do not improve proportionally. This suggests that stronger writing-action prediction does not necessarily lead to better retrieval-side performance, since retrieval is also affected by embedding similarity, thresholds, and top-k ranking. The results therefore indicate limited marginal gains from further scaling in this setting, while showing that Qwen3.5-2B is sufficient for AMU without relying on a large response LLM for memory control.

6 Conclusion

In this paper, we presented AMU: Admission and Memory Update for Personalized Conversations. AMU combines structured memory filtering with SLM-guided storage management. To conduct the evaluation, we construct a controlled memory writing and retrieval benchmark to measure whether systems can retrieve expected memory slots while avoiding redundant memories. In addition, we verify the potential of these components for improving memory maintenance. The results demonstrate the value of combining structured memory filtering with SLM-guided storage management for maintaining cleaner personalized memories before downstream retrieval and response generation.

Limitations of the Work

Hardware-Dependent Running Time. All experiments are conducted on a single NVIDIA RTX 4070 GPU, representing an accessible local evaluation setting rather than a large-scale serving environment. This hardware setting provides a practical reference for evaluating AMU under local deployment constraints. However, the running time reported in our experiments should be interpreted together with this setup, since larger SLM controllers may run faster on stronger accelerators or optimized inference infrastructure.

SLM Guidance Dependence. AMU relies on an SLM to extract structured memory records and perform storage-management judgments. Its effectiveness therefore depends on whether the chosen SLM can follow structured prompts, produce stable fields, and distinguish `duplicate`, `update`, and `separate` relations. Very small or poorly aligned SLMs may generate unstable structures or unreliable relation judgments, which can affect both memory writing and downstream retrieval.

Declaration on Generative AI

Generative AI tools were used solely for language polishing during manuscript preparation, including improvements to grammar, and clarity.

Ethics Statement

This work uses a controlled benchmark constructed for memory writing and retrieval. The benchmark is generated and transformed with GPT-4o by adapting textual contexts from LoCoMo-style long-term conversations into AMU's memory-writing setting. It does not rely on real user conversations or personally identifiable private data. The AMU-specific annotations, including memory-writing action labels, memory identifiers, update links, and retrieval targets, are created for controlled evaluation.

AMU is designed for personalized memory management, where stored memories may reflect user preferences, plans, or states. In practical deployment, users should be informed when memory is enabled and should be able to inspect, edit, or

delete stored memories. Since AMU stores memories as structured records, selected memory entries can be removed or updated when users request deletion or correction.

References

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*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Nuo Chen, Hongguang Li, Jianhui Chang, Juhua Huang, Baoyuan Wang, and Jia Li. 2025. Compress to impress: Unleashing the potential of compressive memory in real-world long-term conversations. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 755–773, Abu Dhabi, UAE. Association for Computational Linguistics.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. LongloRA: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *Preprint*, arXiv:2504.19413.

Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. 2020. Queens are powerful too: Mitigating gender bias in dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8173–8188, Online. 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, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Yukun Huang, Yixin Liu, Raghuveer Thirukovalluru, Arman Cohan, and Bhuwan Dhingra. 2024. Calibrating long-form generations from large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13441–13460, Miami, Florida, USA. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

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.

Seungone Kim, Juyoung Suk, Xiang Yue, Vijay Viswanathan, Seongyun Lee, Yizhong Wang, Kiril Gashteovski, Carolin Lawrence, Sean Welleck, and Graham Neubig. 2025. Evaluating language models as synthetic data generators. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6385–6403, Vienna, Austria. Association for Computational Linguistics.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2022. Internet-augmented dialogue generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8460–8478, Dublin, Ireland. Association for Computational Linguistics.

Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. 2025. Hello again! LLM-powered personalized agent for long-term dialogue. 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 5259–5276, Albuquerque, New Mexico. Association for Computational Linguistics.

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

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. In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.

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.

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.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2024. Ret-llm: Towards a general read-write memory for large language models. *Preprint*, arXiv:2305.14322.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Erasmo Purificato, Ludovico Boratto, and Ernesto William De Luca. 2024. User modeling and user profiling: A comprehensive survey. *Preprint*, arXiv:2402.09660.

Mofasshara Rafique and Laurent Bindschaedler. 2026. Clawvm: Harness-managed virtual memory for stateful tool-using llm agents. In *Proceedings of the Sixth European Workshop on Machine Learning and Systems*, EuroMLSys '26, page 1–12. ACM.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. LaMP: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7370–7392, Bangkok, Thailand. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *Preprint*, arXiv:2301.12652.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal,

Arthur Szlam, Y-Lan Boureau, Melanie Kambadur, and Jason Weston. 2022. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *Preprint*, arXiv:2208.03188.

Haoran Sun, Zekun Zhang, and Shaoning Zeng. 2025. Preference-aware memory update for long-term llm agents. *Preprint*, arXiv:2510.09720.

Juntao Tan, Liangwei Yang, Zuxin Liu, Zhiwei Liu, Rithesh R N, Tulika Manoj Awalgaonkar, Jianguo Zhang, Weiran Yao, Ming Zhu, Shirley Kokane, Silvio Savarese, Huan Wang, Caiming Xiong, and Shelby Heinecke. 2025. PersonaBench: Evaluating AI models on understanding personal information through accessing (synthetic) private user data. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 878–893, Vienna, Austria. Association for Computational Linguistics.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Justin Vasselli, Eunike Andriani Kardinata, Yusuke Sakai, and Taro Watanabe. 2025. Multilingual dialogue generation and localization with dialogue act scripting. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32896–32911, Suzhou, China. Association for Computational Linguistics.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Rebecca Westhäußer, Wolfgang Minker, and Sebatian Zepf. 2025. Enabling personalized long-term interactions in llm-based agents through persistent memory and user profiles. *Preprint*, arXiv:2510.07925.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, volume 38, pages 17577–17604. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Guilin Zhang, Wei Jiang, Xiejiashan Wang, Aisha Behr, Kai Zhao, Jeffrey Friedman, Xu Chu, and Amine Anoun. 2026a. Adaptive memory admission control for llm agents. *Preprint*, arXiv:2603.04549.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Zhenzhen Huang, Pengcheng Zheng, Zhicheng Wang, Ping Guo, Fan Mo, Sung-Ho Bae, Jie Zou, Jiwei Wei, and Yang Yang. 2026b. Lightweight llm agent memory with small language models. *Preprint*, arXiv:2604.07798.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory. *Preprint*, arXiv:2305.10250.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, and 2 others. 2026. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering. *Preprint*, arXiv:2604.08224.

Appendix

A Additional Analysis

We provide more details on the behavior of AMU under different settings. The additional analyses include downstream LLM use with different memory settings, sensitivity to writing-time and retrieval-time similarity thresholds, comparisons of embedding models, comparisons of different SLM backbones for memory management, and session mixing analysis with memories from multiple sessions.

A.1 Downstream LLM Judge Evaluation

To examine how the maintained memory affects downstream personalized responses, we conduct a response-level evaluation on sampled cases from each session. For each session, we construct the memory store from the memory-writing turns and randomly sample 5 evaluation queries for response generation. Responses are generated under three settings: without memory, with full compressed memory, and with AMU-maintained memory. The response model is Gemma3-12B with quantized inference, and AMU uses Qwen3.5-2B as the default SLM for memory management. GPT-4o is used as the judge. For each case, the judge is given the user query, the reference memory, the expected behavior, and the assistant response, and assigns an overall score from 1 to 5 based on memory use, avoidance of irrelevant or outdated personal information, and helpfulness.

As shown in Table 5, AMU receives the highest overall score. No Memory lacks access to relevant user information while Full Compressed Memory provides only a modest improvement, suggesting that unfiltered compressed memories may fail to provide useful personalization when irrelevant or redundant information is included. AMU provides a cleaner maintained memory state, allowing the downstream LLM to use relevant personal information more effectively. This analysis is intended as a small-scale response-level supplement rather than a full end-to-end personalization benchmark.

Method	Overall Score
No Memory	2.92
Full Compressed Memory	3.21
AMU	**4.37**

Table 5: Small-scale downstream response evaluation with an LLM judge.

A.2 Threshold Sensitivity

We analyze the sensitivity of AMU to the writing-time similarity threshold τ_s and the retrieval-time similarity threshold τ_r. We set $\tau_s = 0.60$ and $\tau_r = 0.55$ as the default thresholds, and vary one threshold at a time to examine its effect on memory writing, retrieval quality, redundancy, and efficiency.

τ_s	F1@3	Red.@3 ↓	Acc.	Time/Turn ↓
0.40	43.68	2.75	66.80	327.17
0.60	44.47	2.83	67.70	231.34
0.80	45.82	4.50	65.80	176.74

Table 6: Sensitivity to the writing-time threshold τ_s with $\tau_r = 0.55$. Red.@3 denotes Redundancy@3, where lower values are better.

τ_r	P@3	R@3	F1@3	Red.@3 ↓	HIT@3
0.40	36.00	64.25	43.75	2.83	58.00
0.55	37.33	63.50	44.47	2.83	57.50
0.70	45.33	51.25	46.33	2.00	44.50

Table 7: Sensitivity to the retrieval-time threshold τ_r with $\tau_s = 0.60$. Red.@3 denotes Redundancy@3, where lower values are better.

Table 6 shows the effect of varying τ_s while fixing $\tau_r = 0.55$. A smaller τ_s retains more candidate memories for SLM-guided storage management, which increases the average processing time per turn. As τ_s increases, fewer candidates enter relation judgment, reducing Time/Turn. However, a larger τ_s also leaves fewer opportunities for duplicate or update management, which can increase redundancy. In this setting, $\tau_s = 0.60$ provides the highest Action Acc. and a balanced trade-off between writing reliability, retrieval performance, redundancy, and efficiency.

Table 7 shows the effect of varying τ_r while fixing $\tau_s = 0.60$. Increasing τ_r makes retrieval-time filtering stricter, which improves P@3 and Redundancy@3 by suppressing weakly related memories. However, the stricter threshold also reduces R@3 and HIT@3, indicating that some useful memories may be filtered out. Thus, τ_r mainly controls the precision-recall trade-off during retrieval. We use $\tau_r = 0.55$ as the default retrieval-time threshold because it preserves higher recall and HIT@3 while maintaining low redundancy.

A.3 Embedding Model Comparison

We compare three embedding models used in both memory writing and retrieval: Qwen3-Embedding-0.6B, BGE-M3, and Multilingual-E5-Large. For

each setting, the same embedding model is used for writing-time candidate filtering and retrieval-time memory retrieval.

As shown in Table 8(a), changing the embedding model leads to only small differences across retrieval and writing-related metrics. The three embedding models produce close P@3, R@3, F1@3, HIT@3, Redundancy@3, and Action Acc. scores, indicating that replacing the embedding model in both writing-time candidate filtering and retrieval-time memory retrieval does not substantially alter AMU's behavior. This suggests that AMU is not tightly coupled to a specific retriever encoder, and the structured memory filtering and SLM-guided storage management pipeline remains stable across the tested embedding choices.

A.4 SLM Choice for Storage Management

We compare different SLMs used for memory management to examine whether AMU is sensitive to the choice of the SLM backbone. The compared models include Qwen3.5-2B, Qwen3-4B, Gemma-2B-IT, and Gemma4-E2B-IT. All other components, including the embedding model, thresholds, and retrieval pipeline, are kept unchanged.

As shown in Table 8(b), changing the SLM backbone leads to moderate differences but does not substantially change the behavior of AMU. Models with larger parameter sizes generally provide stronger results, especially in writing-action accuracy and retrieval metrics, while smaller models show slightly weaker performance. However, the differences across most settings remain within a limited range, suggesting that AMU is not tightly coupled to a specific SLM backbone. The structured memory filtering and SLM-guided storage management pipeline remains effective across the tested SLM choices.

A.5 Session Mixing Analysis

We further analyze whether AMU is affected when memories from different sessions are mixed in the same memory store. Sessions are randomly sampled to construct single-session, two-session, and three-session memory stores. Each setting is repeated over five random draws, and the reported scores are averaged across runs.

As shown in Table 8(c), AMU remains stable when memories from multiple sessions are mixed in the same memory store. The retrieval metrics stay close across the single-session and mixed-session settings, and no consistent degradation is observed as the number of mixed sessions increases. This suggests that AMU can still select relevant memories from a moderately mixed memory store, which is consistent with the role of structured filtering and storage management.

Table 8: Additional analysis of embedding models, SLM backbones, and session mixing. Red.@3 and Red. denote Redundancy@3, where lower values are better.

(a) Effect of different embedding models.

Embedding Model	P@3	R@3	F1@3	Red.@3 ↓	HIT@3	Action Acc.
Qwen3-Embedding-0.6B	37.33	63.50	44.47	2.83	57.50	67.70
BGE-M3	36.83	63.00	43.97	2.92	57.00	67.10
Multilingual-E5-Large	36.58	62.75	43.68	2.75	56.50	66.90

(b) Effect of different SLM backbones used for memory management.

SLM Backbone	P@3	R@3	F1@3	Red.@3 ↓	HIT@3	Action Acc.
Qwen3.5-2B	37.33	63.50	44.47	2.83	57.50	67.70
Qwen3-4B	37.33	66.00	45.27	2.83	59.00	78.70
Gemma-2B-IT	30.50	55.00	37.22	8.25	55.00	65.70
Gemma4-E2B-IT	37.67	66.75	45.73	1.67	61.50	64.10

(c) Effect of mixing memories from different sessions.

Setting	P@3	R@3	F1@3	Red. ↓	HIT@3
Single	38.68	69.00	47.12	4.34	64.00
2-mix	38.00	66.00	45.98	2.66	60.00
3-mix	42.52	69.00	47.14	4.00	64.00

Remember, Verify, or Ask? Evaluating Memory Decisions at Reuse Time

Baichuan Li[1,*] **Junyi Yao**[2,*] **Zihao Zheng**[2,*]
[1]Southern Methodist University
Dallas, Texas, USA
[2]Washington University in St. Louis
St. Louis, Missouri, USA
baichuanl@smu.edu
j.yao@wustl.edu; z.zihaogary@wustl.edu

Abstract

Persistent memory can personalize an LLM agent, but an incorrect durable update can distort future behavior. We introduce the Memory–Clarification Boundary benchmark (MCB), which supplies an interaction-derived candidate and a later reuse context and asks whether to persist it, use it locally, verify it, or clarify it with the user. The release contains 140 primary scenarios, 70 evidence-flip contrasts, and a 36-item real-dialogue-derived extension. On 70 adjudicated primary-test items, verification recall exceeds clarification recall in all 12 label conditions across four checkpoints. However, only three comparisons remain significant after an exploratory Fisher test with Holm correction, and none involve Claude. A category oracle attains .800 accuracy, exposing strong scenario-category structure. Original label and tool-selection conditions agree on only 23–57% of items, although their differing prompts prevent attributing this solely to the interface. The naturalistic extension is descriptive: it contains only two clarification items and evaluates local models only. MCB measures reuse-time decision selection under a conservative policy; storage backends, matched policy/tool interventions, and downstream utility remain unevaluated.

1 Introduction

LLM agents retain interaction history to personalize later behavior and support long-horizon tasks (Packer et al., 2023; Chhikara et al., 2025). Memory formation nevertheless requires a decision about scope and authority. A temporary request should not become a standing preference; an old service status may require a current check; and an ambiguous correction may require a question before it changes future behavior.

We study the *memory–clarification boundary*. Given an interaction-derived candidate and a later reuse context, an agent selects one of four actions: *persist*, *ephemeral* use, *verify* against an external source, or *clarify* with the user. Verification and clarification resolve different uncertainties. An external source establishes changing world facts, while the user establishes intent, scope, and ambiguous reference.

The evaluated decision occurs *at reuse time*: the reuse context is already visible to the model. We do not test an acquisition-time filter that predicts which information will be useful before future contexts are known. A reuse-time decision could authorize a subsequent write or defer it, but this benchmark records the decision only. Applications to vector, profile, episodic, or graph memories are architectural proposals rather than demonstrated backend-independent results.

The Memory–Clarification Boundary benchmark (MCB) contains 140 primary scenarios split into 70 development and 70 held-out items, a separate 70-item contrast extension, and 36 independently adjudicated items derived from public dialogues. We evaluate two Claude checkpoints and two local models, Qwen3.5-9B and Gemma4-E4B, under bare, original policy, and few-shot prompts. MCB-ACT asks for a structured tool call instead of a label. The original policy omits several annotation rules, and the original tool condition has no policy prompt; both facts constrain interpretation.

Our contributions are:

1. an auditable four-action benchmark with frozen data, annotations, prompts, and item-level outputs supplied as a downloadable supplementary artifact;
2. class-specific and baseline analyses that distinguish observed recall differences from evidence of a general model property; and
3. a comparison of original label and tool-selection settings, identifying sensitivity that

[*]Baichuan Li, Junyi Yao, and Zihao Zheng contributed equally and share first authorship.

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 15–25
November 10, 2026 ©2026 Association for Computational Linguistics

motivates a future matched-prompt evaluation.

2 Reuse-Time Memory Decisions

Let a candidate update u be extracted from an interaction history h and reconsidered in reuse context c. A commitment policy selects

$$g(h, u, c) \in \{P, E, V, C\}, \tag{1}$$

where P persists the update, E limits it to the current context, V rechecks an external source, and C queries the user. The distinction is based on authority and scope rather than topic words. A restaurant's hours and a user's preferred meeting time may both mention time; the first is world-authoritative and freshness-sensitive, whereas the second may be a durable preference or an ambiguous one-off request.

Figure 1 illustrates the evaluated choice and possible downstream handling. The candidate is supplied rather than extracted by the evaluated model; it need not already exist in durable storage. A system could insert or update it after choosing P, or defer writing after V or C. We do not execute those transitions or assess an existing memory store.

The gate is conservative by design. When persistence and a weaker action remain tied, the stated rules prefer the weaker commitment. This is an assumed cost preference: a wrong durable update can affect later behavior, while an unnecessary question is visible. We do not measure these costs or establish that the preference is optimal.

3 Related Work

Recall and memory lifecycle. LongMemEval, LoCoMo, MemoryBank, and MemBench evaluate long-term conversational memory, knowledge updates, retrieval, and abstention (Wu et al., 2025; Maharana et al., 2024; Zhong et al., 2024; Tan et al., 2025). Newer benchmarks add real-dialogue memory lifecycles and penalties for obsolete-memory reuse (Xiao et al., 2026a; Uddin et al., 2026). These tasks largely ask whether stored information can be recovered or correctly reused. MCB supplies a candidate update and reuse context and scores the selected response to its scope and authority, without executing retrieval or storage.

Storage, editing, and action. PerMemBench learns a binary session-level storage gate (In et al., 2026); Memory-R1 learns ADD/UPDATE/DELETE/NOOP operations (Yan et al., 2025); and Mem2ActBench tests whether retrieved memory grounds later tool parameters (Shen et al., 2026). Mem2ActBench evaluates how retrieved memory supports action; MCB scores a supplied candidate at reuse time, including whether it should authorize a durable update. MCB also separates two types of unresolved uncertainty—checking the world and asking the user—that a binary gate conflates.

Structured and graph memory. A-MEM dynamically links and evolves interconnected memory records (Xu et al., 2025), while Zep represents evolving agent memory in a temporal knowledge graph (Rasmussen et al., 2025). Such systems improve organization, retrieval, and temporal reasoning after information reaches the memory layer. Our focus is complementary: we score a candidate-handling decision, without implementing or comparing either memory architecture. Graph insertion and propagation risks are motivations for future testing.

Clarification and interactive agents. CLAMBER documents broad failures to identify and clarify ambiguous needs; related work uses uncertainty or expected value to decide when to ask (Zhang et al., 2024; Testoni and Fernández, 2024; Zhang and Choi, 2025). Interactive benchmarks such as τ-bench and τ^2-bench show that static answers can diverge from tool-mediated behavior (Yao et al., 2024; Barres et al., 2025). MCB-ACT applies this observation at the memory boundary by replacing labels with structured memory operations.

Grounded operational agents. A YANG-grounded agent combines schema-aware retrieval with post-generation validation for OpenROADM network management (Xiao et al., 2026b). Technical-model validation provides a complementary constraint: it does not establish whether interaction-derived information warrants durable memory or requires user clarification.

4 Benchmark and Method

4.1 Task and Labels

Each item contains an acquisition context, a candidate update, and a later reuse context. The labels operationalize the stated conservative policy rather than universal ground truth. Under that policy:

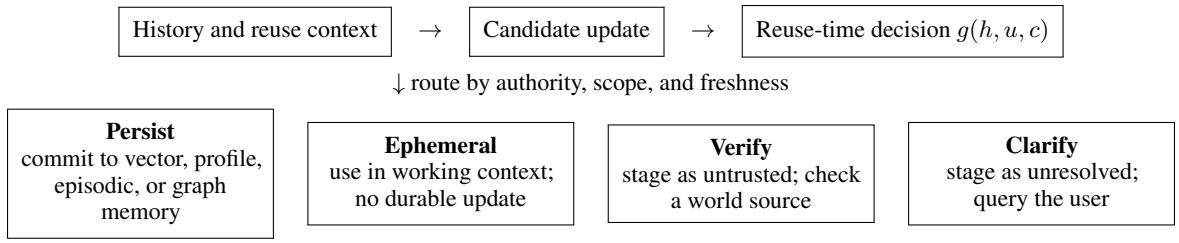


Figure 1: The evaluated reuse-time choice and proposed downstream routes. MCB records action selection; storage writes, source checks, user answers, and backend-specific transitions are not executed.

Table 1: Positioning by explicit scored target. A dash means the capability is not a central scored target, not that the dataset can never contain such an event. "Structured action" includes tool selection or parameterized memory operations.

Benchmark	Recall/reuse	Commitment choice	Ask user	Check world	Structured action
LongMemEval / LoCoMo (Wu et al., 2025; Maharana et al., 2024)	✓	–	–	–	–
MemBench (Tan et al., 2025)	✓	–	–	–	–
PerMemBench (In et al., 2026)	✓	✓	–	–	–
CLAMBER (Zhang et al., 2024)	–	–	✓	–	–
Mem2ActBench (Shen et al., 2026)	✓	–	–	–	✓
MCB (ours)	–	✓	✓	✓	✓

- **persist**: explicitly durable preferences, authorized policies, or stable user/organization facts;
- **ephemeral**: information scoped to one artifact, session, or time window;
- **verify**: changing world state or a single noisy signal that must be rechecked; and
- **clarify**: unresolved referent, durability, scope, conflict, or second-hand preference for which the user is authoritative.

Table 2: Illustrative MCB decisions.

Candidate and reuse context	Gold	Boundary cue
"I always prefer dark mode"; configure a new workspace	Persist	Explicitly durable preference
"Use APA for this report"; prepare an unrelated report	Ephemeral	One-artifact scope
Restaurant is open tonight; rely on it next month	Verify	World state can change
"Make it like last time"; several artifacts fit	Clarify	User must resolve referent

The seven scenario categories are stable preference, episodic preference, freshness-sensitive fact, one-off correction, policy constraint, ambiguous update, and noisy failure signal. Category identity remains highly predictive of the adjudicated action, as the category oracle in Table 4 demonstrates. Eight lexical traps place cues such as "always" or "today" in contexts where the usual heuristic action is wrong. Models never receive category names or rationales. Each category contributes 10 primary-test items (14.3% of that split); Table 3 gives action counts, and Appendix C gives their cross-tabulation.

4.2 Splits and Independent Annotation

The 140 primary items are deterministically split 70/70 by sorted identifier within category. The development split supplies four demonstrations, one per action. A separate 70-item contrast extension contains 35 evidence-flip pairs and is used only as a controlled sensitivity analysis.

Two non-authors independently labeled the 70 held-out primary and 70 contrast items while blinded to author labels, categories, rationales, model outputs, and each other. Agreement across all 140 items was 0.971 ($\kappa = 0.962$); agreement was 0.943 ($\kappa = 0.923$) on the primary test and 1.000 on the contrast set. A blind third non-author labeled only four disagreements. Majority adjudication changed eight primary labels and no contrast labels. The audited held-out distribution is 18 persist, 22 ephemeral, 18 verify, and 12 clarify. These correspond to 25.7%, 31.4%, 25.7%, and 17.1%, respectively.

Split	P	E	V	C
Development	20	18	15	17
Primary test	18	22	18	12
Contrast	20	20	15	15
Naturalistic	7	21	6	2

Table 3: Gold-action counts: persist (P), ephemeral (E), verify (V), clarify (C). Development labels are author-written; the other splits use non-author adjudication. Primary-test scenario categories each contain 10 items.

4.3 Real-Dialogue-Derived Extension

We sample 10,000 rows from a frozen revision of the public non-toxic WildChat corpus (Zhao et al., 2024), without using model outputs. Forty English user turns pass automated privacy/safety screens and a pre-annotation quality review. The user turn is verbatim apart from whitespace; a coordinator paraphrases one candidate update before annotation; and every item receives the same counterfactual reuse context 30 days later. Thus, this is a source-grounded naturalistic check, not a longitudinal memory log.

Two new non-authors independently judge usability and action. Usability agreement is .925; among 35 items both mark usable, action agreement is .743 ($\kappa = .603$). A blind third non-author labels 12 disagreements. Majority vote excludes four unsupported candidates and retains 36: 21 ephemeral, 7 persist, 6 verify, and 2 clarify. Model inference begins only after this freeze.

4.4 Label and Tool-Selection Evaluation

Label prompts request JSON with one action. *Bare* provides definitions only; *policy* adds the original five rules; *few-shot* adds four development examples. The original policy states ambiguous referent/scope/durability rules but omits explicit instructions for conflicts, second-hand preferences, and consequential reuse after a preference's scope expires. Its other rules also compress the annotation guide. These are evaluations of the archived prompts, not fully guide-aligned interventions.

MCB-ACT requests one JSON call: `memory_write(content)`, `use_now(note)`, `check_source(query)`, or `ask_user(question)`. It describes the tools and supplies no five-rule policy. Tool names map to the four actions; deterministic checks assess minimum argument format and relevance. The harness records selection without executing storage, source, or user effects. Its role framing and wording also differ from label mode, so this comparison does not isolate output format as a causal factor.

Appendix B documents the original prompts' coverage and a complete guide-aligned policy for label and tool outputs. The aligned policy and an act-plus-policy condition have not been evaluated; no reported score is attributed to them.

4.5 Metrics and Statistics

Accuracy measures the proportion of correct decisions. Precision measures how often a selected action is correct; recall measures how often the model selects it when required. F1 combines precision and recall, and macro-F1 weights all four actions equally. Precision is set to zero when an action is never predicted. Over-memory (OM) is erroneous persistence divided by all items. Under-memory (UM) is the proportion of gold-persist items missed; OM and UM therefore have different denominators, 70 and 18 on the primary test. Both appear in Table 5.

For primary accuracy and accuracy differences, 2,000 bootstrap resamples quantify uncertainty due to the evaluated items. Paired resampling preserves the same item across compared runs. Exact two-sided McNemar tests count items on which one run alone is correct. Holm correction adjusts for testing multiple comparisons: eight original prompt comparisons and four label–tool comparisons form separate families. We call an adjusted $p < .05$ significant within its family; larger values do not establish absence of an effect. Intervals are unadjusted, so excluding zero need not imply significance after multiple-comparison correction.

For the verify–clarify recall comparison, a two-sided Fisher test compares correct/incorrect counts on 18 verify and 12 clarify items. Holm adjustment covers all 12 label conditions. These tests were added after review and are *exploratory*. Recall-difference intervals resample separately within each gold class. Class difficulty and prompt construction differ, so significance would concern this item set, not a universal model tendency. The small naturalistic extension is reported descriptively, without p-values or significance claims.

5 Experimental Setup

We evaluate Claude Haiku 4.5 (`claude-haiku-4-5-20251001`), Claude Sonnet 4.6 (`claude-sonnet-4-6`), Qwen3.5-9B (Qwen Team, 2026), and Gemma4-E4B (Gemma Team,

Reference	Acc.	F1	OM	UM	C/V R
Always-Persist	.257	.102	.743	.000	.000/.000
Majority action	.314	.120	.000	1.000	.000/.000
Keyword rules	.257	.228	.071	.722	.833/.056
Category oracle	.800	.792	.071	.167	.667/.889

Table 4: Primary-test references ($n = 70$). F1 is macro-F1; C/V R denotes clarify/verify recall. Majority action and category oracle use test labels, and the oracle also sees hidden categories. These two rows are distribution diagnostics, not deployable systems.

2026). Claude calls are isolated per item; served IDs are recorded.

Both open-weight models run locally through Ollama (Ollama, 2026) with Q4_K_M quantization, temperature 0, seed 13, disabled thinking, and isolated item calls. Qwen has 9.7B parameters; the Gemma checkpoint reports 8.0B total parameters (4.5B effective). Complete model digests, runner versions, prompts, token limits, raw JSON, parsed actions, and token counts accompany every prediction. The Gemma digest is `c6eb396dbd59...cf982eb`. All 280 Gemma outputs parse successfully.

Reference systems are Always-Persist, a fixed temporal/scope keyword heuristic, Majority Action, and a category-majority oracle. Majority Action uses the evaluated split's most frequent gold action; the oracle uses both test gold labels and category IDs hidden from models. These two are distribution diagnostics, not deployable baselines. The fixed keyword rules receive the same scenario text as models and do not inspect test labels.

The category oracle scores .800 accuracy and .792 macro-F1. Sonnet bare scores .814/.790: these similar results show how much this authored benchmark's category structure explains. Several model conditions exceed .800, but the oracle's access to test labels and hidden metadata prevents interpreting it as a fair trained competitor. Low keyword accuracy (.257) does not establish that the benchmark lacks exploitable structure.

6 Results

6.1 Recall Differences and Their Uncertainty

Verification recall exceeds clarification recall in all 12 original label conditions (Table 6). After Fisher tests with Holm correction, only Qwen bare ($p_{\mathrm{H}} = .004$), Qwen policy ($p_{\mathrm{H}} < .001$), and Gemma policy ($p_{\mathrm{H}} = .012$) remain significant. No Claude comparison passes this correction. The pattern is descriptive across conditions; it does not establish a general cross-model tendency to prefer verification.

Precision also changes the interpretation. Haiku bare has verify precision .472 versus clarify precision .667; Qwen policy has verify precision .486 versus clarify precision 1.000, but selects clarify correctly on only 1/12 items. High verify recall can coexist with over-selection of that action, and high clarify precision with few opportunities taken. Both classes' precision, recall, and F1 are reported rather than treating recall alone as capability.

Errors remain model-specific: bare Qwen maps seven clarify items to persist and four to verify; Gemma policy maps two to persist, four to verify, and three to ephemeral. Because the original policy omits annotation rules, these counts cannot distinguish prompt coverage failures from intrinsic difficulty. The 12 clarify and 18 verify items also differ in content and difficulty.

6.2 Prompt Effects on Accuracy and Action Rates

Using the expanded eight-comparison family, Haiku policy is significant relative to bare ($\Delta = +.229$, Holm-adjusted $p_{\mathrm{H}} = .003$), and Qwen few-shot remains significant ($\Delta = +.214$, $p_{\mathrm{H}} = .002$). Haiku's few-shot gain is not significant after correction ($p_{\mathrm{H}} = .070$). Gemma's policy and few-shot gains, $+.100$ and $+.086$, are also not significant ($p_{\mathrm{H}} = .898$ for each).

Behavior rates reveal effects hidden by accuracy. Qwen policy reduces OM from .243 to .100 and Gemma policy from .171 to .071, but after the expanded eight-test correction these changes are not significant ($p_{\mathrm{H}} = .051$ and .273). Both prompts increase verification recall more than clarification recall, but their incomplete guide coverage prevents concluding that a fully specified policy fails to improve clarification.

6.3 Sensitivity to Tool-Selection Settings

Table 7 compares each bare label run to its tool-selection run. Sonnet, Qwen, and Gemma have significant accuracy drops after correction; Haiku does not ($p_{\mathrm{H}} = .057$). Raw label–tool agreement is at most .571 and falls to .243 for Gemma. These are not parser artifacts: invalid rate is zero, and every selected Gemma tool includes a payload that passes deterministic well-formedness checks.

Gemma provides the strongest stress test. In label mode it distributes decisions across all four

Model	Condition	Accuracy [95% CI]	Macro-F1	OM↓	UM↓	Clarify R	Verify R
Claude Haiku 4.5	bare	.629 [.514,.743]	.615	.029	.167	.500	.944
	policy	.857 [.771,.929]	.846	.014	.111	.750	.944
	few-shot	.757 [.643,.857]	.732	.057	.056	.500	.944
	act	.514 [.400,.629]	.456	.143	.167	.583	.778
Claude Sonnet 4.6	bare	.814 [.714,.900]	.790	.057	.056	.500	.889
	policy	.843 [.757,.929]	.831	.014	.167	.667	1.000
	few-shot	.814 [.729,.900]	.796	.043	.111	.583	.944
	act	.529 [.414,.643]	.531	.100	.278	.500	.556
Qwen3.5-9B	bare	.557 [.443,.671]	.450	.243	.000	.000	.667
	policy	.629 [.514,.743]	.542	.100	.056	.083	.944
	few-shot	.771 [.671,.871]	.726	.129	.000	.333	.833
	act	.343 [.243,.457]	.236	.057	.833	.083	.056
Gemma4-E4B	bare	.629 [.514,.743]	.608	.171	.111	.417	.722
	policy	.729 [.629,.829]	.675	.071	.167	.250	.889
	few-shot	.714 [.600,.814]	.677	.143	.056	.333	.833
	act	.157 [.086,.243]	.116	.486	.611	.000	.056

Table 5: Original held-out conditions on adjudicated gold ($n = 70$). OM divides erroneous writes by all 70 items; UM divides missed persist decisions by 18 gold-persist items. Recall (R) uses 12 clarify and 18 verify items. Invalid rate is zero. Act requests one tool and argument from tool descriptions only, without the original five policy rules.

Model	Prompt	Clarify P/R/F1	Verify P/R/F1	Δ recall [95% CI]	p_{H}
Haiku	bare	.667/.500/.571	.472/.944/.630	.444 [.139,.750]	.077
Haiku	policy	.692/.750/.720	.810/.944/.872	.194 [-0.028,.500]	.274
Haiku	few-shot	.600/.500/.545	.708/.944/.810	.444 [.139,.750]	.077
Sonnet	bare	.857/.500/.632	.762/.889/.821	.389 [.056,.694]	.102
Sonnet	policy	.800/.667/.727	.783/1.000/.878	.333 [.083,.583]	.090
Sonnet	few-shot	.700/.583/.636	.810/.944/.872	.361 [.083,.667]	.102
Qwen	bare	.000/.000/.000	.480/.667/.558	.667 [.444,.889]	.004
Qwen	policy	1.000/.083/.154	.486/.944/.642	.861 [.639,1.000]	<.001
Qwen	few-shot	.667/.333/.444	.789/.833/.811	.500 [.167,.806]	.077
Gemma	bare	.625/.417/.500	.619/.722/.667	.306 [-0.056,.639]	.272
Gemma	policy	.750/.250/.375	.640/.889/.744	.639 [.333,.917]	.012
Gemma	few-shot	.571/.333/.421	.833/.833/.833	.500 [.167,.833]	.077

Table 6: Class-specific precision (P), recall (R), and F1 on the primary test. Δ is verify minus clarify recall; intervals resample within each gold class. Two-sided Fisher tests compare 18 verify and 12 clarify items; Holm adjustment covers all 12 label conditions. Tests are post-review exploratory analyses.

actions and reaches .629 accuracy. In tool mode it calls `memory_write` 41 times, `use_now` 28 times, `check_source` once, and `ask_user` zero times. Its accuracy falls by .471 ($p_{\mathrm{H}} < .001$), OM rises to .486, and both clarification and verification recall collapse. Valid syntax and minimum argument checks do not establish correct operation choice. Because the label and tool prompts differ, these results demonstrate sensitivity to the two original settings rather than an isolated interface effect.

Table 7: Act minus bare label accuracy. p_{H} uses the four-model tool-selection family.

Model	Δ [95% CI]	p_{H}	Agreement
Haiku	−.114 [−.214,−.014]	.057	.571
Sonnet	−.286 [−.414,−.143]	<.001	.571
Qwen	−.214 [−.386,−.029]	.047	.229
Gemma	−.471 [−.614,−.329]	<.001	.243

6.4 Controlled Contrast Validation

The evidence-flip extension provides a controlled check on Qwen prompt effects beyond the primary 70 items. On the combined 140 Qwen items, bare, policy, and few-shot accuracy is .614, .757, and .843. Relative to bare, policy gains .143 and few-shot gains .229 (both $p_{\mathrm{H}} < .001$ within the two-test contrast family). Clarification remains the weakest class (.074/.407/.519). Both annotators reproduced all 70 contrast labels, but the examples are rule-authored templates aligned with the policy rules. We therefore retain this set as controlled sensitivity evidence, not naturalistic external validation.

Table 8: Descriptive naturalistic results ($n = 36$): six verify and two clarify items. F1 is macro-F1. No significance tests are reported; Claude is not evaluated on this split.

Model	Cond.	Acc.	F1	OM↓	Ver.	Clar.
Qwen3.5	bare	.472	.381	.056	.833	.000
	policy	.444	.352	.111	1.00	.000
	fewshot	.472	.365	.306	.500	.000
	act	.306	.177	.000	1.00	.000
Gemma4	bare	.167	.071	.000	1.00	.000
	policy	.306	.277	.028	1.00	.000
	fewshot	.472	.411	.389	.833	.000
	act	.444	.399	.528	.667	.000

6.5 Descriptive Naturalistic Check

Table 8 reports Qwen and Gemma; Appendix D gives all 16 local conditions, including exploratory Phi-4-mini and Qwen3-4B runs. Claude was not evaluated on this split. The majority-ephemeral reference reaches .583 accuracy but only .184 macro-F1. No local condition exceeds its accuracy. Qwen's best accuracy is .472; Gemma changes from .167 with bare prompting to .472 with few-shot prompting. These are descriptive differences, not significant effects.

Only two items require clarification. All 16 local conditions miss both. With two such examples, one correct decision would change clarification recall by .50; their failure does not provide a precise population recall estimate or corroborate a cross-model property. Even 0/2 has a 95% Wilson interval of approximately [0,.658]. The split contains six verify items and 21 ephemeral items, so its class-specific results are also unevenly supported.

Qwen's tool accuracy is .306 versus .472 in bare label mode, with agreement .556. Gemma's tool accuracy is .444 versus .167, with agreement .111: its 36 verify labels become 26 `memory_write` calls among the tool outputs, and OM reaches .528. All tool arguments pass the minimum checks. These observations illustrate why accuracy alone can conceal erroneous persistence, but do not establish an effect outside this small local-model sample.

7 Design Implications and Untested Extensions

A selected action could be connected to a memory backend, but the present results establish only selection behavior given a reuse context. In a proposed implementation, persist could authorize a durable update, ephemeral could leave the candidate local, verify could stage it pending an external check, and clarify could defer it pending a user answer. These states could be represented in a vector-store metadata layer, profile, episodic record, or temporal graph. We have not compared those implementations or tested the effects on retrieval and downstream utility.

For graph memory, unresolved candidates could carry provenance and restricted retrieval permissions until resolved. Such handling is a design hypothesis, not a demonstrated remedy. A future evaluation should distinguish acquisition-time decisions without known reuse context from the reuse-time decision tested here.

The original label/tool disagreement motivates matching the complete decision rules, examples, and role framing across interfaces, including act-plus-policy. End-to-end testing would additionally execute source checks, ask users, update storage, and measure later task outcomes. ON-RAP's device compliance checks, interoperability tests, and performance measurements provide an operational example of multistage validation (Xiao et al., 2025); adapting that approach to a memory gate remains future work.

8 Conclusion

MCB scores persist, ephemeral use, verification, and user clarification for an interaction-derived candidate at reuse time. On the authored primary test, verify recall exceeds clarify recall in all original label conditions, but only three exploratory comparisons survive correction. The .800 category oracle shows that scenario structure explains much of the attainable accuracy. Original label and tool-selection settings also produce substantially different decisions, with prompt differences limiting causal attribution. The small local-model naturalistic check is descriptive. The benchmark and frozen artifact support auditing these specific findings; evaluating complete policies, matched tool interfaces, larger clarification samples, and downstream memory systems is the next step.

9 Limitations

The primary test has 70 authored English items, including 12 clarify and 18 verify cases; each scenario category has 10 items. The category oracle's .800 accuracy reveals strong category/action structure. Class content and difficulty differ, so the recall comparison cannot establish an intrinsic preference for verification. Its Fisher tests are exploratory analyses added after review. The contrast set con-

sists of rule-authored templates, and development demonstrations retain author labels.

The original policy omits conflict, second-hand preference, and expired-scope clarification rules and compresses other guide instructions. Complete guide-aligned label and tool policies are documented but have not been run. The original tool condition differs in wording and role framing as well as output format; no act-plus-policy comparison isolates these factors. We therefore do not claim that full policy prompting fails to fix clarification or that an interface change alone causes the observed differences.

The naturalistic extension has 36 items, only two requiring clarification, and no Claude evaluation. It uses public user turns with coordinator-written candidates and a standardized counterfactual reuse context. It is neither a longitudinal deployment study nor a statistically stable clarification evaluation. Its local-model descriptive results do not validate a property of all four primary-test checkpoints.

We use one serving configuration per checkpoint and one quantized version per local model. The Gemma checkpoint has 8B total and 4.5B effective parameters; comparisons with dense models do not isolate scale. The benchmark records one decision without extracting candidates, executing writes or checks, obtaining user answers, or measuring retrieval and task utility. Reuse context is available at decision time. Acquisition-time gating and claims of representation-independent performance remain untested. The conservative policy also assumes a cost asymmetry that applications may choose differently.

10 Ethical Considerations

The primary and contrast benchmarks contain no personal records. The frozen naturalistic extension quotes minimized turns from a public, consent-based research dataset; it excludes records flagged toxic or redacted and omits location, timestamp, IP hash, and request-header fields, but automated de-identification cannot guarantee zero residual privacy risk. Its intended use is to study decisions that may contribute to inappropriate durable memory; we do not measure a deployed reduction in such failures. Nevertheless, a clarification policy can burden users, and a verification policy can expose sensitive queries to external services. Real systems should minimize data disclosure, record provenance and consent, support correction and deletion, and calibrate intervention costs with affected users. Benchmark scores should not be treated as permission to deploy a memory-writing agent without domain-specific privacy, security, and human-factors evaluation.

11 Declaration on Generative AI

Generative AI systems were used as experimental subjects and for implementation and language assistance. The authors designed the benchmark, prompts, annotation protocol, analyses, and claims; non-author annotators supplied the reported independent labels. All reported numerical results were generated from frozen item-level outputs by deterministic analysis code and checked against the manuscript. Post-review revision and analysis assistance introduced no new model inference or annotation. The authors remain responsible for the paper's content.

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *Preprint*, arXiv:2506.07982. ArXiv:2506.07982.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready AI agents with scalable long-term memory. *Preprint*, arXiv:2504.19413. ArXiv:2504.19413.

Gemma Team. 2026. Gemma 4 technical report. *Preprint*, arXiv:2607.02770. ArXiv:2607.02770.

Yeonjun In, Wonjoong Kim, Sangwu Park, Kanghoon Yoon, and Chanyoung Park. 2026. Personalize-then-store: Benchmarking and learning personalized memory for long-horizon agents. *Preprint*, arXiv:2605.25535. ArXiv:2605.25535.

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 (ACL)*, pages 13851–13870. Association for Computational Linguistics.

Ollama. 2026. Ollama: Run large language models locally. Software.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *Preprint*, arXiv:2310.08560. ArXiv:2310.08560.

Qwen Team. 2026. Qwen3.5-9B model card. Hugging Face model repository. Accessed: 2026-08-13.

Preston Rasmussen et al. 2025. Zep: A temporal knowledge graph architecture for agent memory. *Preprint*, arXiv:2501.13956. ArXiv:2501.13956.

Yiting Shen, Kun Li, Wei Zhou, and Songlin Hu. 2026. Mem2ActBench: A benchmark for evaluating long-term memory utilization in task-oriented autonomous agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pages 8173–8190, San Diego, California, USA. Association for Computational Linguistics.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19336–19352. Association for Computational Linguistics.

Alberto Testoni and Raquel Fernández. 2024. Asking the right question at the right time: Human and model uncertainty guidance to ask clarification questions. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 258–275. Association for Computational Linguistics.

Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. 2026. From recall to forgetting: Benchmarking long-term memory for personalized agents. In *Findings of the Association for Computational Linguistics: ACL 2026*. Association for Computational Linguistics. ArXiv:2604.20006.

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 *International Conference on Learning Representations (ICLR)*.

Jianfei Xiao, Xiang Yu, Chengbing Wang, Wuqiang Zheng, Xinyu Lin, Kaining Liu, Hongxun Ding, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2026a. AlpsBench: An LLM personalization benchmark for real-dialogue memorization and preference alignment. *Preprint*, arXiv:2603.26680. ArXiv:2603.26680.

Linqi Xiao, Aparaajitha Gomathinayakam Latha, Balagangadhar Bathula, Tianliang Zhang, and Andrea Fumagalli. 2025. Optical network robotic automation platform for multi-vendor optical transport network: Accelerating validation, troubleshooting, and proof of concept efforts. In *2025 International Conference on Optical Network Design and Modeling (ONDM)*, pages 1–6.

Linqi Xiao, Aparaajitha Gomathinayakam Latha, Venkateswarlu Gudepu, and Andrea Fumagalli. 2026b. A YANG-grounded LLM agent supporting multi-vendor OpenROADM optical transport networks. In *Proceedings of the ACM SIGCOMM 2026 Conference*, pages 2109–2115. Association for Computing Machinery.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *Preprint*, arXiv:2508.19828. ArXiv:2508.19828.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *Preprint*, arXiv:2406.12045. ArXiv:2406.12045.

Michael J. Q. Zhang and Eunsol Choi. 2025. Clarify when necessary: Resolving ambiguity through interaction with LMs. In *Findings of the Association for Computational Linguistics: NAACL*, pages 5541–5558. Association for Computational Linguistics.

Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. 2024. CLAMBER: A benchmark of identifying and clarifying ambiguous information needs in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 10746–10766. Association for Computational Linguistics.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

A Reproducibility and Artifact Scope

The accompanying archive `NORA2026_camera_ready_supplement.zip` supplies the benchmark and executable reproduction code. It includes all 140 primary items, 70 contrast items, frozen non-author annotation decisions, prompts, checkpoint metadata, item-level predictions, and tests. Naturalistic materials include all 40 screened candidates, the 36 adjudicated items, minimized source provenance, three non-author packets, and all 16 local-model runs. The archive is provided as supplementary material; no public repository URL is claimed here.

Run `python reproduce_camera_ready.py` from the extracted archive using Python 3.9 or newer. Model access is not needed. This validates the frozen data and regenerates the primary baselines, class metrics, recall comparisons, and descriptive naturalistic analysis. The original prompts and outputs remain unchanged; the full-policy specification below is separately versioned and unevaluated. The original manuscript and submission archive are preserved outside the revised release.

B Prompt Coverage and Complete Policy

The original label policy has five rules: persist explicitly durable preferences, authorized policy-like constraints, or stable personal/organizational facts; keep explicitly scoped instructions ephemeral; verify changing facts or a single noisy failure signal; clarify ambiguous or hedged referent, scope, or durability; and prefer weaker commitment when persistence remains uncertain. It does not explicitly mention conflicts, second-hand preferences, or the annotation guide's consequential expired-scope cases. The original act prompt instead lists the four tools and their arguments, without these policy rules. Appendix material does not retroactively change either evaluated prompt.

For future matched comparisons, the following complete specification covers the guide. It is supplied as executable prompt construction in `src/aligned_policy.py`, with identical decision rules across label and tool output formats. No results in this paper use it.

1. Persist deliberately stated standing preferences, constraints from an authorized source, and stable personal or organizational facts. A durable fact can appear in a local correction; content takes precedence over surface phrasing.

2. Keep clearly artifact-, session-, or time-scoped instructions and local edits ephemeral. A failure clearly caused by temporary context can be retried or left local without a durable belief.

3. Verify changing world state when later reuse exceeds its shelf life, a noisy resource failure when a current probe is the appropriate next step, and second-hand or dated policy claims against an available authoritative source.

4. Clarify unresolved referents, consequential ambiguity about scope or durability, conflicts whose precedence is unclear, second-hand preferences, and expired preferences when consequential reuse makes their continuing validity unclear.

5. For an explicitly ended scope with no consequential uncertainty, choose ephemeral. Distinguish user authority (intent, preferences, referents) from external authority (state, versions, schedules). Prefer weaker commitment when persistence remains uncertain.

Scenario category	P	E	V	C
Ambiguous update	0	2	0	8
Episodic preference	0	9	0	1
Freshness sensitive fact	1	0	9	0
Noisy failure signal	0	3	7	0
One off correction	2	8	0	0
Policy constraint	8	0	2	0
Stable preference	7	0	0	3

Table 9: Primary-test category/action cross-tabulation. Each scenario category has 10 items; action labels are not the category labels.

C Class Distributions and Full Primary Metrics

Table 9 makes category/action structure explicit. Table 10 reports precision, recall, and F1 for all four actions and all original primary conditions. Wilson intervals for each clarify and verify recall, exact contingency counts, raw Fisher probabilities, Holm-adjusted probabilities, and stratified-bootstrap intervals are included in the supplementary `results/camera_ready/analysis.json`.

D Full Descriptive Naturalistic Results

All 16 local conditions use the same 36 items. Qwen and Gemma were selected in the frozen protocol, with Phi-4-mini and Qwen3-4B as exploratory checks. No comparison in this section receives a significance claim; there are only two clarify items. Overall intervals describe variability across this item sample, not uncertainty from repeated model sampling.

Model	Condition	Persist P/R/F1	Ephemeral P/R/F1	Verify P/R/F1	Clarify P/R/F1
Claude Haiku 4.5	bare	.882/.833/.857	.750/.273/.400	.472/.944/.630	.667/.500/.571
	policy	.941/.889/.914	.947/.818/.878	.810/.944/.872	.692/.750/.720
	few-shot	.810/.944/.872	.867/.591/.703	.708/.944/.810	.600/.500/.545
	act	.600/.833/.698	.000/.000/.000	.700/.778/.737	.292/.583/.389
Claude Sonnet 4.6	bare	.810/.944/.872	.857/.818/.837	.762/.889/.821	.857/.500/.632
	policy	.938/.833/.882	.857/.818/.837	.783/1.000/.878	.800/.667/.727
	few-shot	.842/.889/.865	.850/.773/.810	.810/.944/.872	.700/.583/.636
	act	.650/.722/.684	.444/.364/.400	.833/.556/.667	.300/.500/.375
Qwen3.5-9B	bare	.514/1.000/.679	.900/.409/.563	.480/.667/.558	.000/.000/.000
	policy	.708/.944/.810	.900/.409/.563	.486/.944/.642	1.000/.083/.154
	few-shot	.667/1.000/.800	.944/.773/.850	.789/.833/.811	.667/.333/.444
	act	.429/.167/.240	.352/.864/.500	1.000/.056/.105	.125/.083/.100
Gemma4-E4B	bare	.571/.889/.696	.769/.455/.571	.619/.722/.667	.625/.417/.500
	policy	.750/.833/.789	.810/.773/.791	.640/.889/.744	.750/.250/.375
	few-shot	.630/.944/.756	.778/.636/.700	.833/.833/.833	.571/.333/.421
	act	.171/.389/.237	.107/.136/.120	1.000/.056/.105	.000/.000/.000

Table 10: Full per-action primary metrics on the same 70 adjudicated items. P/R/F1 denotes precision/recall/F1. No scores use the unevaluated complete policy.

Model	Condition	Accuracy [95% CI]	Macro-F1	OM	UM	Ephemeral R	Verify R	Clarify R
Qwen3.5-9B	bare	.472 [.306,.639]	.381	.056	.429	.381	.833	.000
	policy	.444 [.278,.611]	.352	.111	.429	.286	1.000	.000
	few-shot	.472 [.306,.639]	.365	.306	.000	.333	.500	.000
	act	.306 [.167,.472]	.177	.000	1.000	.238	1.000	.000
Gemma4-E4B	bare	.167 [.056,.306]	.071	.000	1.000	.000	1.000	.000
	policy	.306 [.167,.472]	.277	.028	.571	.095	1.000	.000
	few-shot	.472 [.306,.639]	.411	.389	.143	.286	.833	.000
	act	.444 [.278,.611]	.399	.528	.000	.238	.667	.000
Phi-4-mini	bare	.111 [.028,.222]	.071	.111	1.000	.000	.667	.000
	policy	.194 [.083,.333]	.170	.278	.429	.000	.500	.000
	few-shot	.250 [.111,.389]	.176	.500	.000	.000	.333	.000
	act	.194 [.056,.333]	.096	.000	1.000	.048	1.000	.000
Qwen3-4B	bare	.278 [.139,.417]	.205	.306	.286	.000	.833	.000
	policy	.444 [.278,.611]	.367	.083	.286	.286	.833	.000
	few-shot	.361 [.194,.528]	.275	.444	.000	.000	1.000	.000
	act	.500 [.333,.667]	.283	.000	1.000	.571	1.000	.000

Table 11: All original local-model naturalistic conditions ($n = 36$), reported descriptively. Gold counts are 7 persist, 21 ephemeral, 6 verify, and 2 clarify. No Claude results or p-values are reported.

Failure Modes of Imperfect Relational Context in LLM Document Extraction

Boris Kundu
Independent Researcher
boriskundu@gmail.com

Pooja Mehta
Independent Researcher
poojamehta.mba@gmail.com

Abstract

LLM-based extraction pipelines increasingly consume relational context produced upstream, for example by a heuristic linker, a retrieval step, or another model. We study how imperfect relations between document entities affect LLM extraction on FUNSD, a benchmark of scanned forms. Using identical instructions in every condition, four LLMs receive a form's entities with no relations (Flat), relations from a heuristic linker (Heuristic), gold relations (Oracle), or relations that are almost all wrong (Deranged). In a secondary contrast, gold relations raise accuracy for every model (4.0 to 7.9 percentage points, pp). Heuristic and deranged relations lower accuracy by 8 to 25 pp for three models, while GPT-5 shows significantly smaller effects. Comparing the same questions across conditions shows what lies behind the drops. Wrong edges are associated with answers copied from a different field, and missing edges with more blank answers even though the answer text is still present. These patterns appear under two prompt variants, but the prompts were developed on this corpus and every result is a single generation, so we treat the prompt comparison as exploratory.

1 Introduction

LLM-based systems increasingly consume structured context they did not generate themselves, such as retrieved passages, a knowledge graph neighborhood, or, as we study here, a relation graph over a document's own entities produced by an upstream linker. This context is supplied to change what the system produces, but it is rarely perfect, and a pipeline has no principled way to know how much to rely on a relation it did not itself verify. The same issue arises whenever an agent is grounded in a knowledge graph that an upstream component built, since its edges can be wrong or missing.

We study one concrete instance of this interface. Document pipelines often run OCR and layout analysis first and then hand an LLM typed text spans, their positions, and links between them. We supply this kind of input, with gold segmentation, in a form the model reads as text, and vary only the links. We choose it because it lets us control and score which relations the model sees. We do not test alternatives such as the page image or layout-derived HTML or Markdown (Limitations item 3), so our findings concern this interface, not the best way to prompt for extraction. That correct relations help and wrong ones hurt is expected. What the controlled setting adds is how the failures differ by type of relation error and by model.

Most prior work asks a different question. Models with a graph layer over document entities are trained end-to-end and scored on a fixed benchmark (Lee et al., 2022; Gemelli et al., 2022; Zhang et al., 2023; Hui et al., 2025), which tests whether an architecture helps, not whether the output of an existing general-purpose LLM changes when it is handed the same relations as input context. Work on LLMs consuming graph-structured prompts studies input format, serialization, and, in one case, robustness to noisy or incomplete subgraphs (Dai et al., 2025; Sun et al., 2025), but not inside a document-extraction task that fixes entity text, labels, and layout and decomposes behavior by whether a relation is correct, wrong, or absent.

We hold the extraction task, pipeline, and document fixed and vary only the relations in the input. We build four representations of every FUNSD document (Jaume et al., 2019) that carries a scorable relation: flat entities with type labels and layout but no relations, a relation graph built by a layout-proximity heuristic over gold[1] entities, the gold relation graph, and a deranged graph in which

[1]Here and throughout, "gold" denotes ground-truth values supplied as *input* to the model (entity text, type labels, boxes, or relations); it is distinct from the gold *answer* key used only for scoring (Section 3.5), which the model never sees.

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 26–38
November 10, 2026 ©2026 Association for Computational Linguistics

nearly every edge is wrong by construction. A fixed extract-then-verify LLM pipeline answers every question field and then re-reads the document to revise it, for four models from four labs (Claude Sonnet 5, GPT-5, Qwen3.6-27B, Gemini 3.7 Flash), under one prompt that is identical across conditions and checked programmatically. Two preliminary rounds on this corpus used other wordings and each led to a different, partly wording-driven conclusion (Appendix A); we treat only the third, fully condition-invariant round as primary evidence.

Our contributions are: (1) a controlled document-extraction comparison in which entity text, labels, and layout are held fixed while the supplied relational channel varies; (2) a matched question-level decomposition identifying distinct error signatures for correct, wrong, and absent heuristic relations; (3) evidence that the effects of heuristic and deranged relational context differ substantially across the tested models, with GPT-5 significantly less affected than the other three, a model-dependence result and not a universal LLM behavior. An exploratory cross-prompt sensitivity analysis on the preliminary rounds supplements these findings (Section 4.3). Figure 1 and Table 2 illustrate the setup on a small example, and code, prompts, and aggregate analysis files are available at `https://github.com/boriskundu/relational-context-failure-modes`.

2 Related Work

Graph-structured models for form understanding. FormNet (Lee et al., 2022), Doc2Graph (Gemelli et al., 2022), GraphDoc (Zhang et al., 2023), and SEG-Doc (Hui et al., 2025) each add an explicit graph layer, convolution, attention, or a text-histogram graph, over document entities to recover layout structure lost during serialization. RE^2 (Ramu et al., 2024) targets the same FUNSD entity-linking task we use to build our heuristic graph condition. All of these are trained end-to-end on FUNSD or a similar benchmark, so their graph layer is optimized on the structure it consumes. To avoid a learned relation component fitted to our own evaluation documents, we use an untrained layout-proximity heuristic and report its precision, recall, and F1 directly.

LLMs consuming graph-structured prompts. Dai et al. (2025) study how LLMs process knowledge-graph facts under different linearizations, from raw triples to fluent text, finding systematic format preferences at both the output and attention level, and additionally report that larger LLMs are more susceptible to noisy, incomplete subgraphs. Sun et al. (2025) introduce a structured-prompt benchmark showing general-purpose LLMs with the right graph-prompt design can match specialized graph models on general graph-learning tasks. Our setting instead varies only the supplied key-value relations, with entity text, labels, and layout fixed, and decomposes behavior by whether a given heuristic relation is correct, wrong, or absent.

LLM-based, prompt-driven document extraction. Closer to our own setup, Colakoglu et al. (2025) map the design space of applying general-purpose LLMs to layout-rich document extraction directly, varying input representation, prompting, and model choice, and show a well-configured LLM can match specialized extractors. Their design space includes representation as a factor but does not isolate relational structure or its correctness as an independent variable, our narrower focus. Other multi-stage document pipelines, ontology-guided extraction with human-in-the-loop validation (Chhetri et al., 2025) and a five-agent enterprise pipeline evaluated on automation rate (Gosmar and Zenezini, 2026), likewise do not vary relational structure's quality as a controlled variable.

LLM robustness to conflicting or noisy context. Work on how LLMs handle context that conflicts with what they already know or with itself (Xu et al., 2024) shows that models can be disproportionately influenced by misleading context that resembles relevant evidence, including that retrieval-augmented models are disproportionately misled by counterfactual noise resembling correct context (Hong et al., 2024) and that LLMs can favor plausible-looking self-generated context over correct retrieved context (Tan et al., 2024). That work studies free-text conflicting context whose correctness is not independently known; we instead supply a structured relation channel whose ground-truth relation status is known, letting us decompose a model's response by whether the specific relation it received was correct, wrong, or absent, rather than by how the conflicting context was generated. Graph-retrieval-augmented generation (Zhang et al., 2025) studies a related but distinct question, how a system should retrieve and traverse a graph at query time, whereas our structure is supplied directly in context.

Condition	Text	Labels	Boxes	Relations
Flat	✓	gold	gold	
Heuristic graph	✓	gold	gold	heuristic
Deranged graph	✓	gold	gold	wrong
Oracle graph	✓	gold	gold	gold

Table 1: Primary input representations. Entity text and gold labels are fixed across conditions.

3 Method

3.1 Dataset

We use FUNSD (Jaume et al., 2019), 199 real, noisy scanned forms with human-verified entity text, type labels, bounding boxes, and an entity-linking annotation connecting questions to answers. We use the original annotation release, not the later revision (Vu and Nguyen, 2020), whose files required gated cloud-storage access we did not have; FUNSD's known annotation-quality issues are a limitation (see Limitations). Five of the 199 documents carry no scorable question-answer link and contribute nothing under any condition or model; every reported accuracy is a field-level average over the remaining 194, with model- and condition-specific exclusions in Limitations item 1. A meaningful fraction of questions link to two or more answer entities, usually one logical answer wrapped across two lines; we concatenate all linked entities per question, in reading order, into one scored answer, a disclosed simplification.

3.2 Four representation conditions

Every document is rendered with entity text held constant across all conditions and layout (bounding boxes) held constant across flat and every graph condition. Table 1 summarizes the four conditions, and Table 2 shows them on a small constructed form. In every condition the model sees a list of spans, each with its text, a label (question, answer, header, or other), and its box. The conditions differ only in whether a list of links between spans is added, and which links. **Flat** adds none. **Heuristic** adds the links produced by a layout-proximity rule. **Oracle** adds FUNSD's gold links. **Deranged** adds a list of the same size and degree structure as the gold one but with the answers reassigned so that nearly every link points to the wrong answer. A background raw condition (text only) is scored separately against flat (Appendix C).

The heuristic graph's edges come from a layout-proximity rule, matching each answer to the nearest question above or to its left, with tie-breaking on remaining distance. We chose this over an LLM-inferred relation (which would entangle any benefit with the model's own self-consistency) or an existing FUNSD-trained linker (which risks the circularity noted above). We measure the heuristic's own precision, recall, and F1 on two disjoint parts of the full 199-document corpus, a 35-document validation subset (precision 0.53, recall 0.43, F1 0.47, Appendix E gives the raw edge counts) and its 164-document complement (precision 0.50, recall 0.45, F1 0.47), a substantially imperfect relation source.

The deranged graph replaces every edge with a wrong one through a degree-preserving derangement over entity identity, not a naive random shuffle, which can leave some edges correct by chance. A few resulting edges are still identity-correct or point at text-duplicate answers; these are disclosed and excluded from any diagnostic that requires a genuinely wrong, text-distinguishable edge (Appendix E gives exact counts). Five documents, each with only one or two scorable questions of a within-document-unique multi-value degree, have no same-degree partner to swap with and cannot be deranged, so every model's Deranged condition excludes them by design (Limitations item 1 gives document counts).

3.3 Pipeline and models

Each document, condition, and model cell is processed by the same fixed extract-then-verify LLM pipeline (a 2-node LangGraph implementation, not an autonomously planning agent). An *extract* step produces an initial answer for every question field, and a forced *verify* step re-reads the same representation and corrects any answer it finds ungrounded or wrong; both the initial and final answers are retained and scored. We use four models from four labs: Claude Sonnet 5 (Anthropic), GPT-5 (OpenAI), Qwen3.6-27B (open-weight, served via Groq), and Gemini 3.7 Flash (Google), each fixed to a low or default reasoning setting that keeps reasoning enabled (Table 5; Groq exposes no lower enabled setting for Qwen3.6-27B), held constant across all conditions. We chose them for diversity of provider and openness, API availability, and a reasoning mode we could bound in cost and latency, not for matched size or architecture (undisclosed by the three proprietary providers), so differences between models are descriptive and we make no claim about why they arise. A fifth candidate was dropped before any evaluation because its reason-

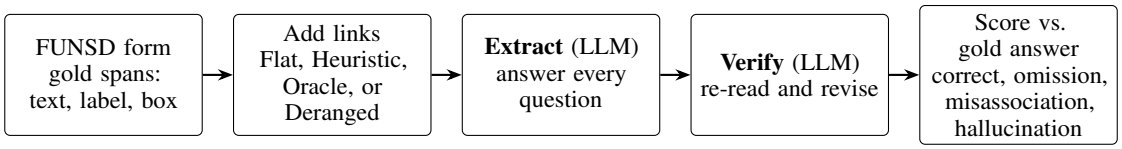


Figure 1: Experimental pipeline. Only the second box varies across conditions. The instruction text, model, and scoring are identical, and both the initial (Extract) and final (Verify) answers are scored.

Condition	Links shown	Resulting bucket for each question
Answer key	(not shown) Date → March 4; Name → J. Smith; Brand → Acme	
Flat	none	no links shown
Heuristic	Date → March 4; Name → Acme	Date: correct edge; Name: wrong edge; Brand: no edge
Oracle	Date → March 4; Name → J. Smith; Brand → Acme	all correct edges
Deranged	Date → J. Smith; Name → Acme; Brand → March 4	all wrong edges

Table 2: A constructed form (not taken from FUNSD) with six spans, three questions (Date, Name, Brand) and their answers (March 4, J. Smith, Acme). In the Heuristic row the three questions fall into the correct-, wrong-, and absent-edge buckets of Section 4.2. Answering "Acme" for Name there is a misassociation (a real value from the Brand field), and answering blank for Brand, although "Acme" is in the input, is a blank response.

ing length could not be bounded.

3.4 Prompt design

Every condition receives one shared instruction block that differs only in the inserted document representation. Prompt identity outside that substitution is checked programmatically, alongside a banned-word list for the input-specific paragraph and a check that the instruction text is byte-identical across the four conditions above plus Raw (Appendix A gives the full safeguard list). The wording was finalized after two preliminary rounds on the same corpus. Round 1 confounded relational structure with an instruction to treat shown links skeptically, and Round 2 removed that instruction but inadvertently also a needed link-explanation sentence, after which two models showed a severe blank-refusal pattern on questions with no shown heuristic edge. Round 3 restores one short, neutral sentence explaining that a link, when present, associates a question with an answer, and that some questions have none; restoring it did not reliably reduce that pattern (Section 4.3). All primary results in Section 4 use Round 3 alone. The two failure signatures of Section 4.2 persist under both condition-invariant regimes, Round 2 and Round 3, which we treat as exploratory sensitivity evidence rather than validation, since both were developed on this same corpus. Appendix A gives the full prompt history and literal templates.

3.5 Scoring

Each predicted answer is scored against the gold answer key with an exact normalized match (lowercased, whitespace collapsed) and a fuzzy similarity score (rapidfuzz `fuzz.ratio`, a normalized edit-based similarity from 0 to 100, with a match at 85 or above). The fuzzy match is tolerant of formatting and OCR-level differences and is our accuracy measure throughout. We fixed the threshold of 85 at the start of development, before any evaluation run, and never tuned it on results; it admits small character-level differences such as punctuation or a one-character OCR variant and typically rejects larger omissions, such as a missing word in a short answer. We checked it by manual review of scores near the threshold (Section 3.6) and a sweep from 80 to 95 (Limitations item 5). A blank answer for a question with a gold answer is an omission (reported as a blank response in Section 4). We classify every other non-matching answer as a misassociation if it fuzzy-matches the text of any entity belonging to a different field, and otherwise as a hallucination, which is therefore a catch-all for answers not supported by any other field, including truncated or paraphrased ones. Questions linked to several answer entities are scored once, against the concatenation of all linked entities in reading order. A partial answer, such as only the first of two wrapped lines, usually falls below the threshold and is classed as a hallucination, not a misassociation, because the question's

own gold entities are excluded from the comparison (Appendix B gives worked examples). For the three graph conditions, we compute a *wrong-edge-consistency* rate, whether the pipeline's answer matches what the shown edge implies, among questions whose shown edge is itself wrong by this same fuzzy-text ruler. The question-level failure-mode buckets of Section 4.2 use the same rule; Appendix E details how this affects the deranged graph's edge-correct bucket.

Because field scores from the same document are correlated, confidence intervals come from a document-level paired bootstrap (20,000 resamples), never treating individual fields as independent. Each model-level comparison is tested once per model, so we report each family under a Bonferroni correction for its own number of tests, a per-comparison confidence level corresponding to 95% family-wise control (98.75% for a four-test family, 99.17% for six), alongside the uncorrected 95% interval.

3.6 Scorer validation

Manual review of two non-overlapping 50-item samples (one stratified across every model and condition, one drawn from scores near the fuzzy-match threshold), a spot check against the authors' own judgment rather than an independently annotated inter-rater study, agreed with the automatic verdict in all 100 cases. A small number of recurring patterns are reported as limitations rather than one-off noise (see Limitations).

4 Results

4.1 Overall effects of supplied relational context

Table 4 reports every primary contrast under our one condition-invariant prompt, each interval already Bonferroni-corrected within its four-test family. Adding the heuristic graph (F1 0.47) to flat significantly *hurts* three of four models, Claude, Qwen, and Gemini; only GPT-5's corrected interval includes zero. The deranged graph shows the same broad split, significantly hurting Claude, Qwen, and Gemini while leaving GPT-5's small negative point estimate non-significant. Gold relations, by contrast, help every model in both comparisons that isolate them: Oracle beats Flat by 4.0 to 7.9pp, and Oracle beats Heuristic by 9.1 to 27.9pp (Appendix E, Table 8), our single largest and most consistent effect, all eight corrected intervals excluding zero. We report Oracle−Flat as a secondary, post-hoc contrast rather than one of our originally locked comparisons, and read both oracle contrasts as an upper bound on faithful relay under complete, correctly labeled information, not evidence of relational reasoning in general.

Condition	Claude	GPT-5	Qwen	Gemini
Flat	75.5	79.3	77.9	79.1
Heuristic graph	59.1	76.8	58.8	61.9
Deranged graph	51.2	78.5	54.0	71.0
Oracle graph	82.0	85.9	86.7	83.1
Flat, on documents matched to:				
Heur., Oracle	75.5	79.3	78.8	79.1
Deranged	75.8	79.3	78.8	79.1
Number of documents:				
Flat	194	194	194	194
Heur., Oracle	194	194	193	194
Deranged	188	189	188	189

Table 3: Field-completion accuracy (%, fuzzy match, final answer) by condition and model, our primary evaluation. Each cell uses every document that model completed under that condition, so the document sets differ slightly (last three rows); Qwen's graph conditions cover 193 documents because one document fails to parse under every graph condition. The paired contrasts in Table 4 use only documents present in both conditions, so each equals the condition's score minus the matched Flat row, up to rounding, not the first Flat row (for Qwen, $58.8-78.8=-20.0$ and $86.7-78.8=+7.9$). Details are in Limitations item 1.

A direct interaction contrast (GPT-5's own Heuristic−Flat and Deranged−Flat delta bootstrapped against each other model's same delta on one shared resampled document set, Appendix E, Table 9) provides evidence beyond the absence of significance against zero: GPT-5 shows significantly smaller condition effects than every other model in this evaluation, in all six pairwise tests after a Bonferroni correction across the family (99.17% per-comparison CI), though the smallest (vs. Gemini on Deranged−Flat) only marginally survives.

4.2 Failure-mode decomposition

We split every scorable question into three buckets by comparing the answer implied by the heuristic graph's shown edge or edges, if any, with the gold answer under the same fuzzy rule used for scoring. For a question linked to several answer entities, the implied answer is their concatenation in reading order, so an edge set that recovers only part of the answer counts as wrong. The buckets are correct

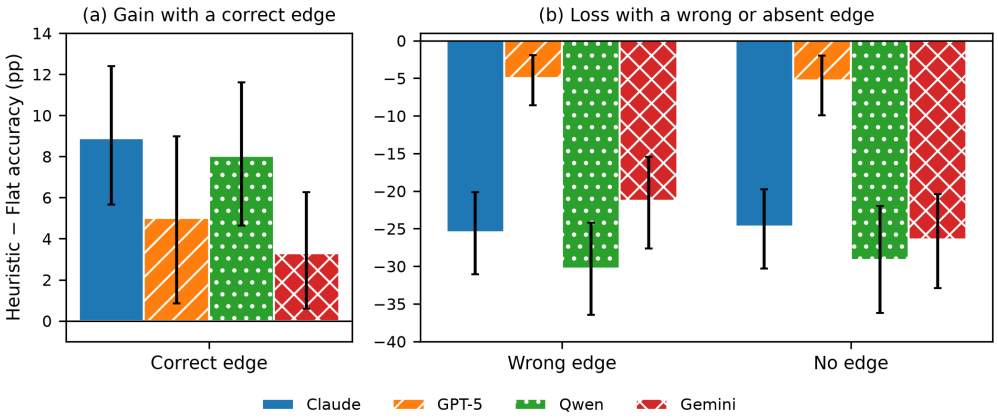


Figure 2: Heuristic−Flat accuracy difference, in percentage points, for the same questions under the two conditions, split by the heuristic relation shown. Panel (a) shows the gain where the edge is correct, and panel (b) the loss where it is wrong or absent (note the different y-axis ranges). Error bars are uncorrected 95% document-bootstrap CIs; Bonferroni-corrected significance is in the text.

Contrast	Model	Δ (pp)	Bonf. CI
Heuristic−Flat	Claude	−16.3	[−21.7, −11.4]
	GPT-5	−2.5	[−7.5, 0.7]†
	Qwen	−20.0	[−25.6, −14.2]
	Gemini	−17.1	[−22.5, −12.2]
Oracle−Flat (secondary)	Claude	+6.5	[4.4, 8.6]
	GPT-5	+6.6	[4.7, 8.5]
	Qwen	+7.9	[5.2, 11.0]
	Gemini	+4.0	[2.6, 5.6]
Deranged−Flat	Claude	−24.6	[−32.7, −17.2]
	GPT-5	−0.8	[−5.7, 2.1]†
	Qwen	−24.7	[−33.3, −16.4]
	Gemini	−8.1	[−13.3, −3.7]

Table 4: Field-completion accuracy effects and paired condition contrasts vs. Flat, each interval already Bonferroni-corrected within its four-test family (98.75% per-comparison CI). † marks the only two intervals that include zero (both GPT-5). Document counts are in Limitations item 1; Oracle−Heuristic is in Appendix E, Table 8.

(25.6–25.7% of scorable questions), wrong (32.4–32.5%), or absent (no edge shown, 41.8–42.0%, the largest bucket). Figure 2 reports the Heuristic−Flat delta within each, per model, with a document-bootstrap 95% CI. Questions with a correct heuristic edge score 3.3 (Gemini) to 8.9pp (Claude) higher under Heuristic than their matched Flat counterparts, significant at the uncorrected 95% level for all four models but surviving a Bonferroni correction across the twelve model-by-bucket tests in this family only for Claude and Qwen, not GPT-5 or Gemini. All eight wrong- and absent-edge deltas survive the same correction. Wrong- and absent-edge questions show different dominant error signatures. Wrong-edge questions show elevated cross-field misassociation, a real value copied from a different field, in 25.9% (Claude), 20.4% (Qwen), 19.4% (Gemini), and 6.3% (GPT-5) of all such questions. No-edge questions instead show elevated blank-response rates, 36.2% (Claude), 44.0% (Qwen), 40.8% (Gemini), 19.1% (GPT-5), even though the same answer-bearing entity text used by Flat remains directly present in the same representation. These percentages are shares of every question in the bucket, not just the incorrect ones. Relative to the same questions under Flat, Round 3 increases no-edge blank responses by 8.2–31.9pp and wrong-edge misassociation by 3.4–21.4pp (Appendix E, Table 11), with all eight matched penalties individually Bonferroni-significant. GPT-5's error rates stay well below the other three models in both failure buckets, consistent with its comparatively flat accuracy across every condition (Table 3); this decomposition is diagnostic rather than a controlled per-edge intervention (Limitations item 4). We exclude the deranged graph's own edge-correct bucket from analysis: 92% of it (84 of 91 cases) reflects gold-text coincidence rather than a genuine identity match (Section 3.2).

4.3 Exploratory: prompt-regime sensitivity

Our strongest sensitivity evidence compares Round 2 and Round 3, condition-invariant prompts that differ only in whether the neutral link-explanation sentence (Section 3.4) is present. The matched no-edge blank penalty and wrong-edge misassociation delta of Section 4.2 (Appendix E, Table 11)

remain positive for every model under both rounds; adding the explanation does not consistently reduce them, and for Claude both grow substantially. Round 1 was not condition-invariant and differs from Round 3 along more than one wording dimension, so we report it only as provenance, not controlled evidence: wrong-edge reliance is far lower under Round 1 for every model (Appendix D gives the figure and Table 10 the bootstrapped deltas). Because every regime was developed on this same corpus, we treat all of this as exploratory, not a causal test: the two failure signatures are present under both condition-invariant regimes examined, not shown to be prompt-invariant in general.

5 Discussion

Adding a relational channel changes what its silence means. For several models, an unlinked question is treated as unsupported even though the same entity text suffices under Flat, so absence within the channel can acquire epistemic weight it would not otherwise carry, while a wrong edge redirects the answer toward a different field's value and correct relations, unsurprisingly, still help. The elevated blank rate suggests that several models treat a missing relation as evidence against answer availability, rather than falling back to the entity text itself; a system consuming externally supplied relational context should therefore distinguish "no relation was predicted" from "no answer exists."

Susceptibility to all of this is strongly model-dependent rather than a universal LLM behavior. GPT-5 shows little heuristic or deranged degradation, the lowest wrong-edge-consistency rate by a wide margin, and error rates well below the other three models in both failure buckets, while Claude, Qwen, and Gemini are all substantially affected. We report this heterogeneity as a result in its own right rather than speculate about its cause.

This also bears on anyone building a pipeline on an automatic relation extractor. A linker with F1 around 0.47 is not automatically safe to add, and its downstream effect also appears sensitive to prompt regime in our exploratory comparison. A single run cannot distinguish a model property from an artifact of one prompt's wording, which is why Appendix A gives the full prompt history.

6 Conclusion

Imperfect relational context can alter downstream LLM extraction in ways that aggregate accuracy obscures. In our FUNSD evaluation, wrong and missing heuristic relations are associated with distinct error signatures whose magnitudes vary across models and remain visible under both condition-invariant prompt regimes examined. Future work should manipulate relation correctness and coverage independently, repeat generations, and test additional datasets.

Limitations

1. **Document counts.** Table 3 gives the documents behind each cell: 194 for Flat, Heuristic and Oracle (193 for Qwen3.6-27B, one persistent parse failure under every graph condition), and 188 or 189 for Deranged, which also excludes five documents that cannot be deranged (Section 3.2) and, for Claude, one further parse failure. Paired contrasts in Table 4 use the intersection of the two conditions' documents, so each equals the difference of the matched Flat rows, up to rounding. Scoring every persistent-parse-failure cell as entirely wrong instead of excluding it shifts every affected Table 4 headline by at most 0.5pp (Qwen3.6-27B's Oracle−Flat, whose excluded document has 41 scorable questions against a corpus average of 14) and every other by at most 0.1pp.

2. **Adaptive prompt development, single generation, and provider versions.** The final prompt (Section 3.4) was informed by this corpus's own results from two preliminary rounds, so the primary round is a frozen-before-execution evaluation, not a hypothesis specified before any data existed. Every cell in every round is one generation, so the document-level bootstrap captures variability across documents, not across repeated generations, and we treat low-single-digit effects cautiously even when their interval excludes zero (GPT-5's heuristic and deranged estimates did not survive even the uncorrected test). None of the four model identifiers is a dated, pinned snapshot (Appendix A), which affects only the exploratory cross-round comparison (Section 4.3). The three prompt regimes also differ in more than one wording dimension, so that comparison is exploratory, not causal.

3. **Gold upstream information, not an end-to-end pipeline.** Every condition receives

FUNSD's gold entity boundaries, type labels, and bounding boxes, which isolates relational structure as the one manipulated variable but means none of our numbers speak to a pipeline that also solves segmentation, OCR, or typing. The heuristic graph is a layout-proximity rule over those gold entities, not an end-to-end relation extractor, and a stronger or weaker linker could shift its result either direction. We also do not compare against feeding the page image directly or against HTML or Markdown produced by layout analysis, so we cannot say whether these failure modes would appear, shrink, or grow under those inputs.

4. **Detectability and plausibility are diagnosed, not independently manipulated.** Heuristic and deranged conditions differ in spatial plausibility, semantic compatibility, and structure at once (equivalently, coverage and correctness: ≈58% coverage at F1 ≈0.47 versus 100% coverage at near-zero correctness), so their contrast does not isolate any one factor. A degree-preserving derangement keeps each entity's participation count, multi-value cardinality, and graph topology, so it is not information-free noise; the few identity-correct and text-duplicate edges it can produce are disclosed and excluded from any wrong-edge-only diagnostic (Section 3.2). Relation-status buckets (Section 4.2) are diagnostic, since status is observed within the heuristic representation and not manipulated per question.

5. **Additional design limitations.** We ran no extract-then-re-extract condition separating a second call's own effect from a second read over relational structure, so our correction-rate pattern is descriptive, not evidence that the relational representation drives verification behavior. We merged our original development/test split into one set partway through, so every document was eligible for both development and the final numbers, weaker than an untouched test set; the heuristic has no learned parameters, though its design was developed within the same study. Hand validation also showed the scorer cannot distinguish a correct but reordered multi-value answer from a wrong one; an order-insensitive rescoring and a fuzzy-threshold sweep (80–95) each change Table 4's Heuristic−Flat and Deranged−Flat deltas, the two contrasts we checked this way, by at most 1.6pp.

6. **Scope of generalization.** All results are on one English-language, form-specific benchmark, FUNSD, with one serialization of each representation, one model roster, and one pipeline architecture; a different choice along any of these axes could change the magnitude or direction of any effect reported here. We use the original FUNSD annotations, whose known block-level limitations are addressed by EC-FUNSD (Zhang et al., 2026), so replication on the corrected annotation set is a natural next step.

Ethical Considerations

Data provenance and use. FUNSD's source documents come from the Truth Tobacco Industry Documents archive, made public through litigation disclosure decades ago, and some forms carry real names, signatures, and dates from that era. We collected no new data. FUNSD is released under a noncommercial research and education license, and we send only document text, never images, to four commercial LLM APIs, for noncommercial academic research. **Human involvement.** There was no new human-subject data collection or crowdsourced annotation; involvement was limited to the authors' review of a sample of scored outputs (Section 3.6). **Reproducibility.** Code, the Appendix A.2 prompt templates, and the aggregate analysis files behind every table are released at the repository linked in the Introduction under an open-source license. FUNSD is not redistributed apart from one small sample annotation used as a test fixture (a script downloads the dataset from its official source), and per-question model outputs are withheld because they contain FUNSD text, so reproducing raw runs requires API access to the four models.

Declaration on Generative AI

Portions of this manuscript were drafted and revised with writing assistance from Claude (Anthropic), which was also used for coding assistance during the experimental pipeline's implementation. All experimental design, analysis, results, and conclusions are the authors' own, reviewed and verified before submission.

References

Tek Raj Chhetri, Yibei Chen, Puja Trivedi, Dorota Jarecka, Saif Haobsh, Patrick Ray, Lydia Ng, and Satrajit S. Ghosh. 2025. STRUCTSENSE: A task-agnostic agentic framework for structured information extraction with human-in-the-loop evaluation and benchmarking. *arXiv preprint arXiv:2507.03674*.

Gaye Colakoglu, Gürkan Solmaz, and Jonathan Fürst. 2025. Problem solved? information extraction design space for layout-rich documents using LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*.

Xinbang Dai, Yuncheng Hua, Tongtong Wu, Yang Sheng, Qiu Ji, and Guilin Qi. 2025. Large language models can better understand knowledge graphs than we thought. *Knowledge-Based Systems*, 312(3):113060.

Andrea Gemelli, Sanket Biswas, Enrico Civitelli, Josep Lladós, and Simone Marinai. 2022. Doc2Graph: A task agnostic document understanding framework based on graph neural networks. In *ECCV Workshops*.

Diego Gosmar and Giovanni Zenezini. 2026. MADP: A multi-agent pipeline for sustainable document processing with human-in-the-loop. *arXiv preprint arXiv:2605.17159*.

Giwon Hong, Jeonghwan Kim, Junmo Kang, Sung-Hyon Myaeng, and Joyce Jiyoung Whang. 2024. Why so gullible? enhancing the robustness of retrieval-augmented models against counterfactual noise. In *Findings of the Association for Computational Linguistics: NAACL 2024*.

Yangyang Hui, Jiahao Liu, Qi Zhang, Tianyi Zhou, and Yonghong Song. 2025. SEG-Doc: A simple yet efficient graph neural network framework for document key information extraction. *Neurocomputing*, 647.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. FUNSD: A dataset for form understanding in noisy scanned documents. In *ICDAR Workshop on Open Services and Tools for Document Analysis (ICDAR-OST)*.

Chen-Yu Lee, Chun-Liang Li, Timothy Dozat, Vincent Perot, Guolong Su, Nan Hua, Joshua Ainslie, Renshen Wang, Yasuhisa Fujii, and Tomas Pfister. 2022. FormNet: Structural encoding beyond sequential modeling in form document information extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Pritika Ramu, Sijia Wang, Lalla Mouatadid, Joy Rimchala, and Lifu Huang. 2024. RE^2: Region-aware relation extraction from visually rich documents. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. 2025. GraphICL: Unlocking graph learning potential in LLMs through structured prompt design. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2440–2459.

Hexiang Tan, Fei Sun, Wanli Yang, Yuanzhuo Wang, Qi Cao, and Xueqi Cheng. 2024. Blinded by generated contexts: How language models merge generated and retrieved contexts when knowledge conflicts? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hieu M. Vu and Diep Thi-Ngoc Nguyen. 2020. Revising FUNSD dataset for key-value detection in document images. *arXiv preprint arXiv:2010.05322*.

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)*.

Chong Zhang, Yixi Zhao, Yulu Xie, Chenshu Yuan, Yi Tu, Ya Guo, Mingxu Chai, Ziyu Shen, Yue Zhang, and Qi Zhang. 2026. Unveiling the deficiencies of pre-trained text-and-layout models in real-world visually-rich document information extraction. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 1–16, Rabat, Morocco. Association for Computational Linguistics.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.

Zhenrong Zhang, Jiefeng Ma, Jun Du, Licheng Wang, and Jianshu Zhang. 2023. Multimodal pre-training based on graph attention network for document understanding. *IEEE Transactions on Multimedia*, 25:6743–6755.

A Full prompt provenance

We ran this ablation three times on the same 194-document corpus. **Round 1**'s graph prompt explained what a link means, instructed treating it as a primary guide, and warned that shown links may be wrong and should not be followed blindly. This text was absent from the flat and raw prompts, so it confounded structure with an explicit skepticism instruction. **Round 2** removed that instruction but, a later diff showed, also removed the generic link explanation along with it, so Round 1 and Round 2 differ along more than one dimension. Under Round 2's condition-blind but under-explained prompt, Qwen3.6-27B and Gemini 3.7 Flash showed a severe blank-refusal pattern specifically on questions whose heuristic graph offered no

edge, consistent with treating an unexplained absence of a links field as license to answer null rather than falling back to the flat entity text present in the same representation. Restoring the explanation in Round 3 did not reliably reduce this pattern (Section 4.3). **Round 3** (reported as primary throughout the main text) restores one short, neutral sentence, identical across all conditions, explaining that the representation may include structural information such as span labels, positions, or links, that a link when present associates a question with an answer, and that some questions have none. Its generic first half is not data-informed; the clause about questions with no link directly responds to Round 2's failure on this same corpus, a distinction we disclose rather than calling the whole sentence either fully generic or fully post hoc. A banned-word list for the input-specific paragraph (correct, incorrect, reliable, unreliable, candidate, trust, distrust, primary guide, plausible, blindly, fall back, and inflections), a check against gating any instruction on link presence, and a regression test that the assembled extract and verify prompts are byte-identical across all five conditions once the document representation is removed all carry over from Round 2. The word list does not apply to the verify prompt's own instructions, which legitimately use words such as "correct". A manifest audit (`git diff` across every module and the dataset file between the two later rounds' launch commits) confirms the prompt files are the only source difference. We describe our analysis plan as frozen before Round 3's execution and informed by Rounds 1 and 2 on this same corpus, not pre-registered in the sense of specified before any data existed.

Scope split. The raw-versus-flat background comparison (Appendix C) is reported from Round 2 only, the first condition-invariant version of that comparison; Raw shares no structural field with the link-definition sentence, so it is unaffected by Round 2's no-edge problem, but it was not rerun under Round 3's exact wording. Flat's own accuracy is stable within 0.6pp for every model between Round 2 and Round 3, which we use as a measured proxy for Raw's likely stability, not a direct measurement of it. All of Section 4 and Section 4.2 come from Round 3 alone; no table merges Round 2 and Round 3 columns as directly comparable.

Model versions across rounds. All three rounds used the same model identifiers (Section 3.3) across roughly a four-day window; a manifest audit confirms our own code, data, and requested identifiers were unchanged. None of the four identifiers is a dated, pinned snapshot, so we cannot independently verify a provider's served implementation was itself unchanged across our calls.

A.1 Model and API reproducibility

Table 5 lists the exact model identifiers, reasoning settings, and call parameters.

A.2 Literal prompt templates

Every extract prompt shares this fixed scaffold across all three rounds and all five conditions, substituting only the emphasized input-specific paragraph below and the document itself:

> "You are a document-processing assistant completing a structured form-filling task.
>
> *[input-specific paragraph, below]*
>
> Instructions: Answer using ONLY information present in the document below. Do not infer, guess, or fabricate any value that is not explicitly present. If you cannot find an answer for a question in the document, output an empty string `""` for that answer, do not leave it out of the list entirely. Respond with ONLY a JSON array, no other text, in this exact format: `[{"question": "<question text as it appears>", "answer": "<answer text, or empty string>"}, ...]`
>
> Document: [the rendered representation]"

The verify step re-reads the same input-specific paragraph alongside the model's own previous answer, under the same JSON-array output format. Its full template, identical across rounds and conditions apart from the same input-specific paragraph, is:

> "You are re-checking your own previous answer to a document form-filling task, to catch and correct any mistakes before finalizing.
>
> *[input-specific paragraph]*
>
> Your previous answer was: *[the model's JSON answer from the extract step]*
>
> Re-read the document below and your previous answer above. For each question, check whether the answer is actually supported by the document, correct any answer that is wrong, ungrounded, or inconsistent, and leave correct answers unchanged.
>
> Instructions: Base corrections ONLY on information present in the document below. Do not infer, guess, or fabricate. If an answer should be blank because the document doesn't contain it, output an empty string `""`. Respond with ONLY a JSON array, no other text, in this exact format: `[{"question": "<question text as it appears>", "answer": "<answer text, or empty string>"}, ...]`
>
> Document: [the rendered representation]"

Model	Provider	Requested ID	Reasoning setting
Claude Sonnet 5	Anthropic	`claude-sonnet-5`	`thinking`: adaptive, `effort`: low
GPT-5	OpenAI	`gpt-5`	`reasoning_effort`: low
Qwen3.6-27B	Groq	`qwen/qwen3.6-27b`	`reasoning_effort`: default
Gemini 3.7 Flash	Google	`gemini-3.7-flash`	`thinking_level`: low

Table 5: Exact requested model identifiers and reasoning settings, held constant across all conditions and rounds. Temperature: Claude and GPT-5 reject any explicit value (omitted); Qwen and Gemini use 0.0. Max output tokens: Claude 128,000 (provider ceiling, required explicitly by the API); GPT-5 and Gemini use the provider default (uncapped); Qwen is capped at 16,384, Groq's own hard per-call maximum for this model. All Round 3 (primary) calls were made on 2026-09-03.

Round 1 input-specific paragraph, flat and raw conditions (Round 1 only; Rounds 2 and 3 use one paragraph for all conditions, below). Flat: "Below is a JSON list of labeled text spans extracted from a scanned form. Each span has a 'text' value and a 'label' of 'question' (a field name), 'answer' (a field value), 'header' (a section title), or 'other'. The spans are NOT linked to each other, you must match each 'question' span to the 'answer' span you judge most likely to be its value, based on the text content, position (each span includes its 'box' pixel coordinates), and typical form layout conventions." Raw: "Below is the plain text content of a scanned form, listed in reading order (top-to-bottom, left-to-right). It has no field labels or structure markup, you must recognize which parts of the text are field labels ('questions') and which are their corresponding values ('answers') yourself, based on the wording and layout implied by the reading order."

Round 1 input-specific paragraph, graph conditions. (Rounds 1 and 2 labeled the Heuristic and Deranged conditions "Predicted" and "Shuffled".) "Below is a JSON structure of labeled text spans extracted from a scanned form, plus a 'links' list of [id, id] pairs indicating which spans the document's structure connects (typically a question span to the answer span that follows it). Each span has a 'text' value and a 'label' of 'question', 'answer', 'header', or 'other'. Use the given links as your primary guide to match each 'question' span to its answer, but note that this linking structure was produced automatically and may contain errors, if a linked answer's text doesn't plausibly correspond to its question, use your own judgment about the document's content instead of following the link blindly."

Round 2 input-specific paragraph (all five conditions, condition-blind but under-explained; Appendix A discusses its no-edge failure). "Below is a machine-readable representation of one scanned form. Use it to identify each field name (a 'question') and the value that answers it (an 'answer')."

Round 3 input-specific paragraph (all five conditions, primary throughout the main text). "Below is a machine-readable representation of one scanned form. The representation may include structural information such as span labels, positions, or links between spans, depending on how the document is represented. A link, when present, associates a question span with an answer span. Some question spans do not have an associated link. For each field name (a 'question'), determine the value that answers it (an 'answer') using all information available in the representation below."

B Scoring examples

Table 6 shows how the scorer of Section 3.5 treats typical answers, using the constructed form of Table 2 extended with a two-line answer. Two examples need a note. A multi-entity answer is scored as one string, so giving only the first line is incorrect and, since that line belongs to the question's own gold answer, is not counted as a misassociation. A misassociation requires that the answer match an entity of a *different* field, so copying "Acme" for Name is a misassociation but inventing "Acme Corp." for Name is a hallucination.

C Background: Raw Text versus Flat Structuring

Moving from raw text to flat, typed, laid-out entities produces a large gap under Round 2's condition-invariant prompt, 14.8 (Gemini) to 23.0 (Claude) points, all Bonferroni-significant across the four models. This is a representation comparison, changing segmentation, labels, layout, and serialization together, not an isolated test of "semantic structure" alone.

Question	Gold answer	Model answer	Score
Name	J. Smith	j. smith	correct (exact after normalization)
Name	J. Smith	J Smith	correct (fuzzy, ratio ≥ 85)
Name	J. Smith	(blank)	omission
Name	J. Smith	Acme	misassociation (matches Brand's answer)
Name	J. Smith	Acme Corp.	hallucination (matches no field)
Address	12 Main St. Ohio	12 Main St.	incorrect, hallucination (partial; own entity excluded)
Address	12 Main St. Ohio	12 Main St. Ohio	correct (both lines)

Table 6: Scoring examples on constructed answers. Accuracy counts only correct rows.

Condition	Claude	GPT-5	Qwen	Gemini
Raw	53.1	57.5	55.8	64.1
Flat	76.1	79.7	77.9	79.0

Table 7: Round 2 background comparison, not part of the primary Round 3 evaluation (Appendix A).

D Exploratory prompt-regime sensitivity, full figure

E Additional statistical analyses

E.1 Dataset and representation details

The heuristic's 35-document validation-subset precision and recall (0.53, 0.43) come from 305 true positives of 576 predicted and 716 gold edges. The degree-preserving derangement can occasionally map an edge back to a technically correct target (7 of 2,609 edges) or to a different entity whose text happens to duplicate the gold answer's text, most often a repeated date or form number (84 of 2,609); both are disclosed and excluded from any diagnostic that requires a genuinely wrong, text-distinguishable edge (Section 3.2).

Model	Δ (pp)	Bonf. CI
Claude	$+22.8$	$[17.9, 28.1]$
GPT-5	$+9.1$	$[5.9, 13.5]$
Qwen	$+27.9$	$[22.6, 33.2]$
Gemini	$+21.1$	$[15.9, 26.8]$

Table 8: Oracle−Heuristic, the same intersection-of-scorable-documents contrast as Table 4, four-test family (98.75% per-comparison CI). All four tests exclude zero.

Contrast	vs.	Δ (pp)	Bonf. CI
Heuristic−Flat	Claude	$+13.8$	$[5.6, 20.7]$
	Qwen	$+17.6$	$[11.4, 23.7]$
	Gemini	$+14.6$	$[7.1, 21.4]$
Deranged−Flat	Claude	$+23.8$	$[13.6, 33.6]$
	Qwen	$+24.0$	$[13.8, 34.0]$
	Gemini	$+7.3$	$[0.1, 13.9]$

Table 9: GPT-5 interaction contrast: GPT-5's own (contrast−Flat) delta minus each other model's same delta, six-test family (99.17% per-comparison CI). Positive values mean GPT-5 is less affected than the comparison model.

Condition	Model	R1	R3	Δ (Bonf. CI)
Deranged	Claude	0.8%	28.2%	$+27.4$ $[19.7, 35.8]$
	GPT-5	0.1%	1.2%	$+1.1$ $[0.2, 2.3]$
	Qwen	0.8%	31.1%	$+30.3$ $[21.5, 39.0]$
	Gemini	0.1%	11.0%	$+11.0$ $[6.2, 16.3]$
Heuristic	Claude	4.8%	25.7%	$+21.0$ $[14.8, 27.8]$
	GPT-5	3.2%	7.0%	$+3.9$ $[1.6, 6.6]$
	Qwen	5.0%	30.8%	$+25.8$ $[18.5, 33.3]$
	Gemini	2.0%	21.6%	$+19.6$ $[12.9, 27.0]$

Table 10: Wrong-edge-consistency rate, Round 1 (R1) vs. Round 3 (R3), restricted to documents scored in both rounds, four-model family (Bonf. CI). Every model's increase excludes zero.

Metric	Model	Round 2	Round 3	Δ R3−R2 (Bonf. CI)
No-edge blank penalty	Claude	$+4.6$	$+24.0$	$+19.5$ $[13.2, 26.5]$
	GPT-5	$+4.8$	$+8.2$	$+3.4$ $[-2.2, 10.3]$
	Qwen	$+41.3$	$+31.9$	-8.7 $[-17.6, 0.0]$
	Gemini	$+35.2$	$+30.1$	-5.1 $[-15.2, 4.3]$
Misassociation delta	Claude	$+8.6$	$+21.4$	$+12.8$ $[7.5, 18.3]$
	GPT-5	$+2.2$	$+3.4$	$+1.2$ $[-1.1, 3.8]$
	Qwen	$+23.9$	$+17.9$	-5.9 $[-12.6, 0.4]$
	Gemini	$+15.4$	$+16.2$	$+0.8$ $[-5.3, 7.0]$

Table 11: No-edge blank penalty and edge-wrong misassociation delta (both, Heuristic minus Flat, matched on the relevant bucket's questions), Round 2 vs. Round 3, and their bootstrapped Round3−Round2 change (four-model family, Bonf. CI). Round 2 and Round 3 values are each individually Bonferroni-significant against zero for every model (CIs not shown here).

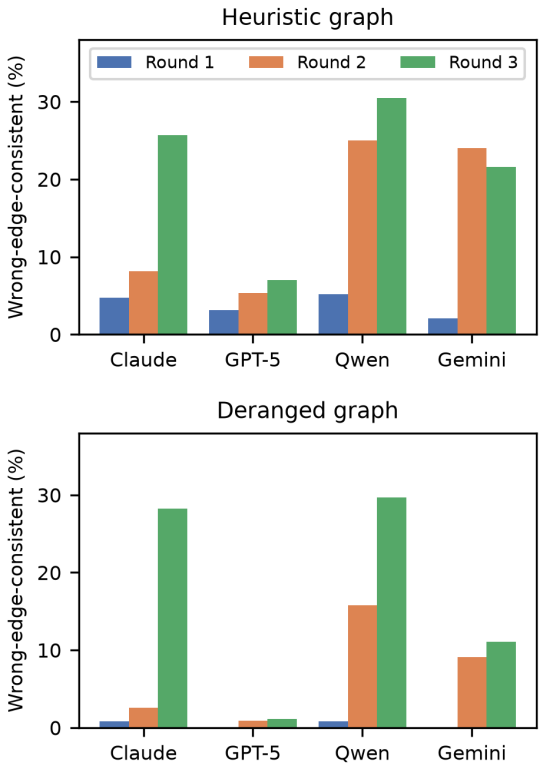


Figure 3: Wrong-edge-consistency rate (among applicable wrong-edge questions) across Rounds 1, 2, and 3, by model, for the heuristic and deranged graph conditions separately. Grouped bars, not a continuous line, since the three prompt regimes are qualitatively distinct rewrites, not points on a single dose axis. See Section 4.3 for discussion.

Asterism: A Hebbian-Weighted Knowledge Graph as a Priority Index over Long-Term LLM Memory

Bidit Das
Independent Researcher, Plano, Texas, USA
orcid.org/0009-0009-3810-2946
github.com/biditdas18/asterism

Abstract

Large language model (LLM) assistants increasingly persist user information across sessions as unstructured "memory" summaries. Such summaries behave like an unindexed heap: every stored fact carries equal retrieval priority, and nothing signals which facts currently matter most. We present Asterism, a local-first system that builds a weighted, decaying knowledge graph from conversation history and injects a ranked subset into the model's context at inference time. The central design claim is architectural: the graph functions as a priority index over flat memory, in the same sense that a B-tree indexes an ordered file, with node weight acting as a salience signal analogous to an index key. We evaluate across five retrieval conditions, two independently authored personal graphs, 100 priority queries per graph, and five assistant models spanning four vendors, with decoding held fixed and the graph reseeded before every call. First, the deployed system — a weighted graph injected with a prioritization instruction — increases *decisiveness under ambiguity* over a flat list of the same facts: scored with a pre-registered, held-out-validated commit/hedge metric, every model commits to a weight-justified answer more often under the graph, in nearly every run. Second, that decisiveness is *more accurate* than flat memory in every model, but absolute accuracy remains low — the best model identifies the true top priority only about 40% of the time and most far less — so the graph delivers a strong relative lift on a problem far from solved. Third, we decompose the effect with two further conditions, the graph with its instruction stripped and a flat list with that instruction added. Decisiveness does not separate them: the instruction alone raises commitment in all ten model–graph cells (mean $+11.2$ points), by as much as the structure alone (mean $+10.7$), so a commit-rate gap is not by itself evidence of a structural effect. Accuracy does separate them. Structure alone improves correctness over a flat list in all nine measured cells, seven separably from zero, against five of ten for the instruction alone, and among committed answers it is more precise than the instruction-only condition in all nine measured cells ($+5.1$ to $+31.4$ points, four separable). The components are complementary rather than redundant: the deployed system's correctness gain matches or exceeds the better single component in all ten cells and strictly exceeds it in nine. A twelve-query pilot had read this decomposition as purely structural; at one hundred queries the supportable reading is that the instruction buys decisiveness while the weighted structure makes that decisiveness accurate. We report the instrument's own reliability (scorer–human agreement $\kappa = 0.64$, with errors that bias the measured gap toward the null), and two components an automatically-populated index requires: an entity-resolution layer whose accuracy ceiling we measure, and a supersession mechanism that demotes contradicted facts. Retrieval latency scales approximately linearly, near 34 ms at 10,000 nodes. We release the full system, both benchmark graphs, and all negative results.

1 Introduction

Conversational LLM systems now routinely retain information about a user between sessions. The dominant form this retention takes is a running natural-language summary: a list of facts, preferences, and history that is prepended to the model's context on later turns. This format is simple and effective for recall, but it is structurally flat. Every fact sits at the same level. There is no representation of which facts are central to the user and which are incidental, no representation of how facts relate, and no representation of which facts have been superseded by newer ones.

In database terms, a flat memory summary is an *unindexed heap*: a store you can scan but cannot query by priority. The contribution of this paper is to take that analogy literally and build the miss-

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 39–52
November 10, 2026 ©2026 Association for Computational Linguistics

ing index. Asterism constructs a weighted directed graph from a user's conversation history, where nodes are concepts and entities, edges are relations, and edge and node weights encode how frequently and recently the user has engaged with each. At inference time, the highest-weight region of the graph is serialized into the model's context. The weight ordering is the index: it tells the retrieval step which facts to surface first, exactly as an index key tells a query planner where to look before touching the underlying data.

This framing makes a falsifiable prediction. If the graph acts as a priority index and not merely as a memory store, its benefit over an unstructured list of the same facts should appear specifically on tasks requiring prioritization and largely vanish on tasks requiring only recall. We test the prioritization half directly; the recall half is not tested (Limitations). The deployed graph condition does two things at once: it imposes a weighted structure *and* instructs the model to prioritize high-weight nodes. A naive graph-versus-flat comparison cannot separate these, so we run five conditions against the same underlying facts, two of which add the instruction without the structure and impose the structure without the instruction (Section 4). We hold decoding fixed, reseed the graph before every call so no response conditions on state a prior response wrote, and repeat the entire design across two independently authored personal graphs, one hundred priority queries per graph, and five assistant models spanning four vendors, five runs each.

We report three findings, each at its actual scope. First, on priority queries the weighted-graph-plus-instruction condition produces more decisive commitment than a flat list of the same facts in every model we test, over two graphs and one hundred queries each, in nearly every run. Second, that decisiveness is *more accurate* than flat memory in every model, but we state absolute levels plainly rather than only the relative lift: even the strongest model is correct only about 40% of the time and most well below that, so the graph improves priority routing without solving it. Third, we decompose the benefit into the graph *structure* and the prioritization *instruction*, correcting a reading a smaller pilot had encouraged. On decisiveness the two are indistinguishable, so a commit-rate gap cannot attribute the effect to structure; accuracy distinguishes them, with structure alone more often right than instruction alone in every measured cell, and the two act complementarily rather than redundantly. We prefer this decomposition, with its low absolute ceiling and underpowered cells stated openly, to the cleaner-sounding structural claim the full data does not support.

An index over a live, automatically-populated store raises two further engineering problems, which we treat as first-class. Entity resolution: without it the graph fragments into near-duplicate nodes that corrupt the weight signal. Supersession: a fact a later turn contradicts must be demoted, so a stale edge does not compete with the new one for retrieval. Sections 5 and 6 take up each in turn.

We make four contributions. We frame a weighted personal knowledge graph as a priority index over flat LLM memory, with node weight as the salience key (Section 3). We run a five-condition evaluation over two independently authored graphs, one hundred priority queries each, and five models from four vendors, which separates the deployed system's benefit from the value of memory per se and from the prioritization instruction (Section 4). We measure where embedding-based canonicalization of short concept labels stops working, and report that negative result (Section 5). We give a supersession mechanism that demotes contradicted facts from retrieval, with a latency study showing approximately linear scaling to 10,000 nodes (Sections 6, 7). All code, benchmarks, and the seeded evaluation graph are publicly available.

2 System Overview

Asterism is a local-first Python system. The entire graph lives in a single SQLite file on the user's machine; no data leaves the device except the user's own prompt and the injected context they choose to send to the model. Figure 1 shows the pipeline.

Extraction. Each conversational exchange is processed by a small extraction model that emits (source, relation, target) triples. Sources and targets become nodes; relations become labeled, weighted edges.

Weighting and decay. Every node and edge carries a scalar weight. Repeated engagement increases weight; disuse decreases it. Following the Hebbian principle that co-activated connections strengthen [1], an edge is reinforced whenever the model traverses it while answering a query. The complementary process is decay: an edge or node that accumulates session exposure without being traversed loses weight and is eventually pruned.

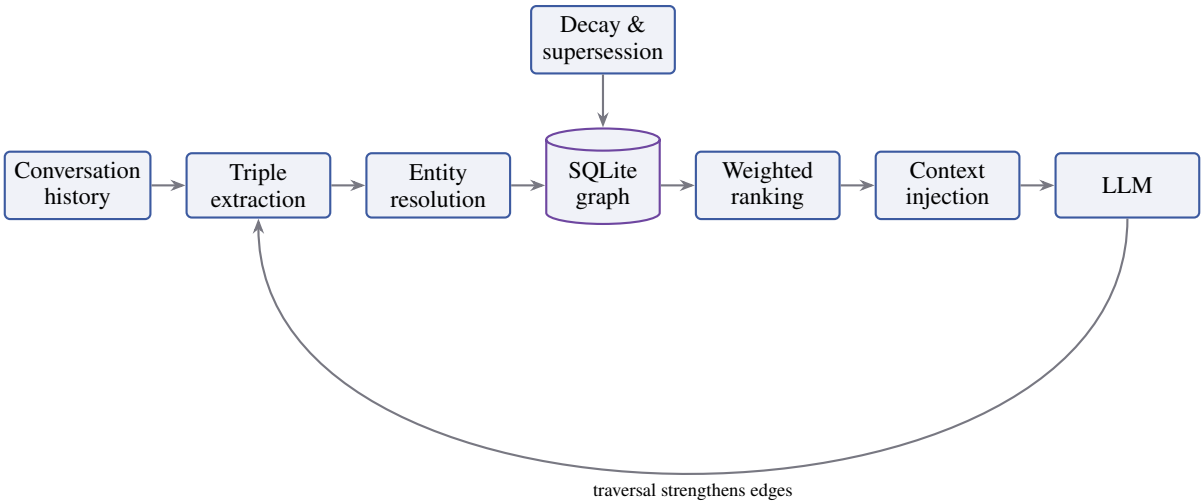


Figure 1: Asterism pipeline. Conversation history is extracted into (source, relation, target) triples, canonicalized by entity resolution, and written to a local SQLite graph. A background process applies decay and supersession. At inference time the highest-weight nodes are ranked and injected into the model's context. When the model traverses a relation in its response, the corresponding edge is strengthened, closing a Hebbian feedback loop.

Decay is measured in active session time rather than wall-clock time, so a concept does not fade while the user is simply away. This weight signal, accumulated over many sessions, is what the index ranks on.

Retrieval. On each turn, the graph is queried for its highest-weight nodes and their relations, which are serialized into a compact block and placed in the model's system context. This is the read path of the index. We deliberately keep it simple: an ordered scan by weight, returning the top-k nodes. Section 7 shows this is inexpensive even at ten thousand nodes.

3 The Graph as a Priority Index

The organizing idea of this work is that the structures a database uses to make an unindexed file queryable map onto the problem of making flat LLM memory prioritizable. We make the correspondence explicit because it drives both the design and the evaluation.

A flat memory summary is a heap file: an append-only list of facts with no access structure. You can read all of it, but you cannot ask it "what are the ten most important facts about this user" without scanning everything and imposing an ordering after the fact. A B-tree [2] solves the analogous problem for ordered files by maintaining a separate structure whose keys let a reader locate relevant records in logarithmic rather than linear cost. The key is not the data; it is a priority-ordered pointer into the data.

Asterism's weighted graph plays the role of that index over the heap of conversational facts. **Node weight corresponds to the index key:** weight is the quantity the retrieval step orders on, so a high-weight node is a high-priority pointer, surfaced first just as a B-tree directs a reader to the most relevant page before touching the file. **Graph traversal corresponds to query planning:** answering a query by walking weighted relations is a plan over the memory space rather than a full scan of the flat summary. **Decay corresponds to an eviction policy:** pruning low-weight, unused nodes keeps the working index small and current, so retrieval priority reflects present rather than historical relevance. And crucially, **the underlying facts persist** — decaying a node removes it from the *index*, not from the record of what was said, mirroring the removal of an index entry without deleting the row.

Graph-structured memory for LLMs is not itself new (Section 8). Our claim is narrower: that the *weighting* of such a graph should be understood and evaluated as an index key, and that doing so predicts where the structure helps. The next section tests that prediction.

4 Core Evaluation: Index Structure versus Flat Recall

4.1 Design

Evaluating a memory system risks conflating three different benefits: the benefit of having memory at all, the benefit of the particular structure imposed on it, and the benefit of instructing the model to act on that structure. The graph condition as deployed carries all three: it supplies facts, it imposes a weighted structure, and its system prompt instructs the model to prioritize high-weight nodes. To separate them, we run five conditions against the *same underlying facts*:

1. **Graph.** The thirty highest-weight nodes are injected in weight order, each labelled with weight and type, with an instruction to prioritize high-weight nodes (the deployed condition). No condition injects edges.
2. **Flat list.** Every node label is injected alphabetically and unweighted: the same facts are present, with the weight ordering and salience signal removed.
3. **Flat list + instruction.** The same unordered list, with the prioritize-and-commit instruction the graph carries added. This isolates the *instruction*.
4. **Graph, neutral.** The same weighted structure, with the explicit prioritize/commit instruction removed. This isolates the *structure*.
5. **None.** No memory is injected.

Two contrasts do the work. GRAPH vs. FLAT LIST isolates structure-and-weighting from information content, since both see identical facts. GRAPH, NEUTRAL vs. FLAT LIST, together with FLAT LIST + INSTRUCTION vs. FLAT LIST, separates the effect of the structure from the effect of the instruction. The pre-registered rule was: if the effect is instructional, the instruction added to a flat list should reproduce it and the instruction-free graph should lose it; if it is structural, the reverse. That rule presumes the two routes are mutually exclusive. They are not. On decisiveness both antecedents fire, so the dichotomy does not resolve on the commit metric (Section 4). We therefore resolve it on accuracy, which the same five conditions support with no change to the frozen scorer, the query set, or the labels. NONE is a reference point.

We evaluate on two independently authored personal graphs of roughly fifty nodes each, organized into weighted domain/theme/concept chains. The two describe deliberately different users: one an independent technical researcher, one a freelance designer with caregiving and family obligations. Their domain mixes and priority *shapes* differ, so a replication across both is evidence of generality rather than of one graph's idiosyncrasy. For each graph we focus the quantitative comparison on *priority queries*: one hundred questions that ask what the user should focus on, what matters most, or which of two things to prioritize, each requiring the model to choose or rank rather than merely recall. The queries were generated by filling each graph's node labels into a fixed bank of question templates, and were frozen before any evaluation call; because the generator is templated and not one of the models under test, the query set cannot favor any model. For each query we derive the ground-truth highest-priority node from the seed weights, and flag as near-ties those whose top-two candidates fall within the decay step. Near-ties are excluded from the correctness metric, since "correct" is ill-defined for them, but retained for the commit/hedge metric, which does not depend on ground truth.

To keep the comparison clean across a large call budget we fix three things. Decoding temperature is pinned to 1.0 on every backend, so cross-model differences cannot be attributed to sampling settings; we report the mean over five independent runs per cell rather than a single draw, and give run-to-run consistency for every contrast so that sampling variance is visible rather than averaged away. The graph is reseeded from the pristine seed before *every* individual call, so no response conditions on graph state that an earlier call's write-back mutated; an invariant check aborts a call rather than run it if the pre-call seed ever drifts. And we repeat the entire five-condition design across five assistant models spanning four vendors — `claude-sonnet-4-6`, `claude-opus-4-8`, `gpt-5.5-2026-04-23`, `deepseek-v4-pro`, and `kimi-k3` — with five independent runs each. These five are all token-billed frontier models with no hard daily request caps, which is what makes a one-hundred-query campaign feasible; four further models appeared in earlier pilot runs and were excluded on quota and throughput grounds, disclosed in full in Appendix D. The graph-maintenance extractor is a separate, fixed model held constant across all conditions (a local Llama 3.2 3B served over

Ollama, with an optional `claude-haiku-4-5` path). Concept-label embeddings come from `BAAI/bge-small-en-v1.5` [10], 384 dimensions, also held fixed. The assistant model is therefore the only moving part; because the graph is reseeded before every call, extractor output never enters the evaluated context.

4.2 Metric

We score each response with a single binary metric: does it COMMIT to a specific ranked answer, or HEDGE? A response commits if it names a single top priority or makes an explicit choice and justifies it; it hedges if it defers the decision to the user, enumerates options without selecting, or answers "it depends." The classifier is a deterministic rule, pre-registered and frozen before the evaluation run. We validated it on a held-out set of eleven hand-labeled responses drawn from earlier development runs, not from the evaluation set; it agreed on 11 of 11. We report the frozen classifier's output without post-hoc adjustment. Restricting the metric to priority queries is deliberate: commit-versus-hedge is only meaningful for questions that call for a ranking. Descriptive recall queries, for which a flat list is sufficient by construction, do not admit this metric and are excluded from it.

Because the entire dependent variable rests on this classifier, we also validate it against human judgment on the evaluation data itself, not only the development set. One author independently labeled all 36 responses of one full run (twelve queries across the three core conditions) as commit or hedge without seeing the classifier's labels: raw agreement was 83% and Cohen's κ was 0.64 (substantial). All six disagreements run in the direction that understates the graph condition relative to the flat list, so the measured graph−flat gap is a conservative estimate of the true difference; we did not retune the classifier after seeing them. Appendix D gives the breakdown, and the scorer–human comparison sheet is released with the code.

4.3 Findings

Table 1 reports the result across five models, two graphs, and one hundred queries per graph. We read it in three passes.

Finding 1: the deployed system increases decisiveness in every model. Compare the GRAPH column (the weighted graph plus prioritization instruction, as deployed) to FLAT LIST (the same facts, unweighted). The graph−flat gap (column G−F) is positive for all five models on both graphs, from $+2.6$ (Kimi G1) to $+38.8$ (Sonnet G2). It is also run-consistent: GRAPH beats FLAT LIST in all five runs for most model–graph pairs. The weakest case is Kimi graph 1 at 3/5, the one cell whose interval spans zero. NONE commits least throughout. Even GPT shows clear positive gaps here ($+15.2$, $+18.4$), though at pilot scale it had committed at a high baseline that masked any structural benefit. With a larger and harder query set, its no-memory rate is no longer a ceiling. The decisiveness benefit is not confined to models that hedge by default; it appears across the panel.

Finding 2: the decisiveness is more accurate than flat memory, but absolute accuracy is low. A metric rewarding commitment could reward confident error. Because each seed graph has known weights, the correct top-priority item is defined. We score correctness only on queries whose ground-truth weight separation exceeds the decay step, excluding near-ties, for which "correct" is not well defined. That leaves 96 of 100 scorable queries on graph 1 and 69 of 100 on graph 2, which has many near-ties by construction. We lead with the absolute numbers rather than the relative lift. Among scorable queries, the graph condition identifies the actually-highest-priority item in the following fractions: Sonnet 23%/30%, Opus 31%/21%, GPT 38%/42%, DeepSeek 9%/10%, Kimi 12%/11% (graph 1 / graph 2). So even the strongest model is correct on roughly four in ten priority queries, and most models on one to three in ten. Absolute priority routing is a hard, unsolved problem, and the weighted graph does not solve it. What it does is improve it: in every model and graph the graph condition is more often correct than the flat list (e.g. Sonnet 23% vs. 8%, Opus 31% vs. 8%, GPT 38% vs. 18%), so the added commitment is disproportionately correct commitment rather than confident noise. The finding is a real relative gain on top of a low absolute base, and we state both. The correctness gaps for all three contrasts, with intervals, two-sided p values, and run consistency, are the right-hand block of Table 1 and Appendix C; absolute correctness and precision for every condition are in Table 4.

Finding 3: decisiveness does not separate structure from instruction; accuracy does, and the two are complementary. Findings 1–2 concern the deployed system, which couples structure with

Model	Gr	Commit rate (%) None	Flat	Fl+in	Gr,neut	Graph	Commit-rate gap Fi−F	Gn−F	G−F	Correctness gap Fi−F	Gn−F	G−F
Sonnet	G1	0.8	26.0	32.4	45.4	47.8	**+6.4**	**+19.4**	**+21.8**	−0.2	**+14.2**	**+15.0**
Sonnet	G2	0.2	10.8	32.2	31.0	49.6	**+21.4**	**+20.2**	**+38.8**	**+8.7**	**+7.5**	**+24.1**
Opus	G1	1.0	13.2	30.8	36.8	49.2	**+17.6**	**+23.6**	**+36.0**	**+7.9**	**+12.9**	**+22.9**
Opus	G2	1.6	5.8	31.0	22.2	38.4	**+25.2**	**+16.4**	**+32.6**	**+11.3**	**+7.5**	**+17.7**
GPT	G1	23.6	35.6	40.4	40.8	50.8	+4.8	**+5.2**	**+15.2**	+0.4	**+13.3**	**+19.6**
GPT	G2	30.8	36.0	50.6	40.2	54.4	**+14.6**	+4.2	**+18.4**	**+8.7**	**+8.1**	**+16.5**
DeepSeek	G1	5.0	10.6	17.4	—	15.0	**+6.8**	—	**+4.4**	+2.9	—	**+6.2**
DeepSeek	G2	3.8	7.4	13.4	14.4	16.8	**+6.0**	**+7.0**	**+9.4**	**+5.2**	**+7.0**	**+7.0**
Kimi	G1	2.0	15.2	20.6	12.6	17.8	**+5.4**	−2.6	+2.6	+2.7	+0.8	+4.2
Kimi	G2	1.8	8.0	11.4	11.2	16.2	+3.4	+3.2	**+8.2**	+1.2	+3.8	**+6.1**

Table 1: Left: mean commit rate (of 100 priority queries) over five runs, by model, graph, and condition. Right: the three single-contrast gaps against FLAT LIST, on commit rate and on correctness per scorable query — **Fi−F** = instruction only (flat list plus instruction), **Gn−F** = structure only (graph with instruction removed), **G−F** = the deployed system (both). **Bold** marks a gap whose 95% percentile bootstrap interval excludes zero, paired by query over 10,000 resamples; the intervals, two-sided p values, and run consistency behind every bolded and unbolded entry are in Appendices B and C. Correctness is scored only on the queries whose ground-truth weight margin exceeds one decay step (96 of 100 on G1, 69 of 100 on G2); absolute correctness levels are in Table 4. Decoding pinned at temperature 1.0 on every backend, graph reseeded from a pristine seed before every call, all conditions scored by the same frozen held-out-validated classifier. DeepSeek's G1 GR,NEUT cell is omitted: the model returned recurring empty responses there (see text).

instruction. The design isolates each: GR,NEUT is the weighted structure with the instruction stripped, FL+IN the instruction without the structure. On decisiveness both work, by indistinguishable amounts. The instruction-only gap (Fi−F) is positive in all ten cells (mean $+11.2$ points, range $+3.4$ to $+25.2$) and separable from zero in eight. The structure-only gap (Gn−F) is positive in eight of nine measured cells (mean $+10.7$) and separable in six (Table 3, Figure 2). A twelve-query pilot had shown the structure-only route separating cleanly, and we read that as evidence the benefit was structural. At one hundred queries the reading does not survive: a bare instruction to prioritize and commit, carrying no weights and no structure, buys about as much decisiveness as the weighted graph does. A commit-rate gap cannot attribute the effect to structure, and we retract that stronger claim.

Accuracy separates them. Against the flat list, the structure-only condition improves correctness in all nine measured cells, separably from zero in seven. The instruction alone improves it in nine of ten cells but separably in only five, and it is flat in exactly the two cells where it moved decisiveness most (Sonnet G1 -0.2, GPT G1 $+0.4$). The sharper contrast is precision among committed answers — of the answers a model does commit to, what fraction are right. The structure-only condition is more precise than the instruction-only condition in all nine measured cells, by $+5.1$ to $+31.4$ points. That direction is unanimous, but only four of nine intervals exclude zero and run consistency is mixed (5/5 runs in three cells, 2/5 in one). We therefore report the sign pattern rather than per-cell effect sizes. Because cells share query sets across models, we attach no pooled significance statistic to the unanimity.

The components are complementary, not redundant. The deployed system's correctness gain matches or exceeds the better of its two single-component gains in all ten cells and strictly exceeds it in nine, and it is the only condition whose correctness gain excludes zero in nine of ten cells. The instruction gets a model to answer; the weighted structure is what makes the answer right; coupling them is worth more than either alone. Kimi graph 1, the weakest cell in the panel, is where nothing separates from zero on either metric, and DeepSeek's graph-1 structure-only cell is missing to the empty-response failure discussed below.

Qualitatively, no-memory responses hedge honestly ("I don't have your priorities"), while flat-list responses, holding the same facts, defer the choice back to the user. Hedging is defensible when no priority signal is present, so commitment measures resolution rather than correctness; Finding 2 supplies that check.

We record one model-specific failure, because the guard against it is part of the method. DeepSeek, in graph_neutral on graph 1, repeatedly returned responses reporting normal completion yet containing no text while still consuming out-

put tokens. A universal guard that refuses to score empty or truncated content caught every instance and aborted the affected cell rather than recording a spurious hedge, so we report that cell as missing; it caught an analogous failure in another model during development (Appendix C). Detecting and excluding such outputs, rather than silently scoring them as hedges, is a necessary property of automated LLM evaluation.

5 Entity Resolution and Its Ceiling

An index is only as good as the keys it orders. Because Asterism populates its graph automatically from extracted triples, the same concept arrives under many surface forms across sessions; without canonicalization "AI memory tools," "AI memory tooling," and "AI Memory Tools" become three nodes, each accumulating a fraction of the weight the underlying concept deserves. Fragmentation splits the salience signal and can prematurely decay a concept that is, in aggregate, highly active.

Two-stage resolution. We resolve labels at write time in two stages. A normalized exact/prefix match lowercases and whitespace-collapses labels and merges a new label that matches or forms a prefix of an existing one; this is cheap, precise, and caught every clear duplicate in our live label set. Because it cannot merge meaning-equivalent variants sharing no prefix, a second stage falls through to cosine similarity between label embeddings from a compact local model [10], merging above a calibrated threshold. The model runs offline on CPU on the write path only, so retrieval latency is unaffected.

The measured ceiling. The useful result is not that embedding resolution works but where it stops, measured against the live graph; the pattern held across every embedding model tested. Reworded or reordered labels reusing the same vocabulary score 0.92–0.98 and merge correctly ("CLI Tools"/"CLI tool" 0.982, "AI Memory Tools"/"AI memory tooling" 0.969). *True synonym substitution* is where it fails. "LLM requires GPUs" against "Large Language Models need graphics processors" scores only 0.745, inside the range of merely topically related pairs that must not merge ("Philosophy & Identity"/"Research Identity" 0.77–0.88). No single threshold separates the two kinds of pair. We calibrate to 0.92, below which false merges begin, making the resolver conservative: it eliminates rewording-driven fragmentation, the common case, while declining synonym pairs it cannot distinguish from distinct ones. Closing that gap needs supervision, either a fine-tuned equivalence model or an LLM semantic check, which we leave to future work.

6 Supersession: Demoting Contradicted Facts

An index over a live store must handle facts that change. When a user who "uses Postgres" switches to SQLite, naive decay eventually fades the old edge, but until it does both facts coexist and both can be retrieved as current. We add an explicit supersession mechanism scoped to the narrow, detectable case: a new edge from the same source, with the same relation, to a different target. On such a conflict the prior edge is marked superseded via a nullable self-reference and its time-to-live dropped so it becomes eligible for pruning; superseded edges are excluded from default injection, so the model sees only the current fact. A node is withheld only when *every* fact-edge to it is superseded, so a concept with any live relation still appears. Superseded edges are not deleted — they remain recoverable for explicit history queries ("what did I used to use"), preserving a distinction the flat summary cannot represent: a fact that is outdated versus one that was never true. The rule is deliberately conservative, avoiding open-ended contradiction detection that would risk demoting facts that only appear to conflict.

7 Latency and Scaling

For the graph to serve as an index without harming interactive use, the read path must stay cheap as the graph grows. We benchmarked the retrieval step (loading and weight-ordering nodes and edges for injection, excluding any model call) at three graph sizes over 20 runs each. Mean latency was 0.34 ms at 102 nodes, 2.81 ms at 1,006, and 33.87 ms at 10,051 (P95 38.7 ms). A $99\times$ node increase yields a $100\times$ latency increase, so retrieval scales approximately linearly. It stays under 34 ms at ten thousand nodes, negligible beside the model call it precedes. The linear rather than sub-linear factor is a full-table weight ordering, reducible with a secondary index on the weight column. The supersession filter adds 12–18% to read latency (30.1 to 35.2 ms at 10,051 nodes), a small cost of correctness within the interactive budget.

8 Related Work

LLM memory systems. MemGPT frames context management as a virtual memory hierarchy, paging information between a limited context window and external storage under LLM control [9]. Mem0 provides a memory architecture that extracts and consolidates salient facts from conversation, including a graph-structured variant [8]. Asterism shares the goal of persistent conversational memory but differs in emphasis: rather than optimizing recall or context-window management, it treats *prioritization* as the core function and evaluates the weighted graph specifically as an index that supplies salience. Our five-condition comparison separates that prioritization benefit from the recall benefit those systems primarily target, and separates the structure from the instruction to use it.

Importance-scored and graph-structured memory. The nearest prior art to a salience-weighted memory is the memory stream of generative agents, which retrieves observations by a weighted sum of recency, relevance, and an LLM-assigned importance score [3]; A-MEM similarly builds an evolving, interlinked note structure over agent memories [4]. Asterism differs in what the weight is and what is claimed for it. Weight accrues from repeated mention, decays on disuse, and attaches to graph nodes rather than episodes. We evaluate it as an *index key*: we test whether ordering the injected context by weight changes the assistant's answer, not whether an importance score improves retrieval recall. Structurally, GraphRAG builds entity graphs and community summaries for global queries [5], HippoRAG applies personalized PageRank over a knowledge graph for multi-hop retrieval [6], and Zep maintains a temporal knowledge graph with explicit fact invalidation [7], close in spirit to our supersession step (Section 6). Those systems optimize retrieval over a corpus or episode store. Our unit is a small personal graph, and our dependent variable is whether the assistant commits to a priority at all. Our contribution is the controlled decomposition of that effect into structural and instructional components.

Retrieval-augmented generation. RAG retrieves passages by similarity to a query [11]: it answers "what is relevant to this query," whereas Asterism's weighting answers "what is persistently important to this user," accumulated across sessions rather than computed per query. We take the empirical comparison between the two as the forward direction (Section 9).

Indexing and biological memory. The B-tree [2] is the canonical priority-ordered access structure over a file too large to scan; we borrow its framing directly as a design and evaluation lever. The weighting dynamics draw on Hebbian plasticity, in which co-active connections strengthen [1, 12], used as motivation for the traversal-strengthening rule, not a claim of biological fidelity.

9 Conclusion

We presented Asterism, a local-first system that builds a weighted, decaying knowledge graph from conversation history and uses it as a priority index over flat LLM memory. Across five conditions, two authored graphs, one hundred queries per graph, and five models from four vendors, the deployed system is more decisive than a flat list of the same facts in every model, and more accurate in every model. Absolute priority-routing accuracy nonetheless stays low, 42% at best and often far less, so the graph improves a hard problem without solving it. Decomposing that benefit, decisiveness does not separate the weighted structure from a bare instruction to prioritize: both raise commitment by comparable amounts, and we retract the stronger structural claim a twelve-query pilot had supported. Accuracy does separate them: the structure is more precise among committed answers in all nine measured cells. The components are also complementary, with the deployed pairing matching or exceeding the better single component in all ten cells. We also characterized entity resolution and supersession, showed retrieval scales approximately linearly to ten thousand nodes, and report the low absolute ceiling, the one weak model, and a model-specific empty-response failure our guard excluded.

The comparison these results motivate, which we do not frame as a contest, is against similarity retrieval. Similarity retrieval answers "what is relevant to this query"; a weighted decaying graph answers "what is persistently important to this user." The graph also adds a traceable, weight-grounded justification for why an item surfaced, and direct user control over what is remembered. Whether that paradigm beats a strong retrieval baseline on prioritization accuracy is worth answering either way, and it is where we take this next.

Limitations

Our core metric is a single binary outcome (commit vs. hedge), scored over one hundred priority queries per graph, five models, and five runs each. This is a substantial sample, but the queries are templated variations on a fixed bank of prioritization phrasings, so they span the intended task without spanning the full range of ways real users express priority. Two graphs is better than one but is still not a population of users; the two were authored to differ in domain and priority shape, yet both were authored by us, and author-constructed personas may share latent structure a real user distribution would not.

The system named in our title is not what the evaluation exercises. The weights in both benchmark graphs are authored, and the graph is reseeded from that pristine state before every call, so the reinforcement rule, the session-time decay, entity resolution and supersession never shape the evaluated context. What we measure is the index: whether a weight signal, once present, is usable as a ranking key. Whether the Hebbian loop produces a weight signal that tracks what a user actually treats as important is untested here, and it is the first thing a deployment study would have to establish. The strengthening rule is also ungated, in that an edge is reinforced whenever the model traverses it whether or not the answer was correct; Kairos [14] gates consolidation on a quality check for exactly this reason, and we did not test a gated variant. A further implication we do not test is the negative half of our own prediction (Section 3): that the graph's advantage should largely vanish on pure recall queries, which our commit/hedge metric cannot score.

Two limitations of the effect itself deserve emphasis. First, absolute accuracy is low: even the best model identifies the true top priority on only about 40% of scorable queries, and most models on 10–30%, so our claim is a relative improvement over flat memory, not a solved priority-routing problem. Second, the structure-only effect is not uniform, and which metric it is measured on matters. On *commit rate* the structure-only gap is separable from zero in only six of the nine measured cells — the interval includes zero for GPT graph 2 and both Kimi cells, and Kimi graph 1 is outright negative (-2.6 points) — which is part of why we do not rest the structural claim on decisiveness. On *correctness* the structure-only gap is positive in all nine measured cells and separable in seven, the two Kimi cells being the exceptions at $+0.8$ and $+3.8$ points; on precision among committed answers it is positive in all nine measured cells but separable in only four. Kimi is the weakest and least consistent model throughout, and DeepSeek's graph-1 structure-only cell is missing entirely because that model returned recurring empty responses in that condition. So the honest scope is that the correctness effect holds across nearly the whole panel, with one weak model, one missing cell, and several underpowered cells stated openly. Our controls isolate structure from instruction, but not the components within the structure, and the graph and flat arms differ in more than one way. The graph conditions inject the thirty highest-weight nodes with each weight printed; the flat conditions inject every node label alphabetically with no weights. The graph arm therefore receives the salience signal, a candidate set already pruned to the high-weight end, and an ordering, while the flat arm receives none of the three; all five prompts are reproduced verbatim in Appendix A. We did not run a weighted flat list, which would separate showing the weights from arranging them as a graph, nor a shuffled-weight condition holding topology fixed, which would separate the weights from the graph shape. What we show is that the deployed bundle beats an unranked list, the comparison a user actually faces, and not which element of the bundle is responsible. That the effect is bounded well below ceiling bears on this: a model handed an explicit ranking key identifies the top-weighted item on only 9–42% of scorable queries, so the contrast is not reducible to reading the answer off the prompt. Our panel also omits models we attempted but could not complete at this scale — two Gemini models (daily-quota-limited) and two free-tier open-weight models (throughput and task-comprehension failures) — which is a disclosed constraint of cross-vendor evaluation on consumer and free tiers.

We also do not compare end-task accuracy against a retrieval baseline such as RAG, or against external memory systems on a shared benchmark; the findings here are characterized behavioral effects on this setup, and the retrieval comparison is the forward direction outlined in the conclusion rather than a result we claim now. The entity-resolution ceiling is a limitation of the current unsupervised approach: true synonym pairs are not reliably merged and can still fragment the index. Supersession is scoped to the same-source/same-relation/different-target case and misses subtler

contradictions. Because decay is measured in active-session rather than wall-clock time, decay topology depends on usage frequency: a daily user prunes old interests quickly while a sparse user's graph decays little in real time and can accumulate stale structure over a year. Latency, while linear and small, is measured on synthetic graphs and on retrieval alone; end-to-end latency is dominated by the model call. Finally, the system is single-user and local by design, and we have not studied multi-user or privacy-preserving extensions.

Ethics Statement

Asterism is local-first: the graph is stored on the user's own device and no conversational data is transmitted except the user's prompt and the context they choose to inject into their model provider. Deleting the local store removes all derived data. The system infers user priorities from conversation history, and such inferences can be sensitive; because the data and index remain under the user's direct control and can be inspected or deleted, we regard the privacy surface as limited, but any deployment that centralized these graphs would raise concerns that the present design specifically avoids.

Declaration on Generative AI

The author used generative AI tools to assist with code for the system implementation and evaluation harness, and to assist with the language of the manuscript, by paraphrasing and polishing the author's own text for grammar, clarity, and organization rather than to generate new content. LLMs also appear as objects of study in the evaluation, as the assistant models being measured; those uses are described in the relevant sections. All research design, experimental decisions, analysis, and conclusions are the author's own, and the author reviewed and verified all content and takes full responsibility for it.

Acknowledgements

The author thanks the NORA 2026 reviewers, whose comments prompted the disclosures added to the Limitations section and the prompts reproduced in Appendix A. Generative AI tools were used in preparing this work: to assist with code for the system implementation and evaluation harness, and to assist with language editing of the author's own manuscript text. No text presenting research ideas, results, or conclusions was model-generated, and all research design, experimental decisions, analysis, and conclusions are the author's own. The Declaration on Generative AI above states this in full.

References

[1] Donald O. Hebb. 1949. *The Organization of Behavior: A Neuropsychological Theory*. Wiley, New York.

[2] Rudolf Bayer and Edward M. McCreight. 1972. Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3):173–189.

[3] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[4] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[5] 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*.

[6] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[7] 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*.

[8] 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*.

[9] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.

[10] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged resources to advance general Chinese embedding. *arXiv preprint arXiv:2309.07597*.

[11] 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.

[12] Carla J. Shatz. 1992. The developing brain. *Scientific American*, 267(3):60–67.

[13] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA.

[14] Pragya Singh and Stanley Yu. 2025. Validation-gated Hebbian learning for adaptive agent memory. In *Proceedings of the 1st Workshop on Knowledge Graphs & Agentic Systems Interplay (NORA 2025)*, volume 4162 of *CEUR Workshop Proceedings*, pages 42–54.

A Injected context for all five conditions

The system prompt is the only thing that differs across conditions; the user turn is the query verbatim and the assistant has no conversation history. In the two graph conditions $k = 30$: the thirty highest-weight nodes are injected, each rendered as `- {label} (weight: {w}, type: {t})` with the weight rounded to an integer. In the two flat conditions every node label is injected, alphabetically ordered, with no weight and no type. Both benchmark graphs hold roughly fifty nodes, so the graph conditions see a pruned candidate set and the flat conditions see all of it. **Relations are not injected in any condition**: the graph conditions receive a weighted node list, not a subgraph with edges, and the `TRAVERSAL` directive in the `graph` prompt only asks the model to label traversals it states in its own answer, which we parse for display. Table 2 reproduces all five system prompts verbatim. Long lines are wrapped there for the page; the strings themselves are single lines.

B Bootstrap Confidence Intervals on the Commit-Rate Gaps

The gaps in Table 1 are differences of means over a shared, frozen query set, so they admit a paired interval estimate without any additional model calls. For each model–graph cell we reduce every query to its commit rate across the five runs, pair each condition against FLAT_LIST by query identifier, and resample the one hundred query pairs with replacement 10,000 times (seed fixed at 20260902), reporting the 2.5th and 97.5th percentiles of the resampled mean difference and a two-sided bootstrap p-value. Pairing is at the query level rather than the run level because runs are exchangeable draws under identical decoding while queries are the unit the two conditions share. The frozen scorer is used as-is; no label was recomputed or reinterpreted for this analysis. The procedure is a single script in the repository (`research/bootstrap_ci.py`) that reads the committed per-cell result files and regenerates `n100_bootstrap_ci.csv` deterministically; p-values are floored at $1/B$, which is the resolution the resample count supports.

C Absolute Correctness, Precision, and Follow-up Intervals

Table 1 reports gaps; this section reports the absolute levels those gaps are computed from, and the interval estimates behind every bolded and unbolded entry. Correctness is the fraction of *scorable* queries (ground-truth weight margin exceeding one decay step: 96 of 100 on graph 1, 69 of 100 on graph 2) on which the condition committed to the actually-highest-priority node. Precision is the fraction of *committed* answers that were correct, with the number of committed answers in parentheses; it is the metric that separates the structural from the instructional component, and its denominators vary widely across conditions, so we print them. The NONE column's precision rests on single-digit denominators in most cells and should not be read as an estimate. Interval estimates use the same procedure as Appendix B — paired by query over 10,000 resamples of the frozen labels, no additional model calls, no relabelling — computed by `research/followup_ci.py`. That script draws its randomness from a generator seeded per row from the cell's own identifiers, so adding a contrast cannot perturb an interval already published in Table 3. Table 5 covers the instruction-only contrast on both metrics together with the structure-only correctness gap; Table 6 covers the deployed correctness gap and the precision difference between the two single components. Every table body in this paper is emitted from the committed result files by `research/make_paper_tables.py` rather than transcribed.

graph

```
You are Asterism, a personal knowledge assistant
for {user}.

You have access to {user}'s personal knowledge
graph. The following nodes represent their current
priorities and thought patterns, weighted by
recency and usage frequency:

ACTIVE NODES (highest weight first):
{node_lines}

When answering, reference these nodes naturally.
Prioritize information from high-weight nodes.
Note when you are drawing on specific parts of
their graph.
When you traverse a concept explicitly, format it
as: TRAVERSAL: NodeA -> NodeB
```

graph_neutral

```
You are Asterism, a personal knowledge assistant
for {user}.

You have access to {user}'s personal knowledge
graph. The following nodes represent their current
priorities and thought patterns, weighted by
recency and usage frequency:

ACTIVE NODES (highest weight first):
{node_lines}

Answer using this graph if relevant.
```

flat_list

```
You are Asterism, a personal knowledge assistant
for {user}.

Here are topics {user} has previously discussed,
in no particular order:

{node_lines}

Answer using this list if relevant.
```

flat_list_prioritized

```
You are Asterism, a personal knowledge assistant
for {user}.

Here are topics {user} has previously discussed,
in no particular order:

{node_lines}

When answering, reference these topics naturally.
Prioritize information from these topics. Note
when you are drawing on specific parts of this
list.
```

none

```
You are Asterism, a personal knowledge assistant
for {user}.

Answer using only the current conversation. You
have no access to {user}'s knowledge graph or
prior history.
```

Table 2: System prompts for all five retrieval conditions, verbatim. `{node_lines}` is the injected memory block: in the two graph conditions it is the thirty highest-weight nodes in weight order, each line `- {label} (weight: {w}, type: {t})`; in the two flat conditions it is every node label, alphabetical, with no weight and no type. No condition injects edges.

		Deployed: graph − flat_list					Structure-only: graph_neutral − flat_list				
Model	**Gr**	Gap	95% CI	p	Runs>0	SD	Gap	95% CI	p	Runs>0	SD
Sonnet	G1	+21.8	[15.2, 28.6]	<0.001	5/5	4.3	+19.4	[11.6, 27.6]	<0.001	5/5	3.9
Sonnet	G2	+38.8	[31.6, 45.8]	<0.001	5/5	4.9	+20.2	[13.8, 26.8]	<0.001	5/5	1.5
Opus	G1	+36.0	[30.4, 41.8]	<0.001	5/5	4.6	+23.6	[18.4, 28.8]	<0.001	5/5	5.8
Opus	G2	+32.6	[27.2, 38.0]	<0.001	5/5	6.4	+16.4	[11.0, 21.8]	<0.001	5/5	7.3
GPT	G1	+15.2	[9.4, 21.0]	<0.001	5/5	5.5	+5.2	[0.4, 10.0]	0.037	5/5	3.6
GPT	G2	+18.4	[13.4, 23.6]	<0.001	5/5	3.6	+4.2	[−1.4, 9.8]	0.146	4/5	5.4
DeepSeek	G1	+4.4	[0.6, 8.4]	0.025	4/5	4.7	—	—	—	—	—
DeepSeek	G2	+9.4	[5.2, 13.8]	<0.001	5/5	6.1	+7.0	[2.6, 11.4]	0.002	5/5	4.2
Kimi	G1	+2.6	[−3.0, 8.2]	0.369	3/5	6.0	−2.6	[−7.4, 2.2]	0.293	2/5	6.7
Kimi	G2	+8.2	[3.4, 13.2]	0.001	5/5	3.3	+3.2	[−0.6, 7.0]	0.119	5/5	0.4

Table 3: Paired bootstrap on the commit-rate gaps, in percentage points. RUNS>0 counts how many of the five runs show a positive gap; SD is the standard deviation of the gap across those five runs. The deployed gap excludes zero in nine of ten cells (Kimi graph 1 is the exception); the structure-only gap excludes zero in six of the nine measured cells, the exceptions being GPT graph 2 and both Kimi cells. DeepSeek's graph-1 structure-only cell was aborted by the empty-response guard and is not estimated.

D Reproducibility Details

Scorer–human agreement. Of the six disagreements in the 36-response validation run, the frozen classifier labeled as HEDGE four graph-condition responses a human read as commitments — for instance a bare imperative ("Ship the package") whose form the frozen rule's vocabulary did not cover — and over-credited two flat-list responses as commitments. Both directions understate the graph−flat gap, which is why we report it as a conservative estimate. We did not retune the classifier after seeing these disagreements, since doing so would fit the frozen scorer to the evaluation data it is used to score.

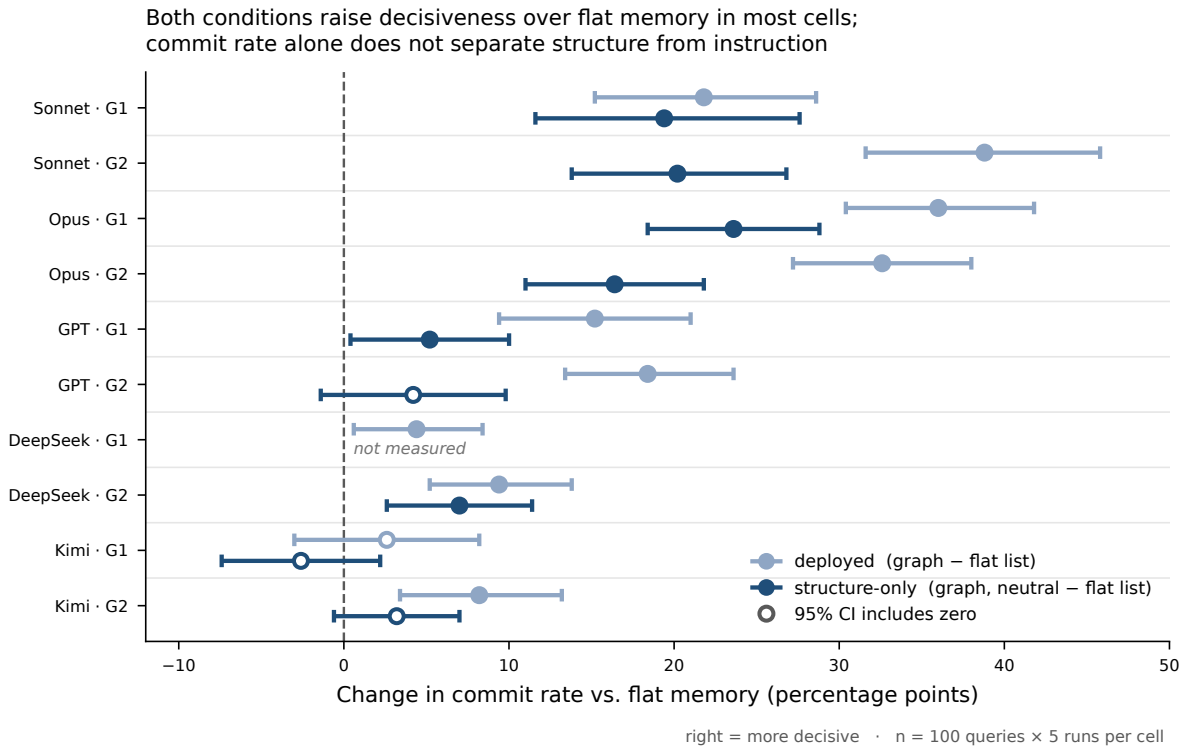


Figure 2: Change in commit rate relative to a flat list of the same facts, by model and graph, with 95% paired bootstrap confidence intervals. Filled markers denote intervals that exclude zero; open markers denote intervals that include it. The deployed contrast (graph plus prioritization instruction) is separated from the structure-only contrast (weighted graph, instruction removed) to show how much of the deployed benefit survives stripping the instruction. DeepSeek's graph-1 structure-only cell is annotated rather than plotted because the empty-response guard aborted it.

		Correct per scorable query (%)					**Precision among commits (%), (n committed)**				
Model	**Gr**	None	Flat	Fl+in	Gr,neut	Graph	None	Flat	Fl+in	Gr,neut	Graph
Sonnet	G1	0.0	8.3	8.1	22.5	23.3	0.0 (4)	32.0 (125)	25.3 (154)	50.9 (212)	48.9 (229)
Sonnet	G2	0.3	5.8	14.5	13.3	29.9	100.0 (1)	45.5 (44)	41.7 (120)	47.9 (96)	61.7 (167)
Opus	G1	0.0	8.1	16.0	21.0	31.0	0.0 (5)	60.0 (65)	51.7 (149)	56.7 (178)	63.1 (236)
Opus	G2	0.3	3.5	14.8	11.0	21.2	50.0 (2)	57.1 (21)	43.2 (118)	49.4 (77)	56.2 (130)
GPT	G1	7.9	18.3	18.8	31.7	37.9	34.9 (109)	52.7 (167)	47.4 (190)	78.8 (193)	76.2 (239)
GPT	G2	17.7	25.5	34.2	33.6	42.0	49.2 (124)	62.4 (141)	61.1 (193)	71.6 (162)	69.7 (208)
DeepSeek	G1	1.7	2.7	5.6	—	9.0	33.3 (24)	26.0 (50)	34.6 (78)	—	62.3 (69)
DeepSeek	G2	2.3	3.5	8.7	10.4	10.4	53.3 (15)	54.5 (22)	57.7 (52)	75.0 (48)	70.6 (51)
Kimi	G1	0.0	8.3	11.0	9.2	12.5	0.0 (10)	54.1 (74)	54.6 (97)	73.3 (60)	72.3 (83)
Kimi	G2	0.3	4.9	6.1	8.7	11.0	16.7 (6)	56.7 (30)	56.8 (37)	69.8 (43)	60.3 (63)

Table 4: Absolute correctness and precision by model, graph, and condition, the levels underlying the correctness gaps in Table 1. Correctness is over scorable queries (96 on graph 1, 69 on graph 2); precision is over committed answers only, with committed counts in parentheses, pooled over the five runs. Absolute correctness is low throughout — the best cell is GPT graph 2 at 42.0% — which is the ceiling we report rather than the relative gap alone. Precision is where the structure-only condition separates from the instruction-only condition: it is higher in all nine measured cells. NONE-column precision rests on denominators as small as one and is reported only for completeness.

Models excluded from the full-scale campaign. Earlier pilot runs also included two Gemini models and two free-tier open-weight models. The Gemini models' 250-request daily quota made a hundred-query, five-condition, two-graph campaign infeasible within the evaluation window. The two free-tier open-weight models failed both on throughput and on task comprehension, frequently naming the user node itself as the top priority. We therefore report the five token-billed models at full scale and exclude these four, which is a constraint of evaluating across vendors on consumer and free tiers

Model	Gr	Gap	95% CI	p	Runs>0	SD
Commit rate: FL+IN − FLAT *(instruction only)*						
Sonnet	G1	**+6.4**	[0.6, 12.2]	0.034	5/5	3.9
Sonnet	G2	**+21.4**	[16.0, 26.8]	<0.001	5/5	4.5
Opus	G1	**+17.6**	[12.6, 23.2]	<0.001	5/5	9.0
Opus	G2	**+25.2**	[20.6, 29.8]	<0.001	5/5	5.0
GPT	G1	+4.8	[−0.2, 9.8]	0.062	5/5	3.0
GPT	G2	**+14.6**	[9.2, 20.0]	<0.001	5/5	4.5
DeepSeek	G1	**+6.8**	[2.6, 11.0]	0.002	4/5	7.2
DeepSeek	G2	**+6.0**	[1.6, 10.6]	0.010	5/5	2.7
Kimi	G1	**+5.4**	[1.2, 9.4]	0.014	4/5	8.2
Kimi	G2	+3.4	[−0.2, 7.0]	0.066	4/5	3.6
Correct per scorable query: FL+IN − FLAT *(instruction only)*						
Sonnet	G1	−0.2	[−3.5, 3.1]	0.959	1/5	3.5
Sonnet	G2	**+8.7**	[4.1, 13.6]	<0.001	5/5	4.2
Opus	G1	**+7.9**	[3.8, 12.3]	<0.001	4/5	7.4
Opus	G2	**+11.3**	[6.4, 16.5]	<0.001	5/5	4.2
GPT	G1	+0.4	[−4.2, 5.0]	0.888	2/5	4.9
GPT	G2	**+8.7**	[2.9, 14.5]	0.005	5/5	4.7
DeepSeek	G1	+2.9	[0.0, 5.8]	0.054	4/5	2.3
DeepSeek	G2	**+5.2**	[2.0, 8.7]	<0.001	5/5	2.6
Kimi	G1	+2.7	[−0.6, 6.0]	0.111	4/5	2.4
Kimi	G2	+1.2	[−1.7, 4.1]	0.506	3/5	4.0
Correct per scorable query: GR,NEUT − FLAT *(structure only)*						
Sonnet	G1	**+14.2**	[9.2, 19.4]	<0.001	5/5	2.5
Sonnet	G2	**+7.5**	[2.9, 12.2]	0.003	4/5	6.9
Opus	G1	**+12.9**	[8.3, 17.7]	<0.001	5/5	7.0
Opus	G2	**+7.5**	[3.8, 11.9]	<0.001	5/5	5.5
GPT	G1	**+13.3**	[8.1, 18.8]	<0.001	5/5	5.2
GPT	G2	**+8.1**	[0.6, 15.4]	0.033	3/5	10.3
DeepSeek	G1	—	—	—	—	—
DeepSeek	G2	**+7.0**	[2.6, 11.9]	<0.001	4/5	6.1
Kimi	G1	+0.8	[−3.1, 4.8]	0.707	4/5	6.0
Kimi	G2	+3.8	[−0.3, 7.8]	0.077	4/5	2.6

Table 5: Paired bootstrap intervals, in percentage points, for the instruction-only contrast on both metrics and the structure-only contrast on correctness. RUNS>0 counts how many of the five runs show a positive gap; SD is the standard deviation across runs. **Bold** marks an interval excluding zero. The instruction alone raises the commit rate in all ten cells (eight separably), matching the structure-only commit gap of Table 3 in magnitude — which is why decisiveness cannot attribute the effect to structure. On correctness the instruction alone is separable in only five of ten cells, against seven of nine for the structure.

Model	Gr	Gap	95% CI	p	Runs>0	SD
Correct per scorable query: GRAPH − FLAT *(deployed)*						
Sonnet	G1	**+15.0**	[9.6, 20.4]	<0.001	5/5	5.5
Sonnet	G2	**+24.1**	[16.8, 31.6]	<0.001	5/5	3.3
Opus	G1	**+22.9**	[17.3, 28.8]	<0.001	5/5	4.5
Opus	G2	**+17.7**	[12.5, 23.2]	<0.001	5/5	4.0
GPT	G1	**+19.6**	[14.0, 25.4]	<0.001	5/5	6.2
GPT	G2	**+16.5**	[10.1, 23.5]	<0.001	5/5	3.3
DeepSeek	G1	**+6.2**	[3.1, 9.4]	<0.001	5/5	2.9
DeepSeek	G2	**+7.0**	[3.2, 11.0]	<0.001	5/5	3.1
Kimi	G1	+4.2	[−0.6, 9.2]	0.104	4/5	4.5
Kimi	G2	**+6.1**	[1.7, 10.7]	0.008	5/5	3.1
Precision among commits: GR,NEUT − FL+IN						
Sonnet	G1	**+25.6**	[13.6, 36.9]	<0.001	5/5	6.3
Sonnet	G2	+6.2	[−6.8, 19.3]	0.348	3/5	18.4
Opus	G1	+5.1	[−6.4, 16.7]	0.390	2/5	14.2
Opus	G2	+6.1	[−8.0, 21.1]	0.410	4/5	17.4
GPT	G1	**+31.4**	[22.7, 40.2]	<0.001	5/5	9.8
GPT	G2	**+10.5**	[0.6, 20.2]	0.039	4/5	8.7
DeepSeek	G1	—	—	—	—	—
DeepSeek	G2	+17.3	[−0.8, 33.7]	0.062	4/5	25.6
Kimi	G1	**+18.7**	[4.4, 31.8]	0.011	4/5	17.0
Kimi	G2	+13.0	[−7.1, 33.8]	0.204	3/5	32.2

Table 6: Paired bootstrap intervals, in percentage points, for the deployed correctness gap and for the precision difference between the two single components. The deployed correctness gap excludes zero in nine of ten cells, more than either component alone. The precision contrast is positive in all nine measured cells but separable in four, with run consistency between 2/5 and 5/5: the direction is unanimous while individual cells are underpowered, so we report the sign pattern and not per-cell effect sizes. No pooled statistic is given because cells share query sets across models and are therefore not independent.

rather than a property of the models' best available configurations.

The system is implemented in Python over SQLite, with the graph loaded into NetworkX [13] for analysis. Triple extraction uses a small instruction-tuned model; entity-resolution embeddings use a compact local sentence-embedding model [10] producing 384-dimensional vectors, run offline on CPU. The evaluation graph, the five-condition comparison script and its multi-model runner, the latency benchmark, and the entity-resolution and supersession test suites are included in the public repository, together with the frozen scorer, the scorer–human agreement sheet, and all per-run results; reproduction instructions are in the repository README (https://github.com/biditdas18/asterism). The exact state of the code behind every number in this paper is tagged `nora2026-camera-ready`: https://github.com/biditdas18/asterism/tree/nora2026-camera-ready. Latency figures were produced by timing the retrieval function directly with a monotonic clock over 20 runs per scale, on synthetic graphs seeded with weighted domain/theme/concept chains; the real user database is never touched by the benchmark.

Multi-View Contract Graph: A Knowledge Graph-Based Retrieval Framework for Contract Information Extraction

Momojit Biswas
TCS Research, India
momojit.biswas@tcs.com

Preethu Rose Anish
TCS Research, India
preethu.rose@tcs.com

Abstract

Legal contract understanding often requires connecting information distributed across multiple clauses and cross-referenced provisions. While Retrieval-Augmented Generation (RAG) grounds Large Language Models (LLMs) in relevant evidence, conventional retrieval primarily prioritizes direct query relevance and may overlook supporting contractual context. We propose the Multi-View Contract Graph (MCG), a contract-specific knowledge graph that models semantic, entity, keyword, and structural relationships among clauses. By traversing these relationships, MCG retrieves supporting evidence that may not directly match the query but is necessary for accurate interpretation. We evaluate MCG on the Contract Understanding Atticus Dataset (CUAD), covering 15 information categories, including audit rights, governing laws, and liability-related provisions. Across five open-source LLMs, MCG consistently outperforms traditional retrieval, achieving a best F_1 score of 0.460 with Mistral-7B-Instruct-v0.3. Compared with BM25-RAG and ColBERT-RAG, the best MCG configuration improves F_1 by approximately 10.0% and 12.2%, respectively. These results demonstrate that modeling clause-level relationships improves retrieval and information extraction for legal contracts.

1 Introduction

Legal contracts are fundamental to business operations, governing commercial relationships, regulatory compliance, procurement, licensing, employment, and risk management (Junior et al., 2026; Khan, 2022). As organizations manage growing volumes of increasingly complex contracts, automated contract understanding has become essential for contract governance, legal analytics, and contract lifecycle management (Liu et al., 2025). Despite rapid advances in LLMs, contract understanding remains challenging because critical information is often distributed across multiple clauses and sections within a contract (Singh et al., 2025, 2026). Interpreting a contractual provision frequently requires supporting context found elsewhere in the agreement, such as related obligations, defined terms, dates, or notice related requirements (Wang et al., 2025). For example, a termination clause may depend on a notice period defined elsewhere, while a licensing provision may rely on distant definitions or exclusions. Thus, understanding contractual intent often requires reasoning across interconnected clauses rather than analyzing provisions in isolation.

Retrieval-Augmented Generation (RAG) has emerged as a widely adopted approach for grounding LLMs through lexical and semantic retrieval. However, contract understanding remains challenging because relevant evidence is often distributed across non-adjacent clauses, cross-referenced provisions, and structurally related sections (Liu et al., 2025). Conventional retrieval methods primarily optimize query relevance and may overlook supporting provisions needed for accurate interpretation. Such evidence may be linked through definitions, obligations, temporal conditions, cross-references, and recurring contractual concepts. Existing knowledge graph approaches have demonstrated the value of structured relationships for legal retrieval and reasoning, but largely focus on relationships among legal entities, statutes, cases, and legal concepts. In contrast, contract analysis requires modeling heterogeneous dependencies among clauses within the same document. To address this gap, we propose the Multi-View Contract Graph (MCG), a contract-specific knowledge graph that integrates semantic, entity-level, keyword-based, and structural relationships among clauses. Given a query, MCG first identifies relevant anchor clauses and then expands retrieval through graph-connected clauses to recover supporting evidence that may be missed by similarity-based retrieval. The resulting evidence is provided

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 53–62
November 10, 2026 ©2026 Association for Computational Linguistics

to an LLM for contract question answering and information extraction.

We evaluate MCG on contracts from the Contract Understanding Atticus Dataset (CUAD) (Hendrycks et al., 2021), selecting 15 categories from the original 41 annotations to cover diverse contractual information, including dates, parties, governing law, licensing, renewal, audit rights, and liability provisions. Definitions of the selected categories are provided in Table 1. While CUAD formulates contract review as an extractive span-detection task, our setting reformulates it as retrieval-augmented contract information extraction, where relevant and supporting evidence must first be retrieved and then interpreted to answer category-specific questions. This enables evaluation of whether retrieval methods can recover evidence distributed across multiple clauses rather than simply identify a predefined span. We compare MCG with extractive QA, BM25-RAG (Robertson and Zaragoza, 2009), ColBERT-RAG (Khattab and Zaharia, 2020), and Dense-Retrieval-RAG (Lewis et al., 2020) across five open-source LLMs (Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), Llama-3-8B-Instruct (Grattafiori et al., 2024), Saul-7B-Instruct-v1 (Colombo et al., 2024), and Qwen-7B-Chat (Bai et al., 2023)). MCG consistently achieves the strongest performance, with the best configuration attaining an F_1 score of 0.460 using Mistral-7B-Instruct-v0.3, improving over BM25-RAG and ColBERT-RAG by approximately 10.0% and 12.2%, respectively.

The contributions of this paper are threefold: (1) We propose a graph-enhanced retrieval strategy that extends retrieval beyond directly relevant clauses by leveraging weighted inter-clause relationships to recover supporting contractual context. (2) We introduce the Multi-View Contract Graph (MCG), which integrates semantic, entity-level, keyword-based, and structural relationships to model dependencies among contract clauses. (3) We demonstrate through evaluation on CUAD across 15 contract information categories and five open-source LLMs that MCG consistently outperforms extractive QA and multiple retrieval-based baselines.

2 Literature Review

Knowledge Graphs (KGs) have been widely explored for representing legal knowledge and supporting legal information retrieval, reasoning, and question answering. Early efforts focused on constructing structured legal knowledge bases from case law and statutory documents. For example, dAmato et al. (2025) developed an ontology-driven knowledge graph for European Court of Human Rights, while Jain et al. (2022) proposed an RDF-based legal knowledge graph for Indian Supreme Court judgments. Similarly, Sovrano et al. (2020) introduced a knowledge extraction framework that leveraged structured legal representations to support knowledge graph-driven legal question answering. These studies demonstrated the value of explicit knowledge representations for enhancing legal search and reasoning.

Subsequent research investigated the integration of knowledge graphs with legal text processing and semantic retrieval. Filtz et al. (2021) proposed a linked legal data framework connecting legal resources across jurisdictions through standards such as ELI and ECLI. The Lynx platform (Schneider et al., 2022) further combines legal knowledge graphs with multilingual NLP services, including entity linking, temporal analysis, semantic similarity computation, question answering, and summarization. These efforts highlighted the benefits of coupling structured knowledge with natural language processing capabilities, while also exposing practical challenges related to annotation costs, extraction complexity, and the limited availability of large-scale legal evaluation benchmarks.

More recently, knowledge graphs have been integrated with retrieval-augmented generations and large language models. Calvin Barron et al. (2025) proposed a legal KG-RAG architecture that combines vector retrieval, knowledge graphs, and topic modeling to support grounded legal question answering. Chen et al. (2026) introduced the Case-Enhanced Law Article Knowledge Graph (CLAKG), which combines statutory provisions, case law, graph embeddings, and constrained LLM for legal recommendation tasks. Similarly, Li et al. (2024) explored knowledge-enhanced LLMs for legal knowledge extraction, while Pham et al. (2026) demonstrated the effectiveness of graph-based reasoning through explicit modeling of legal concepts and inference rules. Collectively, these approaches illustrate a growing trend toward combining structured knowledge representations, semantic retrieval mechanisms, and LLMs to improve legal-AI systems.

Despite these advances, contract understanding

Table 1: Contract Information Extraction Categories Extracted from CUAD Contracts.

Category	Definition
Agreement Date	The date on which the agreement was executed or signed by the contracting parties.
Audit Rights	A contractual provision granting one party the right to inspect, verify, or review the records, processes, or compliance activities of another party.
Cap on Liability	A clause specifying the maximum amount of financial liability or damages that a party may be required to pay under the agreement.
Effective Date	The date on which the rights, obligations, and terms of the agreement become legally effective.
Exclusivity	A provision granting exclusive rights or restricting parties from entering similar agreements with competitors or third parties.
Expiration Date-Answer	The specific calendar date stated in the agreement on which the contract expires or terminates.
Expiration Date	A clause describing the conditions or timing under which the agreement expires or ceases to remain in effect.
Governing Law	The jurisdiction or legal framework whose laws govern the interpretation and enforcement of the agreement.
Insurance	A contractual requirement that one or more parties maintain specified insurance coverage during the term of the agreement.
License Grant	A provision defining the rights, permissions, and scope of use granted by one party to another over intellectual property, software, or other assets.
Non-Compete	A clause restricting a party from engaging in activities that compete with the business interests of another party during or after the agreement term.
Notice Period to Terminate Renewal	The advance notice required to prevent or terminate an automatic contract renewal.
Parties	The legal entities or individuals entering into the agreement, including any defined names or aliases used within the contract.
Renewal Term	The duration and conditions under which the agreement may be renewed or extended beyond its initial term.
Uncapped Liability	A provision specifying obligations or circumstances for which liability is not subject to any contractual monetary limit.

presents challenges that differ substantially from those encountered in case of law retrieval, statutory reasoning, or legal recommendation. Existing legal knowledge graph systems primarily model relationships among legal entities, cases, statutes, and legal concepts. In contrast, contract analysis often requires identifying dependencies among clauses within the same document, where critical contextual evidence may be distributed across non-adjacent sections connected through definitions, obligations, exceptions, temporal conditions, and structural references. Conventional retrieval methods generally prioritize direct query relevance and may therefore fail to recover supporting contractual context necessary for accurate interpretation. Moreover, existing KG-RAG approaches typically emphasize document-level or concept-level relationships rather than fine-grained clause-level dependencies.

To address this gap, we propose the Multi-View Contract Graph (MCG), a contract-specific retrieval framework that models clause-level relationships using multiple complementary views, including semantic similarity, named entities, contractual keywords, and document structure. By combining these heterogeneous relationship types within a unified graph representation, MCG enables retrieval to extend beyond directly relevant clauses and recover supporting contractual evidence required for downstream contract question answering and information extraction.

3 Methodology

3.1 Overview

We propose MCG, a contract-specific knowledge graph and graph-based retrieval framework for contract question answering and information extraction. The key motivation is that contractual information is often distributed across multiple clauses that may be related through shared entities, contractual concepts, or document structure. Conventional retrieval methods typically rank text chunks according to their direct relevance to a query and may therefore fail to recover supporting clauses required for complete contractual interpretation.

MCG addresses this limitation by representing each contract as a contract-specific knowledge graph that serves as a structured memory representation of contractual knowledge, whose nodes correspond to semantically coherent contract

chunks. Each node is enriched with three complementary views of contractual content: dense semantic embeddings, named entities, and contractual keywords. Relationships between nodes are modeled using semantic similarity, entity overlap, keyword overlap, and positional proximity within the document. At query time, MCG first identifies relevant clauses through semantic matching and subsequently expands retrieval through graph-connected clause expansion to recover supporting context. The retrieved clauses, together with their associated metadata, are then provided to a downstream LLM for answer generation. The framework consists of two stages: (i) contract preprocessing and graph construction, (ii) graph-based retrieval and LLM-based answer generation. An overview of the proposed method is illustrated in Figure 1.

3.2 Multi-View Contract Graph Construction

Given a contract document D, MCG first constructs a weighted graph representation of the document. Text is extracted from PDF files and normalized through Unicode standardization, correction of broken hyphenation, removal of page-level artifacts, whitespace normalization, and filtering of non-standard characters.

Unlike fixed-size chunking approaches, MCG employs semantic chunking to generate semantically coherent contract segments. The document is first divided into individual sentences, and each sentence is encoded using the all-MiniLM-L6-v2 (Wang et al., 2021) Sentence Transformer due to its favorable trade-off between retrieval quality and computational efficiency. The proposed MCG framework is model-agnostic and does not depend on a specific embedding architecture. Alternative embedding models, including domain-specific encoders such as Legal-BERT (Chalkidis et al., 2020) and Contract-BERT (Zheng et al., 2021), as well as retrieval-oriented models such as E5 (Wang et al., 2022) and BGE (Xiao et al., 2024), can be incorporated without modifying the graph construction or retrieval procedures. For two consecutive sentences with embeddings s_i and s_{i+1}, semantic continuity is measured using cosine similarity: $\mathrm{Sim}(s_i, s_{i+1}) = \frac{s_i \cdot s_{i+1}}{\|s_i\| \times \|s_{i+1}\|}$.

A chunk boundary is introduced whenever the cosine similarity between adjacent sentences falls below 0.80, provided that the current chunk satisfies a minimum length of 400 tokens. The similarity threshold was selected empirically to balance semantic coherence and chunk granularity. Lower thresholds tended to merge distinct contractual provisions into larger chunks, whereas higher thresholds often produced fragmented chunks with insufficient context. Chunk sizes are constrained between 400 and 500 tokens to provide sufficient contractual context while remaining within the effective retrieval and context-processing limits of downstream LLMs. In addition, the final two sentences of each chunk are preserved to overlap with the subsequent chunk to maintain contextual continuity across chunk boundaries. The resulting process generates a sequence of semantically coherent contract chunks, $C = \{c_1, c_2, \ldots, c_n\}$, which subsequently serve as the nodes of the graph.

Each chunk is enriched with three complementary representations. First, a dense semantic embedding h_i is generated using the all-MiniLM-L6-v2 Sentence Transformer model. Second, named entities are extracted using the Flair OntoNotes Fast NER model (Akbik et al., 2019). The model identifies diverse entity categories, including persons, organizations, locations, legal references, dates, temporal expressions, and numerical entities. The resulting entity set for chunk c_i is denoted by E_i. Third, contractual keywords are extracted using KeyBERT (Grootendorst, 2020) with unigram phrases (n-gram range = 1). For each chunk, the top-ranked keywords are retained to represent salient contractual concepts. These keywords capture important contractual topics such as confidentiality, indemnification, assignment, renewal, insurance, and limitation of liability. The resulting keyword set is denoted by K_i.

Each graph node is therefore represented as: $v_i = (c_i, h_i, E_i, K_i)$, where c_i denotes the chunk text, h_i its semantic embedding, E_i its entity set, and K_i its keyword set. The contract is modeled as a contract-specific knowledge graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents semantically coherent contract chunks and E captures the semantic, entity-level, conceptual, and structural relationships between them. In the proposed formulation, each chunk is connected to every other chunk within the contract to preserve potential semantic, entity-level, conceptual, and structural dependencies among contractual provisions. This design choice is motivated by the observation that contractually related provisions may be distributed across distant sections of a document and are not necessarily linked through local structure or explicit references. By preserving all pairwise clause relationships, the graph captures the full spectrum

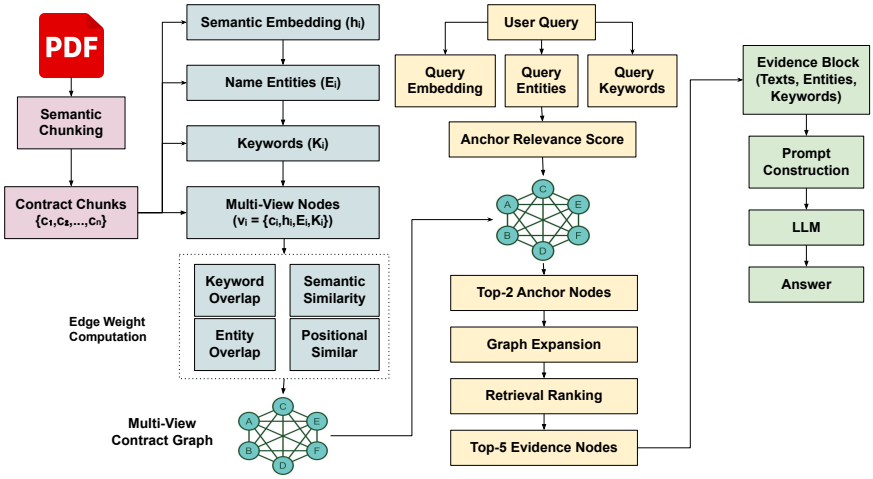


Figure 1: Overview of the proposed Multi-View Contract Graph (MCG).

of potential semantic, entity-level, conceptual, and structural interactions within the contract. Consequently, potentially relevant supporting clauses are not excluded during graph construction, allowing retrieval to leverage both directly relevant clauses and related supporting context. The strength of the relationship between two nodes is determined using four complementary signals. Semantic similarity is computed as: $S_{\text{sem}}(i,j) = \frac{h_i \cdot h_j}{\|h_i\| \times \|h_j\|}$. Entity overlap is measured using Jaccard similarity: $S_{\text{ent}}(i,j) = \frac{|E_i \cap E_j|}{|E_i \cup E_j|}$. Keyword overlap is similarly computed as: $S_{\text{key}}(i,j) = \frac{|K_i \cap K_j|}{|K_i \cup K_j|}$. To capture structural proximity within the document, positional similarity is defined as: $S_{\text{pos}}(i,j) = \frac{1}{|i-j|}$.

Since graph edges are constructed only between distinct nodes ($i \neq j$), self-connections are not included in the graph and the denominator never becomes zero. Consequently, the positional similarity measure remains well-defined for all valid node pairs.

The four signals are combined into a single edge weight: $EW(i,j) = \alpha \cdot S_{\text{sem}}(i,j) + \beta \cdot S_{\text{ent}}(i,j) + \gamma \cdot S_{\text{key}}(i,j) + \delta \cdot S_{\text{pos}}(i,j)$, where α, β, γ, and δ control the relative contributions of semantic similarity, entity overlap, keyword overlap, and positional proximity, respectively. In our implementation, all four coefficients are assigned equal weights ($\alpha = \beta = \gamma = \delta = 0.25$). The resulting graph captures semantic, entity-level, conceptual, and structural relationships among contractual clauses, enabling retrieval beyond direct semantic similarity.

3.3 Graph-Based Retrieval and Answer Generation

Given a user query q, MCG first computes a dense query embedding using the same sentence embedding model employed during graph construction. In addition, lightweight keyword and entity representations are derived from the query text.

To identify relevant entry points into the graph, each node is assigned an anchor relevance score based on its correspondence with the query. The anchor relevance score is computed as: $L(i) = w_1 \times S_q(i) + w_2 \times S_{key}(i) + w_3 \times S_{ent}(i)$, where $S_q(i)$ denotes the cosine similarity between the query embedding and the embedding of node i, while $S_{key}(i)$ and $S_{ent}(i)$ represent the keyword-overlap and entity-overlap scores between the query and node i, respectively. In our implementation, equal weights are assigned to all three components ($w_1 = w_2 = w_3 = 0.33$).

The highest-scoring nodes are selected as anchor nodes, representing the clauses most directly relevant to the user query. Rather than restricting retrieval to these anchor clauses alone, MCG expands retrieval by incorporating graph-connected clauses that exhibit strong relationships with the anchors. For each candidate node i connected to an anchor node a, a retrieval score is computed as: $R(i) = \lambda \times L(a) + (1 - \lambda) \times EW(a,i)$, where $L(a)$ denotes the relevance score of anchor node a, $EW(a,i)$ denotes the edge weight between anchor node a and candidate node i, and λ controls the relative influence of anchor relevance and graph connectivity. In our implementation, equal weighting is adopted ($\lambda = 0.5$). Candidate nodes are

subsequently ranked according to $R(i)$, enabling retrieval to consider both direct query relevance and the multi-view relationships captured within the contract graph. Consequently, clauses that may not be highly similar to the query can still be retrieved when they share strong semantic, entity-level, conceptual, or structural connections with highly relevant anchor clauses.

In our implementation, the two highest-scoring nodes are selected as anchor nodes. Neighboring graph-connected clauses are ranked according to their anchor relevance and edge-weight strength, and the final retrieval context consists of the five highest-ranked nodes obtained from the combined anchor-selection and graph-expansion process.

The retrieved nodes are transformed into evidence blocks containing both the original contract text and their associated metadata, including extracted named entities and contractual keywords. These evidence blocks are then provided to a downstream LLM together with an extraction-oriented prompt that instructs the model to generate answers using only the retrieved evidence. The final answer is therefore grounded in both the retrieved contractual clauses and the contextual relationships captured by the Multi-View Contract Graph, reducing unsupported generation while preserving supporting contractual context that may not be recovered through similarity-based retrieval alone.

4 Dataset and Experimentation Details

4.1 CUAD Dataset

We evaluate the proposed framework using contracts from the Contract Understanding Atticus Dataset (CUAD) (Hendrycks et al., 2021), a benchmark corpus containing over 500 commercial contracts annotated across 41 legal clause categories. The dataset includes diverse agreement types and contractual provisions, making it a representative benchmark for contract understanding and information extraction.

4.2 Task Formulation and Evaluation Setting

Our evaluation differs from the original CUAD benchmark, which formulates contract review as an extractive span-detection task and evaluates performance using metrics such as AUPR and Precision at fixed recall levels. In contrast, we reformulate contract analysis as a retrieval-augmented question-answering task for contract information extraction. Given a natural-language question corresponding to a contractual category, the system retrieves relevant contractual evidence and provides it to an LLM, which generates the final answer. From the original 41 CUAD categories, we evaluate 15 categories (Table 1) covering both straightforward contractual attributes (e.g., parties, dates, and governing law) and more complex provisions (e.g., exclusivity, non-compete, liability, and audit rights). Consequently, our evaluation measures the effectiveness of retrieval-augmented answer generation rather than supervised clause-span classification.

4.3 Baseline Methods

To evaluate the effectiveness of the proposed MCG, we compare against several commonly used retrieval and question-answering baselines: (1) **Extractive QA**: Pre-trained transformer-based question-answering models (Özkurt, 2024) (DistilBERT-QA, RoBERTa-QA, and BERT-QA) without CUAD-specific fine-tuning. This baseline evaluates conventional span-extraction approaches for contract information extraction and provides a retrieval-free benchmark for comparison with retrieval-based methods. (2) **BM25-RAG** (Robertson and Zaragoza, 2009): A retrieval-augmented generation framework that employs BM25 as the retriever. BM25 retrieves contract chunks based on lexical matching, and the retrieved context is supplied to an LLM for answer generation. (3) **Dense-Retrieval-RAG** (Lewis et al., 2020): A conventional retrieval-augmented generation framework that retrieves contract chunks using dense embedding similarity and supplies the retrieved context directly to an LLM for answer generation. (4) **ColBERT-RAG** (Khattab and Zaharia, 2020): A retrieval-augmented generation framework that employs ColBERT as the retriever. Retrieved contract chunks are ranked using late-interaction semantic matching and provided to an LLM for answer generation.

For answer generation, the BM25-RAG, ColBERT-RAG, Dense-Retrieval-RAG, and MCG retrieval frameworks were each evaluated using five open-source LLMs: Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), Llama-3-8B-Instruct (Grattafiori et al., 2024), Saul-7B-Instruct-v1 (Colombo et al., 2024), and Qwen-7B-Chat (Bai et al., 2023). All methods were evaluated using the same set of contractual questions, retrieval settings, and evaluation protocols to ensure a fair comparison.

4.4 Evaluation Metrics

Since our task is formulated as retrieval-augmented answer generation rather than span detection, the evaluation metrics differ from those used in the original CUAD benchmark. For each contractual category, generated answers were compared against reference answers using Precision, Recall, and F_1 Score. To assess semantic similarity, we additionally report BERTScore-P, BERTScore-R, and BERTScore-F_1 (Zhang et al., 2019), which compare generated and reference answers using contextualized language model embeddings. Together, these metrics evaluate both lexical accuracy and semantic faithfulness.

5 Results and Discussion

5.1 Main Results

Table 2 presents the overall performance of Extractive QA, BM25-RAG, ColBERT-RAG, Dense-Retrieval-RAG, and the proposed MCG across five open-source LLMs. The best overall results were obtained by MCG combined with Mistral-7B-Instruct-v0.3, achieving a Precision of 0.458, Recall of 0.471, F_1 score of 0.460, and BERTScore-F_1 of 0.779. Compared with conventional Dense-Retrieval-RAG, MCG improves F_1 by around 17.9%, 12.7%, 13.8%, 16.0%, and 10.8% for Mistral-7B-Instruct-v0.2, Mistral-7B-Instruct-v0.3, Saul-7B-Instruct-v1, Qwen-7B-Chat, and Llama-3-8B-Instruct, respectively. Similar improvements are observed across BERTScore metrics, indicating that graph-enhanced retrieval improves both answer accuracy and semantic faithfulness.

Among the other retrieval-based baselines, BM25-RAG and ColBERT-RAG generally perform competitively and often slightly outperform Dense-Retrieval-RAG. However, MCG achieved the highest F_1 score for four of the five evaluated LLMs and attains the best overall F_1 score of 0.460 with Mistral-7B-Instruct-v0.3. For the same model, BM25-RAG and ColBERT-RAG achieve F_1 scores of 0.418 and 0.410, respectively, while MCG achieves 0.460, corresponding to relative improvements of approximately 10.0% and 12.2%. These results suggest that direct query-to-chunk similarity alone may be insufficient for contract understanding tasks. By incorporating semantic, entity, keyword, and structural relationships between clauses, MCG is able to recover supporting contractual context that is often overlooked by traditional retrieval methods.

Retrieval-based approaches also substantially outperform conventional extractive QA models. The strongest extractive baseline, BERT-QA, achieved an F_1 score of 0.261, whereas the best MCG configuration achieved 0.460, representing an absolute improvement of 19.9%. This result highlights the limitations of retrieval-free span extraction for contract information extraction, where relevant evidence is frequently distributed across multiple clauses and sections of a contract.

5.2 Ablation Study

To assess the contribution of each relationship signal, we perform a leave-one-component-out ablation study using the best-performing configuration (MCG + Mistral-7B-Instruct-v0.3). In each variant, one relationship component is removed from the edge-weight computation while keeping the retrieval procedure unchanged. The *No Semantic Edge*, *No Entity Edge*, *No Keyword Edge*, and *No Positional Edge* variants remove semantic similarity, entity overlap, keyword overlap, and positional proximity, respectively. Results are summarized in Table 3. Removing semantic relationships causes the largest performance drop, reducing F_1 from 0.460 to 0.409 ($\Delta F_1 = -0.051$), followed by entity relationships (0.438, $\Delta F_1 = -0.022$), keyword relationships (0.444, $\Delta F_1 = -0.016$), and positional relationships (0.451, $\Delta F_1 = -0.009$).

Overall, all four relationship signals contribute to retrieval effectiveness, with semantic similarity providing the strongest contribution. The highest performance is achieved when all four signals are combined, supporting the effectiveness of multi-view clause relationship modeling for contract retrieval and information extraction.

6 Limitations and Future Work

Our framework has three main limitations: (1) **Scalability**: MCG involves $O(n^2)$ pairwise relationship computation, which can be costly for large contracts. Future work will explore graph sparsification, approximate nearest-neighbor search, and hierarchical retrieval. (2) **Graph Construction**: MCG relies on manually designed semantic, entity, keyword, and positional relations and the quality of extracted entities and keywords. Future work will investigate learnable graph construction and adaptive edge weighting. (3) **Limited Evaluation**: MCG is evaluated on 15 CUAD categories, which may not capture the diversity of real-world legal

Table 2: Overall Performance Comparison on CUAD.

Method	Model	Precision	Recall	F_1	BERTScore-P	BERTScore-R	BERTScore-F_1
Extractive QA	DistilBERT-QA	0.284	0.187	0.227	0.641	0.582	0.613
Extractive QA	RoBERTa-QA	0.349	0.206	0.253	0.651	0.575	0.610
Extractive QA	BERT-QA	0.344	0.218	0.261	0.677	0.601	0.636
BM25-RAG	Mistral-7B-Instruct-v0.2	0.352	0.413	0.376	0.685	0.704	0.695
BM25-RAG	Mistral-7B-Instruct-v0.3	0.390	0.463	0.418	0.704	0.713	0.707
BM25-RAG	Saul-7B-Instruct-v1	0.355	0.437	0.376	0.694	0.732	0.709
BM25-RAG	Qwen-7B-Chat	0.281	0.523	0.362	0.664	0.727	0.692
BM25-RAG	Llama-3-8B-Instruct	0.262	0.553	0.351	0.613	0.706	0.657
ColBERT-RAG	Mistral-7B-Instruct-v0.2	0.370	0.443	0.397	0.694	0.710	0.702
ColBERT-RAG	Mistral-7B-Instruct-v0.3	0.372	0.469	0.410	0.711	0.723	0.716
ColBERT-RAG	Saul-7B-Instruct-v1	0.319	0.461	0.372	0.683	0.734	0.708
ColBERT-RAG	Qwen-7B-Chat	0.323	0.528	0.395	0.679	0.738	0.705
ColBERT-RAG	Llama-3-8B-Instruct	0.285	0.551	0.372	0.626	0.712	0.668
Dense-Retrieval-RAG	Mistral-7B-Instruct-v0.2	0.341	0.413	0.369	0.681	0.702	0.692
Dense-Retrieval-RAG	Mistral-7B-Instruct-v0.3	0.382	0.454	0.408	0.696	0.709	0.702
Dense-Retrieval-RAG	Saul-7B-Instruct-v1	0.345	0.420	0.377	0.686	0.726	0.707
Dense-Retrieval-RAG	Qwen-7B-Chat	0.284	0.499	0.357	0.663	0.725	0.690
Dense-Retrieval-RAG	Llama-3-8B-Instruct	0.247	0.529	0.334	0.601	0.702	0.648
Proposed Method (MCG)	Mistral-7B-Instruct-v0.2	0.424	0.455	0.435	0.763	0.767	0.765
Proposed Method (MCG)	Mistral-7B-Instruct-v0.3	0.458	0.471	0.460	0.791	0.773	0.779
Proposed Method (MCG)	Saul-7B-Instruct-v1	0.355	0.548	0.429	0.718	0.760	0.741
Proposed Method (MCG)	Qwen-7B-Chat	0.345	0.528	0.414	0.720	0.755	0.736
Proposed Method (MCG)	Llama-3-8B-Instruct	0.301	0.500	0.370	0.711	0.754	0.733

Table 3: Ablation Study of Graph Construction Components.

Variant	F_1	ΔF_1
MCG (Sem + Ent + Key + Pos)	**0.460**	–
No Semantic Edge	0.409	-0.051
No Entity Edge	0.438	-0.022
No Keyword Edge	0.444	-0.016
No Positional Edge	0.451	-0.009

documents. Future work will extend evaluation to larger and more diverse legal datasets.

7 Data and Code Availability

All datasets used in this work are publicly available[1]. The code is provided in a public repository[2].

8 Conclusion

This paper presented the Multi-View Contract Graph (MCG), a contract-specific knowledge graph and graph-based retrieval framework for contract question answering and information extraction. By modeling semantic, entity, keyword, and structural relationships among contract clauses, MCG extends retrieval beyond directly relevant clauses and recovers supporting contractual context that may be overlooked by conventional retrieval methods. Experimental results on the CUAD dataset across 15 contract information categories and five open-source LLMs demonstrate the effectiveness of the proposed approach. MCG achieved the strongest overall performance among the evaluated methods, attaining a best F_1 score of 0.460 and a BERTScore-F_1 of 0.779 with Mistral-7B-Instruct-v0.3. For the best-performing configuration, MCG improved F_1 by approximately 10.0% and 12.2% over BM25-RAG and ColBERT-RAG, respectively. Furthermore, the ablation study showed that all relationship signals contribute to retrieval effectiveness, with semantic relationships providing the largest contribution. Overall, the findings demonstrate that explicitly modeling inter-clause relationships improves evidence of retrieval and downstream contract information extraction. The proposed MCG framework provides a promising foundation for future legal AI systems supporting contract review, legal search, compliance analysis, and contract intelligence applications.

Generative AI Usage Declaration

The authors declare that no Generative AI or Large Language Models were used in the writing, drafting, or content generation of this manuscript.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019.

[1] https://www.atticusprojectai.org/cuad/
[2] https://github.com/mb16biswas/MCG/

Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)*, pages 54–59.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Ryan Calvin Barron, Maksim Eren, Olga Serafimova, Cynthia Matuszek, and Boian Alexandrov. 2025. Bridging legal knowledge and ai: Retrieval-augmented generation with vector stores, knowledge graphs, and hierarchical non-negative matrix factorization. In *Proceedings of the Twentieth International Conference on Artificial Intelligence and Law*, pages 51–60.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. In *Findings of the association for computational linguistics: EMNLP 2020*, pages 2898–2904.

Yongming Chen, Miner Chen, Ye Zhu, Juan Pei, Siyu Chen, Yu Zhou, Yi Wang, Yifan Zhou, Hao Li, and Songan Zhang. 2026. Leverage knowledge graph and large language model for law article recommendation: A case study of chinese criminal law. *Journal of Web Semantics*, page 100879.

Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre FT Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, et al. 2024. Saullm-7b: A pioneering large language model for law. *arXiv preprint arXiv:2403.03883*.

Claudia dAmato, Giuseppe Rubini, Francesco Didio, Donato Francioso, Fatima Zahra Amara, and Nicola Fanizzi. 2025. Automated creation of the legal knowledge graph addressing legislation on violence against women: resource, methodology and lessons learned. *arXiv preprint arXiv:2508.06368*.

Erwin Filtz, Sabrina Kirrane, and Axel Polleres. 2021. The linked legal data landscape: linking legal data across different countries. *Artificial Intelligence and Law*, 29(4):485–539.

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*.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review. *arXiv preprint arXiv:2103.06268*.

Sarika Jain, Pooja Harde, Nandana Mihindukulasooriya, Sudipto Gosh, Ankush Bisht, and Abhinav Dubey. 2022. Constructing a knowledge graph from indian legal domain corpus. In *TEXT2KG/MK@ ESWC*, pages 80–93.

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.

Mrs Junior, Barigye Brighton, and Avance University. 2026. Automated contract analysis and the reduction of legal operational costs.

Md Nazrul Islam Khan. 2022. A systematic review of legal technology adoption in contract management, data governance, and compliance monitoring. *American Journal of Interdisciplinary Studies*, 3(01):01–30.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

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.

Jun Li, Lu Qian, Peifeng Liu, and Taoxiong Liu. 2024. Construction of legal knowledge graph based on knowledge-enhanced large language models. *Information*, 15(11):666.

Shuang Liu, Zelong Li, Ruoyun Ma, Haiyan Zhao, and Mengnan Du. 2025. Contracteval: Benchmarking llms for clause-level legal risk identification in commercial contracts. *arXiv preprint arXiv:2508.03080*.

Cem Özkurt. 2024. Comparative analysis of state-of-the-art q&a models: Bert, roberta, distilbert, and albert on squad v2 dataset. *Chaos and Fractals*, 1(1):19–30.

Vuong T Pham, Dang V Dung, Hung Q Ngo, Binh Nguyen, Nhon V Do, Ali Selamat, and Hien D Nguyen. 2026. Ontology-based knowledge graph approach for legal queries. *Informatica*, 37(1):159–192.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174.

Julián Moreno Schneider, Georg Rehm, Elena Montiel-Ponsoda, Víctor Rodríguez-Doncel, Patricia Martín-Chozas, María Navas-Loro, Martin Kaltenböck, Artem Revenko, Sotirios Karampatakis, Christian Sageder, et al. 2022. Lynx: A knowledge-based ai service platform for content processing, enrichment and analysis for the legal domain. *Information Systems*, 106:101966.

Amrita Singh, Aditya Joshi, Jiaojiao Jiang, and Hye-young Paik. 2025. A survey of classification tasks and approaches for legal contracts. *Artificial Intelligence Review*, 58(12):380.

Amrita Singh, H Suhan Karaca, Aditya Joshi, Hye-young Paik, and Jiaojiao Jiang. 2026. Evaluating customized vs. generalist transformer-based models for legal contract classification. In *Proceedings of the Second Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)*, pages 44–54.

Francesco Sovrano, Monica Palmirani, and Fabio Vitali. 2020. Legal knowledge extraction for knowledge graph based question-answering. In *Legal Knowledge and Information Systems: JURIX 2020: The Thirty-third Annual Conference, Brno, Czech Republic, December 9–11, 2020*, pages 143–153. SAGE Publications Pvt. Ltd 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Steven H Wang, Maksim Zubkov, Kexin Fan, Sarah Harrell, Yuyang Sun, Wei Chen, Andreas Plesner, and Roger Wattenhofer. 2025. Acord: An expert-annotated retrieval dataset for legal contract drafting. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24739–24762.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Lucia Zheng, Neel Guha, Brandon R Anderson, Peter Henderson, and Daniel E Ho. 2021. When does pretraining help? assessing self-supervised learning for law and the casehold dataset of 53,000+ legal holdings. In *Proceedings of the eighteenth international conference on artificial intelligence and law*, pages 159–168.

BD-AgriKG: A Knowledge Graph and Bilingual Text2Cypher Benchmark for Crop Suitability and Productivity in Bangladesh

Md. Aukerul Moin Shuvo **Md. Nazrul Islam Mondal**
Department of Computer Science & Engineering
Rajshahi University of Engineering & Technology, Rajshahi-6204, Bangladesh
2204103003@student.ruet.ac.bd, mondal@cse.ruet.ac.bd

Abstract

Agricultural advisory in Bangladesh rests on public statistics locked in PDF yearbooks, and on language technology that rarely speaks Bangla. We present BD-AgriKG, a knowledge graph that integrates five government sources on one administrative spine keyed by standard place codes: upazila-level land suitability for 16 crops, agro-meteorological thresholds, crop calendars, soil associations, and a newly extracted district production panel covering twelve crop years, validated against the yearbooks' own published totals. On top of the graph we release what is, to our knowledge, the first bilingual (English and Bangla) execution-based Text2Cypher benchmark: 460 questions instantiating 26 templates, built Cypher-first so that every item carries an executable gold query and a non-empty executed answer. We report execution-accuracy baselines for seven large language models (LLMs) in both languages. The best model reaches 95.2% in English and 90.9% in Bangla under a strict exact-projection metric whose frontier failures are almost all extra returned columns. Every model scores lower in Bangla, by 1 to 14 points; after multiple-comparison correction the gap is significant for three of the seven models under a template-clustered bootstrap and for one under template-level tests, and for four models, both frontier models among them, two question templates carry most of it. One turn of execution feedback, the simplest use of the graph's own execution signal, recovers 43–73% of the invalid and empty failures of mid-tier models and helps Bangla at least as much as English, though not significantly more, while small open models recover almost nothing. The graph, panel, benchmark and harness are released.

1 Introduction

Bangladesh's agricultural extension workers answer district-level what, where and when questions

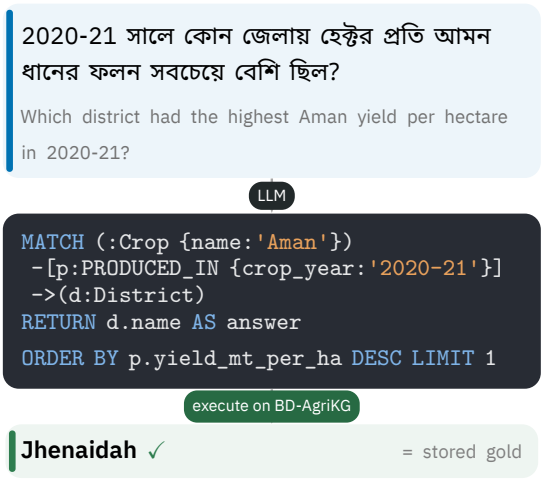


Figure 1: A released benchmark item end to end: the Bangla question and its English twin, a model-generated Cypher query, and read-only execution against the graph, graded against the stored gold result.

every day: which upazila suits a crop, when to sow, what a district produced last year, which pest threatens the main crop. The answers exist in public statistics, land-suitability zoning, crop calendars and soil surveys, but they sit in PDF yearbooks and scattered agency portals rather than in any queryable form. Large language models (LLMs) answer such questions fluently and, on exactly the facts that matter (planting dates, seed rates, input doses), often wrongly (Ibrahim et al., 2024); a recent Bangla agricultural benchmark finds that even retrieval-augmented systems keep a residual hallucination floor on safety-critical values (Reza et al., 2026). Grounding generation in a structured knowledge graph (KG) is the obvious remedy, but no Text2Cypher or KG question answering (KGQA) resource exists in Bangla at all, and the existing Text2Cypher benchmarks are synthetic and English-only (Ozsoy et al., 2025; Ozsoy and Tai, 2025). The measurement matters beyond one-shot question answering (QA). In agentic systems the graph serves as a verifiable tool environment

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 63–76
November 10, 2026 ©2026 Association for Computational Linguistics

in which agents interleave queries with reasoning (Shen et al., 2025; Yang et al., 2024), so the question-to-query step we isolate is the primitive every such action rests on, and execution feedback is a signal the agent itself can observe. Figure 1 shows one released benchmark item end to end.

We make four contributions: (1) BD-AgriKG, a P-code-keyed KG over five public agricultural sources including a 12-year district production panel extracted from PDF yearbooks with checksum validation; (2) the first bilingual execution-based Text2Cypher benchmark (460 items, Cypher-first); (3) execution-accuracy baselines in English and Bangla, an analysis of the Bangla gap, and a one-turn execution-feedback repair experiment; (4) full release of data, extraction pipeline and harness.

2 Related work

Agricultural knowledge graphs. Agricultural KGs are usually extracted from text (Chen et al., 2019). The closest resource to ours is BDAKG (Deb Nath et al., 2024), a Resource Description Framework (RDF) graph of Bangladesh Bureau of Statistics (BBS) data built for online analytical processing (OLAP); it carries production statistics but no land suitability, crop calendar, agrometeorological or pest data, and it has no natural-language layer. BD-AgriKG differs in both respects: it integrates five sources on one administrative spine and is built to be queried in natural language.

GraphRAG in agriculture. Recent graph-based retrieval-augmented generation (GraphRAG) systems pair Neo4j graphs with LLMs for agricultural QA: Crop GraphRAG for pests and diseases (Wu et al., 2025) and a garlic-cultivation assistant (Zhou et al., 2026), both building their graphs by LLM extraction from text and evaluating end-answer accuracy, in Chinese. ShizishanGPT couples an agri LLM with a KG module and tools (Yang et al., 2024). The general GraphRAG line (Edge et al., 2024) likewise assumes an LLM-extracted graph. Our graph is curated from statistical tables, and our evaluation isolates the query-translation step with execution accuracy.

Bangla agricultural NLP. Bangla advisory systems retrieve from documents rather than a graph: KrishokBondhu answers voice queries over scanned handbooks digitised by optical character recognition (OCR) (Ameen et al., 2025), and Reza et al. (2026) release a provenance-traceable Bangla benchmark showing residual chemical-dosage hallucination even with oracle evidence. General-purpose Bangla resources exist for exam QA (Shafayat et al., 2024) and automatic KG construction (Wasi et al., 2024), but none grounds answers in a structured graph. The closest resource is concurrent work: IndicDB (Dawar et al., 2026) evaluates execution-based Text2SQL over 20 relational databases of Indian open data in seven language variants including Bengali, reporting Bengali at 56.2% against 64.7% for English. That 8.5-point Bengali penalty on relational schemas is of the same order as the gaps we measure on a property graph. No Bangla Text2Cypher resource exists.

Text2Cypher. The Neo4j Text2Cypher dataset (Ozsoy et al., 2025) and its multilingual extension (English, Spanish, Turkish; no Bangla) (Ozsoy and Tai, 2025) train and evaluate mostly on synthetic schemas, with string-similarity metrics and, on the subset with live database access, execution-based exact match; CypherBench (Feng et al., 2024) argues for execution-based evaluation on full-scale graphs, which we adopt, following the Text2SQL practice of Spider and BIRD (Yu et al., 2018; Li et al., 2023). Multilingual semantic parsing has been benchmarked for SQL across seven languages, none of them Bangla (Dou et al., 2023), and multilingual KGQA datasets cover nine languages each, again without Bangla (Perevalov et al., 2022; Sen et al., 2022). Translating an English benchmark while keeping its Cypher, as Ozsoy and Tai (2025) do for Spanish and Turkish, would give a larger Bangla set cheaply, but over schemas unrelated to Bangladesh; it would complement a benchmark on a real Bangladeshi graph rather than replace it.

Repair from execution feedback. Feeding execution results back to the generator is an established remedy in code generation (Chen et al., 2024) and Text2SQL, where DIN-SQL adds a self-correction step (Pourreza and Rafiei, 2023) and MAC-SQL a refiner agent that rewrites queries on SQL errors (Wang et al., 2025). We apply the minimal one-turn version of this loop to Text2Cypher and, to our knowledge for the first time, measure it per language.

The gap. Taken together, these lines leave three openings. No agricultural knowledge graph in-

tegrates Bangladeshi land suitability, crop calendars and production statistics on one administrative spine with a natural-language layer. No Bangla resource grounds question answering in a structured graph, and the only Bangla structured-query data targets relational rather than graph schemas. And no Text2Cypher benchmark covers Bangla or a curated government-statistics graph. BD-AgriKG addresses all three: to our knowledge it is the first execution-based Text2Cypher benchmark in Bangla and the first on a real government-statistics property graph.

3 BD-AgriKG

3.1 Administrative spine and entity resolution

Figure 2 summarises how the graph is built. All administrative units are keyed by P-codes from version 3 of the UN OCHA Common Operational Datasets on administrative boundaries (COD-AB). The 466 crop-zoning units of the Bangladesh Agricultural Research Council (BARC) were resolved onto the 507 admin-3 units (upazilas) in four ways: 310 matched exactly after normalisation, 55 through *sadar* naming conventions (the upazila that carries its district's name), 82 through fuzzy matches that were each audited by hand, and 21 manually. Units that merged or split across vintages are handled explicitly (two BARC units map to two admin units each, yielding 468 links), including 11 upazila names that occur in two or three different districts and would silently collide under name-based keys. We have no held-out estimate of resolution accuracy; each of the 468 links is released with the method that produced it.

3.2 Sources and graph shape

The five sources are the BARC crop-suitability zoning, the BBS Yearbook of Agricultural Statistics (production), the BBS crop calendar, the agro-meteorological crop thresholds of the Bangladesh Agro-Meteorological Information Service (BAMIS), and the soil associations of the Soil Resource Development Institute (SRDI). Together they yield a graph of 650 nodes and 21,212 relationships; Appendix A lists every label, relationship type and property. Each of the 7,456 SUITABLE_IN edges links a crop to an upazila and carries the five BARC suitability-class areas together with a derived suitability factor, which is left null where the source area is zero. Each of the 11,632 PRODUCED_IN edges links a crop to a district for one crop year and carries area, production and derived yield, with provenance and a quality flag; rows with zero or missing production stay in the released panel but are not loaded into the graph. The 1,471 HAS_SOIL edges attach soil associations to upazilas, the 82 AFFECTED_BY edges attach pests and diseases to crops, IN_DISTRICT and IN_DIVISION edges carry the administrative hierarchy, and the crop calendar and agro-meteorological thresholds complete the graph as node properties.

3.3 The production panel

The production panel is extracted from ten yearbook editions. The tables come in two families with two to five value columns per year block; most editions print jute in 400 lb bales, which we convert at 181.44 kg per bale, division subtotals are numbered, and long rows are split across lines. Every extracted table is checked against its own printed division and national totals; disagreements are recorded in the released extraction report rather than gating ingestion, and in practice the cross-edition consensus rule below is what resolves them. Where a crop year appears in several editions, we take the newest value that agrees with the median production within 2%; if the newest edition disagrees, we take the median-agreeing value and flag the row, which rejected 264 newest-edition outliers. Twelve adversarial cell-level spot checks against the source PDFs all matched, but a sample that small does not establish a cell-level error rate, so the checksums remain the primary validation. A further 134 source-level anomalies are flagged rather than corrected.

Post-review audit. After the reviews we extended the validation beyond the checksums: crop years present in two editions were reconciled, per-crop plausibility bands applied, and the calendar and pest sheets re-read against the agencies' publications. This found some extraction defects (a unit and a table-parsing error in single yearbook editions, a change of series for chili, a consensus rule that preferred the newer edition, and copy-paste errors in the pest and calendar sheets), all corrected in the released data (v1.1) and listed in its changelog. A minority of gold answers change between v1.0 and v1.1; the changelog lists them. Every result in this paper is on v1.0, which is released beside v1.1; since grading is by execution,

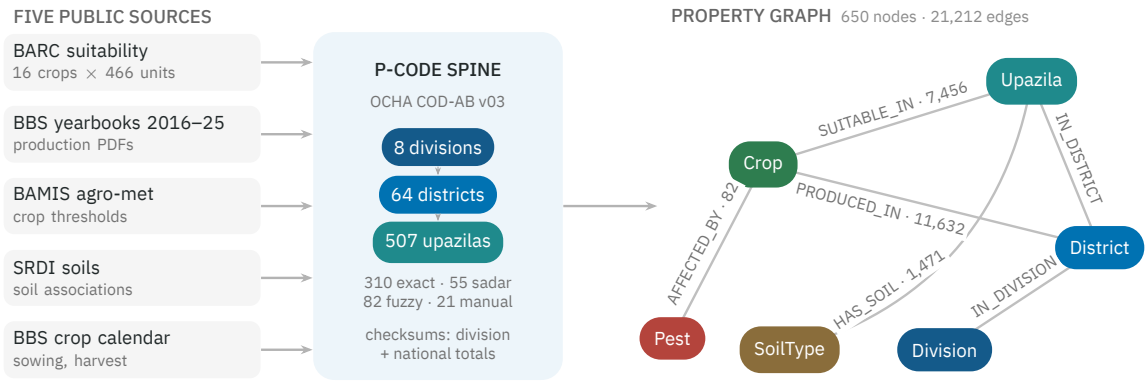


Figure 2: BD-AgriKG construction. Five government sources are resolved onto one P-code administrative spine; every extracted production table is checked against its own printed division and national totals (disagreements are logged; Section 3.3), plus 12/12 cell-level PDF spot checks.

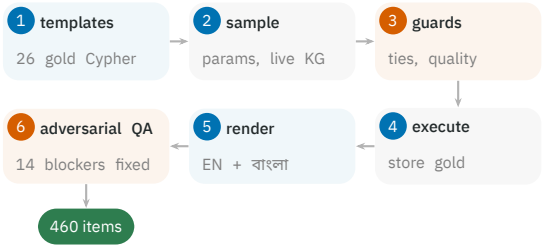


Figure 3: Benchmark construction. Questions are rendered from executed gold queries, never the reverse, so every item is answerable; sampler guards reject ranking ties and quality-flagged data.

a model's verdict measures whether its query retrieves what the graph holds.

4 Benchmark

Construction. The benchmark is built Cypher-first over the live graph (Figure 3). Each of the 26 templates pairs a parameterised gold query with an English and a Bangla question pattern. Parameters are sampled from the graph itself, the gold query is executed and its result stored, and items whose answer is degenerate (empty, tied or resting on quality-flagged data) are rejected. The result is 460 items in six categories: 118 lookup, 90 aggregation, 90 ranking, 72 multi-hop, 54 temporal and 36 comparison. Three readings that the questions leave open are fixed by the gold queries: *most suitable* ranks by `suitability_factor`; *highly suitable* means a factor above 0.7, which the prompt states (Figure 5); and the *top crop* of a district is the one with the largest production in tonnes, so high-tonnage crops such as sugarcane often win. A model that ranks suitability by the very-suitable area instead is graded wrong; across the fourteen runs, 4 of the 160 failing queries that execute on the two suitability templates return that reading, so the ambiguity costs little.

The Bangla side. Bangla questions have Bangla syntax and Bangla crop names but keep three classes of literal in Latin script, namely place names (252 items), crop years (216) and pest or disease names (54), the forms stored in the graph (Limitations discusses this choice). The Bangla templates were LLM-drafted under author supervision and checked for morphology by the Bangla-phrasing agent described below; Appendix B lists all 26 pairs. After the review period the first author, a native speaker, rated every template for fluency and for equivalence with its English twin: 22 are natural and 4 acceptable, none unnatural; 25 ask exactly what the English asks, while *seed req* differs, since the Bangla asks only for seed and the English *planting material* can be read as planting equipment. For *water req* a more natural phrasing was suggested. No template was changed, since the evaluation used them as released; the suggested rewordings are recorded in the released rating sheet.

Quality assurance. As a cross-family adequacy check, all 460 Bangla items were back-translated by Pixtral Large and judged against the English original by Llama 3.3 70B: 422 (92%) were judged equivalent, and the 38 flags, in six templates, are translator errors or idiomatic differences (e.g. *Aman* vs. *Aman rice*) that do not change what is asked (Appendix B). Quality assurance used five independent LLM review agents (Claude family), each with its own adversarial brief: grading exploits, boundary ties, gold-answer audit, Bangla phrasing and source-data cross-checks. Together they attacked the templates and the grader and

surfaced 14 blocking defects, among them grading loopholes, boundary ties, source-data errors and answer leakage. Each defect was confirmed against the source tables before being fixed, the benchmark was then regenerated, and tie and plausibility guards were added to the item sampler.

5 Evaluation

5.1 Protocol

We score execution accuracy. A generated Cypher query runs read-only against the graph with a 15 s timeout, and its result is compared with the stored gold result as a multiset of row values with a relative numeric tolerance of 10^{-6}. Under the strict metric any extra returned column is a failure, write clauses are rejected outright, and models are instructed to return only the values asked for. A gold adapter that replays the reference queries scores 460/460. Because the strict metric fails on any extra returned column, a query that dumps columns to improve its chances cannot pass by construction. All runs are single-pass with a 1,500-token output limit. Non-Anthropic models decode at temperature 0. Anthropic models run without sampling parameters, that is at the API's default temperature, because Claude 5 rejects them and Haiku 4.5 was run the same way for uniformity; their item-level outcomes are therefore not deterministic, and Appendix C measures this for one model with a repeated run. DeepSeek-R1's reasoning trace, which the API returns separately, is discarded before query extraction. Exact model identifiers, run dates and settings are listed in Appendix C. Because the 460 items instantiate 26 templates, we treat templates as the sampling unit. Template-clustered percentile-bootstrap 95% confidence intervals (CIs; 2,000 resamples, no multiple-comparison correction) span ±6–17 points for absolute accuracy, so we do not read small cross-model differences as rankings. The *paired* English–Bangla (EN–BN) gap, computed on identical items, is estimated far more precisely and is the quantity our language claims rest on.

5.2 Baselines

We evaluate seven models on Amazon Bedrock, zero-shot with the schema prompt, covering four vendor families and a large-to-small spread: Claude Opus 5, Sonnet 5 and Haiku 4.5 (one family, three sizes), Llama 3.3 70B and Llama 3.1 8B,

	English			Bangla			
Model	val	str	len	val	str	len	Δ
Opus 5	99.8	**95.2**	99.3	100.0	**90.9**	99.3	+4.3
Sonnet 5	100.0	94.8	99.6	100.0	84.3	97.4	+10.4
DeepSeek-R1	88.0	80.4	81.5	86.7	73.7	77.4	+6.7
Pixtral Large	91.7	78.3	80.0	89.8	64.6	65.4	+13.7
Haiku 4.5	88.3	51.3	52.8	90.0	47.4	50.9	+3.9
Llama 3.3 70B	67.8	34.1	34.1	71.3	28.9	28.9	+5.2
Llama 3.1 8B	85.4	8.9	8.9	86.1	8.0	8.0	+0.9

Table 1: Valid-Cypher rate (val), strict (str) and lenient (len) execution accuracy in percent, zero-shot; Δ is the English-to-Bangla strict gap in points, computed from item counts, so it can differ by 0.1 from the difference of the rounded columns. Opus, Sonnet and Haiku are Claude models. Relative to the reviewed version, the grader now reads a single `collect()` list as rows and matches booleans only to booleans; this changed five DeepSeek-R1 verdicts (two English, three Bangla) and two of its repairs, and no other number.

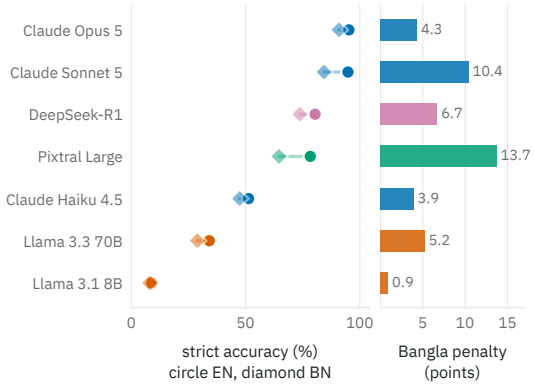


Figure 4: Left: English (circles) and Bangla (diamonds) strict execution accuracy per model. Right: the Bangla penalty isolated, in points; it is largest for Pixtral Large and Sonnet 5 and smallest for Llama 3.1 8B, Haiku 4.5 and Opus 5 (per-category detail in Appendix D).

Pixtral Large 25.02, and the reasoning model DeepSeek-R1. We report the strict metric (exact projection) and a lenient variant that tolerates at most one extra returned value per row, which accepts "winner plus its metric" answers while still rejecting column dumping.

Three findings stand out (Table 1 and Figure 4). First, the benchmark has headroom and spread: the best model solves 19 of 20 English questions while an 8B open model solves fewer than 1 in 10, so zero-shot prompting separates the current model landscape without saturating it (few-shot and fine-tuned baselines are left to future work). Second, every model scores lower in Bangla, but the evidence is of two kinds. The paired template-clustered

95% interval on the EN–BN gap excludes zero for five of the seven models (Opus 5 $[0.2, 10.0]$, Sonnet 5 $[0.9, 21.6]$, DeepSeek-R1 $[1.9, 11.7]$, Pixtral Large $[4.7, 21.1]$, Llama 3.3 70B $[0.7, 13.4]$); it is indistinguishable from zero for the weakest model (Llama 8B, $+0.9$, $[-0.9, 3.7]$) and for Claude Haiku 4.5 ($+3.9$, $[-1.9, 10.5]$). Seven models are seven tests, so we also report the Holm-corrected result, under which three gaps remain significant (DeepSeek-R1, Pixtral Large and Llama 3.3 70B), and two template-level tests, a sign-flip permutation test and a sign test over the 26 templates, under each of which the same three are significant before correction and only Pixtral Large after it (Table 5). The two kinds of evidence separate the models. For Pixtral Large and DeepSeek-R1 the penalty is distributed: 21 and 13 templates favour English, the two largest templates carry 29% and 39% of the gap, and without them the gap is still 10.6 and 4.5 points. For Opus 5, Sonnet 5, Haiku 4.5 and Llama 70B it is concentrated: two templates carry 67–89% of each gap (for Opus 5, *top districts production* and *compare two districts*; for Sonnet 5, *water req of top crop* and *top districts production*; for Llama 70B, mostly *count crops pest*), and without them the gap is 0.5–3.8 points. For these four models the penalty sits in a few templates rather than across the language. With one wording per template we cannot in general separate a template's phrasing from the Bangla construction it uses, but for Llama 70B the wording is visible: the Bangla *count crops pest* question makes the pest its subject where the English is passive, and all 18 of its Bangla queries reverse the AFFECTED_BY edge, against 1 of 18 in English. The gap is not monotone in capability: it is largest for Pixtral Large (13.7 points) and Sonnet 5 (10.4) and smallest for the weakest model (0.9), Haiku 4.5 (3.9) and the strongest (Opus 5, 4.3), though the overlapping intervals do not support an ordering among the gaps. Under the lenient metric the frontier gap nearly vanishes (Opus 5 99.3 in both languages; Sonnet 5 99.6 vs. 97.4), so at the frontier the residual Bangla deficit is mostly output formatting rather than query logic. The three gaps significant under the corrected bootstrap are not mainly formatting: under the lenient metric they are 4.1 points for DeepSeek-R1, 14.6 for Pixtral Large and 5.2 for Llama 3.3 70B. Third, the difference between strict and lenient scores localises failure types. For the open models it is negligible, because their errors are genuinely wrong queries, with hallucinated properties such as `p.crop` or `c.PRODUCED_IN` or missed multi-hop structure; for the frontier models it absorbs almost all errors, which are correct queries with one extra column. Per-category results are shown in Figure 6 (Appendix D) and per-template results in Figure 7 (Appendix E).

Model	fmt	wrong	empty	inv
Opus 5	**6.3**	0.5	0.0	0.1
Sonnet 5	**8.9**	1.5	0.0	0.0
DeepSeek-R1	2.4	6.5	1.4	**12.6**
Pixtral Large	1.3	6.8	**11.2**	9.2
Haiku 4.5	2.5	**31.5**	5.8	10.9
Llama 3.3 70B	0.0	28.5	9.6	**30.4**
Llama 3.1 8B	0.0	28.3	**48.8**	14.5

Table 2: Failure composition as a percentage of all 920 items per model (both languages pooled, so each row sums to the model's pooled failure rate): formatting, wrong result, empty result, invalid Cypher. The dominant mode (bold) shifts from formatting to logic to grounding as capability decreases.

5.3 Error analysis

Re-executing every failed query yields a four-way taxonomy (Table 2): *formatting* (correct answer, extra returned values), *wrong* (executes, non-empty, incorrect), *empty* (executes but matches nothing), and *invalid* (does not execute). Failure modes shift systematically with capability. Frontier models fail almost exclusively by formatting (58 of Claude Opus 5's 64 failures; its single invalid query is a stray colon before `MATCH`). Mid-tier models fail by logic. Claude Haiku 4.5's dominant error is a plausible query that executes and returns the wrong answer. Small open models fail by grounding. Nearly half of Llama 3.1 8B's items execute yet return nothing because they invent schema elements (`c.PRODUCED_IN` as a node property, a `HAS` relationship), which Cypher silently matches to zero rows. DeepSeek-R1 and Llama 3.3 70B additionally produce a large share of invalid Cypher despite the schema being in the prompt.

Why Llama 3.3 70B scores zero on ranking, temporal and multi-hop items. A reviewer asked whether a uniform zero over these 216 items per language could be a pipeline artefact. It is not. In English, 114 of the 216 queries fail to parse and 102 run; of those 102, 89 reference a relationship property through a node variable and the relationship name, as in `ORDER BY c.produced_in.production_mt`. Cypher ac-

cepts this (it reads a property called `produced_in` on the node, which does not exist) and sorts by null, so the query returns an arbitrary district. The Bangla runs show the same idiom in 51 of 114 valid-but-wrong queries. Llama 3.1 8B fails the same items differently: it reverses the PRODUCED_IN edge, so the node it sorts on matches nothing and the query returns no rows (35 English and 22 Bangla queries). Neither error is visible to a syntax check, which is why the taxonomy counts these queries as wrong or empty rather than invalid, and why execution-based grading is necessary here.

5.4 Execution-feedback repair

Because grading is execution-based, the signal that scores a model is also available to an agent at inference time. We test the simplest form of this, a single retry. Every first attempt that was invalid or returned zero rows is retried once: the model receives the same system prompt and question, its previous query, and a fixed sentence carrying either the Neo4j error message (at most 300 characters) or the notice that the query executed but returned zero rows, and is asked for a corrected query under the same rules. Empty results are identified by re-executing the first query with a 6 s timeout; one of the 1,421 retried items timed out at this step and was retried with the timeout message. Wrong-but-non-empty results receive no retry, since no honest signal exists without the gold answer, and no API call failed. Table 3 reports repairable counts, recoveries and strict accuracy after one turn; Appendix F splits them by signal type.

	English			Bangla		
Model	rep.	rec.	acc.	rep.	rec.	acc.
Opus 5	1	1	95.2→95.4	0	0	90.9→90.9
Sonnet 5	0	0	94.8→94.8	0	0	84.3→84.3
Haiku 4.5	72	42	51.3→60.4	81	46	47.4→57.4
DeepSeek-R1	58	37	80.4→88.5	71	52	73.7→85.0
Pixtral Large	74	34	78.3→85.7	114	49	64.6→75.2
Llama 3.3 70B	191	11	34.1→36.5	177	2	28.9→29.3
Llama 3.1 8B	304	1	8.9→9.1	278	0	8.0→8.0

Table 3: One-turn execution-feedback repair. rep.: first attempts that were invalid or returned zero rows (repairable); rec.: recovered to a strictly correct answer; acc.: strict execution accuracy before and after the repair turn (%). Wrong-but-non-empty answers are not retried.

Three observations. First, the ability to use feedback is itself a capability, and it separates the tiers: Claude Haiku 4.5, DeepSeek-R1 and Pixtral Large recover 43–73% of their repairable items, gaining 7–11 points, whereas Llama 3.3 70B recovers 13 of 368 and Llama 3.1 8B one of 582. Within the mid-tier, however, recovery rate does not follow first-pass accuracy (Haiku 58%, Pixtral 44%). The open models largely ignore the message: Llama 70B reproduces the same undefined-variable error in 88 of the 181 retries that carried it, and 189 of its 368 second attempts are still invalid; Llama 8B returns a byte-identical query in 85 of 304 English retries. For Haiku 4.5 the gain is not sampling variance: a repeated Haiku zero-shot run without feedback makes only 10 of the same 72 items correct (Appendix C), against 42 with the error message; 36 items are recovered by feedback only and 4 by resampling only (exact McNemar $p = 1.9 \times 10^{-7}$). We have no such control for DeepSeek-R1 or Pixtral Large. Frontier models have almost nothing to repair (one invalid Opus 5 query, none for Sonnet 5), consistent with their residual errors being formatting rather than grounding. Second, the signal matters (Appendix F, both languages pooled). Haiku fixes 66 of 100 syntax errors but only 22 of 53 empty results, where the notice carries no diagnosis; DeepSeek-R1's failures are almost all syntactic and it fixes 82 of 116; Pixtral shows the opposite profile, repairing 56 of 103 empty results but only 27 of 85 syntax errors, typically by dropping an over-restrictive filter. Third, repair helps Bangla at least as much as English. Bangla first attempts fail more often in repairable ways (81 vs. 72, 71 vs. 58 and 114 vs. 74 items for the three mid-tier models), the Bangla gain exceeds the English gain for all three, and the paired EN–BN gap moves from 13.7 to 10.4 points for Pixtral, 3.9 to 3.0 for Haiku and 6.7 to 3.5 for DeepSeek-R1, whose gap is no longer distinguishable from zero (95% CI $[-1.8, 9.4]$) while Pixtral's remains significant (uncorrected) and Haiku's was not significant before repair either. None of these narrowings is individually significant with 26 template clusters: the 95% intervals on the change are $[-1.9, 8.7]$ for Pixtral, $[-2.2, 8.3]$ for DeepSeek-R1 and $[-2.8, 4.5]$ for Haiku. We therefore read the narrowing as consistent in direction, not as an established effect. Llama 3.3 70B moves the other way (5.2 to 7.2 points) because the few repairs it manages are in English.

6 Conclusion

We released BD-AgriKG, a knowledge graph that makes five scattered Bangladeshi government sources jointly queryable on one administrative spine, together with the first bilingual execution-based Text2Cypher benchmark and baselines for seven LLMs. The benchmark shows a lower Bangla score for every model, significant for three of seven under multiple-comparison control, carried by two templates for four of them and nearly vanishing under lenient grading at the frontier, and an error taxonomy whose dominant failure mode shifts from formatting to logic to schema grounding as capability decreases. The graph, panel, benchmark and harness support two immediate directions: fine-tuning small open models on Bangla Text2Cypher, and generating KG-grounded agricultural reports whose every claim can be verified by executing queries against the same graph. Because grading is execution-based, the benchmark doubles as a verifiable environment for systems that act on execution feedback: one turn recovers 43–73% of the repairable failures of mid-tier models and helps their Bangla at least as much as their English, though not significantly more, while small open models cannot yet use the signal.

Limitations

Suitability reflects 2015-era zoning; production is district-level (no public upazila-level production exists); 20 % of panel cells rest on a single yearbook edition; source misprints are flagged rather than corrected. The benchmark's effective sample size is its 26 templates rather than its 460 items, which our clustered intervals make explicit; each template has one fixed wording and no item is a question asked by a farmer or extension worker, so the results describe these 26 phrasings rather than Bangla Text2Cypher in general; paraphrases and naturally authored questions are future work. Every result is a single run, and the Claude models were sampled at the API's default temperature; the one repeated run (Haiku 4.5, English) moved accuracy by 0.4 points but changed the outcome of 64 of 460 items, so per-template findings carry run-to-run variation that our intervals do not include. Single-value integer answers cluster on a few common values, so a wrong query can occasionally coincide with the right answer. Bangla questions keep place names, years and pest or disease names in Latin script. Code-mixing of this kind is common in Bangladeshi extension material, but several of these pests do have established Bangla names (*Rice Bug* is গান্ধি পোকা, *Aphid* is জাব পোকা), so for the 54 affected items the Latin form is a simplification, and the native-speaker rating of Appendix B judged that an extension worker would write district names, years and pest names in Bangla script as well. Since every Latin-script literal matches the graph verbatim and removes an entity-linking step, fully Bangla surface forms would likely widen the measured gap, an expectation we have not tested. The adversarial QA agents were drawn from a single model family that is also among the systems evaluated; this circularity is mitigated, though not eliminated, by execution-based grading, by manual confirmation of every defect and by the cross-family adequacy check of Appendix B, and the native-speaker rating was made by an author after the review period, not by an independent speaker before evaluation. In particular, Claude-authored phrasing may be easier for Claude models to parse, so comparisons between Claude and non-Claude rows of Table 1 should be read with that caveat. The repair loop is a single turn on failures that carry an execution signal, so wrong-but-non-empty answers are untouched, the only no-feedback control is Haiku 4.5 in English, and the zero-rows notice presumes a non-empty answer exists, which holds for every benchmark item by construction; a deployed agent would need an abstention path. Two baselines we did not run would sharpen the comparison: an existing English Text2Cypher benchmark rendered in Bangla, which would separate the cost of the language from that of this graph's content, and few-shot or fine-tuned models.

Acknowledgements

We thank the three NORA reviewers; their comments led to the corrected significance analysis, the per-template analysis, the Llama 70B inspection and the post-review data audit in this version.

Generative AI use

Generative AI assisted with drafting and revising the manuscript, with literature search, and with parts of the code. The authors reviewed all output and take full responsibility for the content of this paper.

Availability

The graph tables, production panel, benchmark, evaluation harness and all raw per-item model outputs are released at https://github.com/aukerul-shuvo/BD-AgriKG (code under MIT, derived data under CC BY 4.0) and archived with a DOI at https://doi.org/10.5281/zenodo.23203983. The release holds both the v1.0 data on which the results were obtained and the corrected v1.1 data, with a changelog.

References

Muhammad Ibrahim Ameen, Tasnim Islam, Mst. Sanjida Aktar, and Redwan Karim Rafat. 2025. KrishokBondhu: A retrieval-augmented voice-based agricultural advisory call center for Bengali farmers. *Preprint*, arXiv:2510.18355.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2024. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Yuanzhe Chen et al. 2019. AgriKG: An agricultural knowledge graph and its applications. In *Database Systems for Advanced Applications (DASFAA)*, LNCS. Springer.

Aviral Dawar, Roshan Karanth, Vikram Goyal, and Dhruv Kumar. 2026. IndicDB: Benchmarking multilingual text-to-SQL capabilities in Indian languages. *Preprint*, arXiv:2604.13686.

Rudra Pratap Deb Nath, Archita Das, Anika Das, and Mahmudul Raihan. 2024. Knowledge graph generation and enabling multidimensional analytics on Bangladesh agricultural data. *IEEE Access*, 12:87512–87531.

Longxu Dou, Yan Gao, Mingyang Pan, Dingzirui Wang, Wanxiang Che, Dechen Zhan, and Jian-Guang Lou. 2023. MultiSpider: Towards benchmarking multilingual text-to-SQL semantic parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12745–12753.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Yanlin Feng, Simone Papicchio, and Sajjadur Rahman. 2024. CypherBench: Towards precise retrieval over full-scale modern knowledge graphs in the LLM era. *Preprint*, arXiv:2412.18702.

Kadiriye Ibrahim, Kalimuthu Senthilkumar, and Kazuki Saito. 2024. Evaluating responses by ChatGPT to farmers' questions on irrigated lowland rice cultivation in Nigeria. *Scientific Reports*.

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 *Advances in Neural Information Processing Systems*, volume 36, pages 42330–42357.

Makbule Gulcin Ozsoy, Leila Messallem, Jon Besga, and Gianandrea Minneci. 2025. Text2Cypher: Bridging natural language and graph databases. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK) at COLING*.

Makbule Gulcin Ozsoy and William Tai. 2025. Text2Cypher across languages: Evaluating and finetuning LLMs. *Preprint*, arXiv:2506.21445.

Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022. QALD-9-plus: A multilingual dataset for question answering over DBpedia and Wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 229–234.

Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, volume 36, pages 36339–36348.

Md. Tanzim Reza, Tahsina Nimi, and Riasat Shahid. 2026. A provenance-traceable multi-task Bengali agricultural benchmark with safety-critical chemical advisory. *Preprint*, arXiv:2606.29243.

Priyanka Sen, Alham Fikri Aji, and Amir Saffari. 2022. Mintaka: A complex, natural, and multilingual dataset for end-to-end question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1604–1619.

Sheikh Shafayat, H M Quamran Hasan, Minhajur Rahman Chowdhury Mahim, Rifki Afina Putri, James Thorne, and Alice Oh. 2024. BEnQA: A question answering benchmark for Bengali and English. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1158–1177.

Zhili Shen, Chenxin Diao, Pavlos Vougiouklis, Pascual Merita, Shriram Piramanayagam, Enting Chen, Damien Graux, Andre Melo, Ruofei Lai, Zeren Jiang, Zhongyang Li, Ye Qi, Yang Ren, Dandan Tu, and Jeff Z. Pan. 2025. GeAR: Graph-enhanced agent for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12049–12072.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, pages 540–557.

Azmine Toushik Wasi, Taki Hasan Rafi, Raima Islam, and Dong-Kyu Chae. 2024. BanglaAutoKG: Automatic Bangla knowledge graph construction with semantic neural graph filtering. In *Proceedings of LREC-COLING 2024*, pages 2100–2106.

Jian Wu, Wenzhe Xie, Tao Wang, Xinyu Fan, Yang Li, and Xiangzeng Meng. 2025. Crop GraphRAG: pest and disease knowledge base question answering system for sustainable crop protection. *Frontiers in Plant Science*, 16.

Shuting Yang, Zehui Liu, and Wolfgang Mayer. 2024. ShizishanGPT: An agricultural large language model integrating tools and resources. In *Web Information Systems Engineering (WISE)*, LNCS. Springer.

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.

Xin Zhou et al. 2026. Knowledge-enhanced large language model construction for garlic cultivation using GraphRAG. *Frontiers in Plant Science*.

A Graph schema

The schema is small: 6 node labels, 6 relationship types and 42 label-specific property keys (Table 4). The prompt carries its query-relevant subset verbatim (Figure 5), so the benchmark tests grounding and composition on a fully specified schema rather than schema retrieval. Per graph this is a schema of ordinary size for the task: CypherBench graphs average 7.5 relation types and 18.7 properties (Feng et al., 2024), also given in full in the prompt; the difference is entity scale, 650 nodes here against hundreds of thousands in a Wikidata-derived graph.

B Template inventory

Tables 6 and 7 list one released item for each of the 26 templates in both languages, with the number of items instantiated from it. Bangla items keep place names, years and pest or disease names in Latin script, as these are the forms stored in the graph; crop names are always in Bangla.

Cross-family adequacy check. Each of the 460 Bangla items was back-translated to English by Pixtral Large (given the benchmark's 16-entry crop-name glossary so that vocabulary recall is not what is tested) and Llama 3.3 70B judged whether the back-translation asks for the same entities, quantity and constraints as the original English item; neither model belongs to the family that drafted or audited the Bangla. 422 of 460 items were judged equivalent. The 38 flags fall in six templates: *water req of top crop* (18; Bangla *prodhan foshol* does not mark number and was rendered "major crops", yet every model that answered it in Bangla returned a single crop, and Sonnet 5's

```
You translate natural-language questions about
↪ Bangladeshi agriculture into Cypher queries for a
↪ Neo4j knowledge graph.

Node labels and properties:
- Division {pcode, name}
- District {pcode, name}
- Upazila {pcode, name, area_sqkm, center_lat,
↪ center_lon}
- Crop {name, sowing_time, harvest_time,
↪ seed_requirement_per_acre,
        favourable_temperature, favourable_humidity,
        favourable_soil_temperature, photo_period,
↪ favourable_rainfall,
        water_requirement}
- SoilType {name}
- PestDisease {name}

Relationships:
- (Upazila)-[:IN_DISTRICT]->(District)
- (District)-[:IN_DIVISION]->(Division)
- (Crop)-[:SUITABLE_IN {very_suitable, suitable,
↪ moderately_suitable,
   marginally_suitable, not_suitable, total_area, zone,
↪ suitability_factor}]
  ->(Upazila)   // BARC land-suitability areas in
↪ hectares; factor 0..1
- (Upazila)-[:HAS_SOIL {characteristics}]->(SoilType)
- (Crop)-[:AFFECTED_BY {conditions}]->(PestDisease)
- (Crop)-[:PRODUCED_IN {crop_year, area_acres,
↪ production_mt,
   yield_mt_per_ha, source, quality_flag}]->(District)
  // BBS district production; crop_year like '2023-24',
↪ 2013-14..2024-25

The 16 crops: Aus, Aman, Boro, Wheat, Maize, Potato,
↪ Lentil, Mung, Gram,
Mustard, Groundnut, Chili, Onion, Garlic, Sugarcane,
↪ Jute.
District names use current spellings (Bogura,
↪ Chattogram, Jashore,
Chapainababganj, Barishal, Cumilla, ...).
A crop is called "highly suitable" in a place when
↪ suitability_factor > 0.7.

Rules:
- Output ONLY the Cypher query, no explanation, no code
↪ fences.
- The question may be in English or Bangla; entity
↪ names in the graph are in
  English (crops: Aus, Aman, Boro, ...; districts:
↪ current spellings).
- Read-only queries only (MATCH/RETURN); never CREATE,
↪ MERGE, SET or DELETE.
- Return ONLY the value or values the question asks
↪ for, nothing extra, and
  alias the main answer column AS answer.
```

Figure 5: The system prompt given verbatim to every model, in both languages; the question follows as the user turn. Long lines are broken at the column edge, and each continuation is marked ↪. The schema block is the query-relevant subset of Table 4: provenance properties (`source_span`, `barc_source_name`, and `source` except on `PRODUCED_IN`) are omitted, and the coordinate keys are named `center_lat` and `center_lon` where the graph stores `lat` and `lon`, a mismatch no released item touches.

Label or type	Count	Properties
Division	8	name, pcode
District	64	name, pcode
Upazila	507	name, pcode, area_sqkm, lat, lon
Crop	16	name, sowing_time, harvest_time, seed_requirement_per_acre, water_requirement, favourable_temperature, favourable_rainfall, favourable_humidity, favourable_soil_temperature, photo_period
SoilType	20	name
PestDisease	35	name
IN_DIVISION	64	–
IN_DISTRICT	507	–
SUITABLE_IN	7,456	suitability_factor, very_suitable, suitable, moderately_suitable, marginally_suitable, not_suitable, total_area, zone, barc_source_name, source, source_span
HAS_SOIL	1,471	characteristics, source
AFFECTED_BY	82	conditions, source
PRODUCED_IN	11,632	crop_year, production_mt, area_acres, yield_mt_per_ha, quality_flag, source

Table 4: Node labels (top) and relationship types (bottom) of BD-AgriKG with instance counts and property keys; 650 nodes and 21,212 relationships in total.

Bangla failures on it are an extra returned column, not a plural reading); *sowing time* (8; *ropon* covers both planting and transplanting); *count crops pest* (6; the translator dropped *koyti*, "how many", which is present in the Bangla); *pest conditions* and *water req* (5; Bangla idiomatically adds *dhan*, "rice", to the season names Aman, Aus and Boro); and *suitable and affected* (1; the translator dropped *otyonto*, "highly"). None changes the quantity, entities or constraints asked for, so no template was revised. Per-item back-translations and verdicts are released with the benchmark. This check bounds translation adequacy; it is not a native-speaker fluency rating.

Native-speaker rating. After the review period the first author, a native Bangla speaker, rated each of the 26 templates on two example items with different parameters, for fluency (1 unnatural, 2 acceptable, 3 natural) and for whether the Bangla asks exactly what the English asks. Fluency: 22 templates rated 3 and four rated 2 (*seed req*, *water req*, *division of top producer*, *pests of top crop*), none rated 1. Equivalence: 25 yes and one no, *seed req*, since the Bangla বীজ asks only for seed, while the English *planting material* can be read as planting equipment. Two rewordings were suggested without a change of meaning, পানি কেমন লাগে for পানির চাহিদা in *water req* and দেশে for জাতীয় উৎপাদন in *compare two crops*. The rater also noted that an extension worker would write district names, years and pest names in Bangla script, which is the design choice discussed in Limitations. The filled rating sheet is released with the benchmark; no template was changed after evaluation.

C Reproducibility details

All model runs used the Amazon Bedrock Converse API from a single account in region us-east-1 with cross-region inference profiles. The zero-shot sweep ran on 31 August 2026 and the repair runs on 2 September 2026, each item once, with a 1,500-token output limit. Non-Anthropic models were decoded at temperature 0; Anthropic models were called without sampling parameters. Table 8 gives the exact identifiers.

Because Anthropic models therefore sample at the API default, we repeated the Claude Haiku 4.5 English zero-shot run once (2 September 2026). Aggregate strict accuracy moved from 51.3% to 50.9% (236 vs. 234 correct), but 64 of 460 items changed outcome (31 failures became correct, 33 correct items failed) and only 136 of 460 queries were byte-identical across the two runs. Single-run differences of one or two points between models should therefore not be read as rankings, which is also what the clustered intervals imply. The repeated run doubles as a no-feedback control for Section 5.4: of the 72 Haiku English items that were invalid or empty in the first run, re-sampling alone made 10 correct (8 of 54 invalid, 2 of 18 empty), against 42 with the execution message.

D Per-category results

Figure 6 breaks strict execution accuracy down by question category for every model and language. Lookup questions saturate for every capable model, while comparison and multi-hop questions produce the widest spread between models. Table 5 gives, per model, the EN–BN gap with the interval, the raw and Holm-corrected p-values of three tests, and how much of each gap two templates carry.

E Per-template results

Because the 26 templates are the effective sampling unit (Section 5.1), Figure 7 reports strict execution accuracy for every template, model and language; the item count per template is given in parentheses. One template stands out: *compare two crops* is failed strictly even by the frontier models, which return the winning crop together with

Model	gap	95% CI	bootstrap p raw	bootstrap p Holm	sign-flip p raw	sign-flip p Holm	templates EN>BN / BN>EN / tie	sign test p raw	sign test p Holm	top-2 share	gap w/o top-2
Opus 5	+4.3	$[0.2, 10.0]$	0.044	0.132	0.126	0.496	5 / 1 / 20	0.219	0.474	85%	+0.7
Sonnet 5	+10.4	$[0.9, 21.6]$	0.023	0.092	0.124	0.496	4 / 0 / 22	0.125	0.474	67%	+3.8
DeepSeek-R1	+6.7	$[1.9, 11.7]$	0.006	0.030	0.019	0.113	13 / 3 / 10	0.021	0.128	39%	+4.5
Pixtral Large	+13.7	$[4.7, 21.1]$	0.005	0.030	0.005	0.034	21 / 2 / 3	0.000	0.000	29%	+10.6
Haiku 4.5	+3.9	$[-1.9, 10.5]$	0.215	0.430	0.291	0.582	11 / 4 / 11	0.118	0.474	89%	+0.5
Llama 3.3 70B	+5.2	$[0.7, 13.4]$	0.000	0.000	0.031	0.157	6 / 0 / 20	0.031	0.156	83%	+0.9
Llama 3.1 8B	+0.9	$[-0.9, 3.7]$	0.717	0.717	1.000	1.000	1 / 2 / 23	1.000	1.000	150%	-0.5

Table 5: Tests of the EN–BN strict gap per model (points). The interval and p are the paper's template-clustered percentile bootstrap; the sign-flip test permutes the sign of each template's paired difference; the sign test counts templates by the direction of their difference; Holm corrects each across the seven models. The last two columns give the share of the gap carried by its two largest templates and the gap recomputed without them.

Template (category)	n	English example	Bangla example
harvest time (lookup)	16	When is Onion harvested?	পেঁয়াজ কখন ঘরে তোলা হয়?
pest conditions (lookup)	18	Under what conditions does Rice Bug attack Aman?	কোন অবস্থায় আমন ধানে Rice Bug আক্রমণ করে?
seed req (lookup)	16	How much seed or planting material is needed per acre for Sugarcane?	আখ চাষে একর প্রতি কী পরিমাণ বীজ লাগে?
sowing time (lookup)	16	When is Wheat sown, planted or transplanted in Bangladesh?	বাংলাদেশে গম কখন বপন বা রোপণ করা হয়?
upazila district (lookup)	18	Which district is Mithapukur upazila in?	Mithapukur উপজেলা কোন জেলায় অবস্থিত?
upazila soils (lookup)	18	What soil types are found in Sreebardi upazila?	Sreebardi উপজেলায় কোন কোন ধরনের মাটি পাওয়া যায়?
water req (lookup)	16	What is the water requirement of Groundnut?	চীনাবাদাম চাষে পানির চাহিদা কেমন?
count crops pest (aggregation)	18	How many crops are attacked by Seedling Blight?	Seedling Blight কয়টি ফসলে আক্রমণ করে?
count producing districts (aggregation)	18	In how many districts was Lentil produced in 2015-16?	2015-16 সালে কতটি জেলায় মসুর উৎপাদিত হয়েছিল?
count upazilas (aggregation)	18	How many upazilas are there in Netrakona district?	Netrakona জেলায় কয়টি উপজেলা আছে?
division production (aggregation)	18	How much Groundnut was produced in Chattogram division in 2015-16?	2015-16 সালে Chattogram বিভাগে কত চীনাবাদাম উৎপাদিত হয়েছিল?
national production (aggregation)	18	What was the total national production of Mustard in 2024-25?	2024-25 সালে সারা দেশে সরিষার মোট উৎপাদন কত ছিল?
best upazila suitability (ranking)	18	Which upazila of Brahmanbaria district is most suitable for growing Aus?	Brahmanbaria জেলার কোন উপজেলা আউশ ধান চাষের জন্য সবচেয়ে উপযোগী?

Table 6: Template inventory, part 1 of 2: one released item per template, with the number of items it yields.

the compared production total; the lenient metric of Table 1 accepts these answers, so the template measures output discipline rather than query logic.

F Repair by signal type

Table 9 splits the repairable first attempts of Section 5.4 by the signal the model received (a Neo4j error message for invalid queries, a zero-rows notice for empty results) and reports how many of each were recovered to a strictly correct answer. Tables 2 and 9 classify empty versus invalid by re-executing failed queries with 3 s and 6 s timeouts; two Llama 3.1 8B queries that time out at 3 s return zero rows at 6 s, hence the two-item difference between the tables. The retry turn keeps the system prompt and appends to the original question the text "`Your previous query:`" followed by the first-attempt Cypher, then one of "`Executing your query failed with this error: <message>.`" or "`Executing your query executed but returned zero rows.`", and finally "`Reply with a corrected Cypher query only, following the same`

Template (category)	n	English example	Bangla example
highest yield district (ranking)	18	Which district had the highest Maize yield per hectare in 2024-25?	2024-25 সালে কোন জেলায় হেক্টর প্রতি ভুট্টার ফলন সবচেয়ে বেশি ছিল?
most suitable crop upazila (ranking)	18	Which three crops are most suitable for cultivation in Basail upazila?	Basail উপজেলায় চাষের জন্য সবচেয়ে উপযোগী তিনটি ফসল কী কী?
top crop in district (ranking)	18	Which crop had the highest production in Chapainababganj district in 2014-15?	2014-15 সালে Chapainababganj জেলায় কোন ফসলের উৎপাদন সবচেয়ে বেশি ছিল?
top districts production (ranking)	18	Which three districts produced the most Mustard in 2018-19?	2018-19 সালে কোন তিনটি জেলায় সবচেয়ে বেশি সরিষা উৎপাদিত হয়েছিল?
area trend (temporal)	18	How has the cultivated area of Onion changed over the years in Bhola district?	Bhola জেলায় পেঁয়াজ চাষের জমির পরিমাণ বিভিন্ন বছরে কীভাবে বদলেছে?
best year yield (temporal)	18	In which year did Chuadanga district get its best Mustard yield?	Chuadanga জেলায় কোন বছর সরিষার ফলন সবচেয়ে ভালো হয়েছিল?
production two years (temporal)	18	Compare the production of Garlic in Rajbari district between 2019-20 and 2023-24.	2019-20 এবং 2023-24 সালে Rajbari জেলায় রসুনের উৎপাদন তুলনা করুন।
compare two crops (comparison)	18	Which had the higher national production in 2023-24, Chili or Groundnut?	2023-24 সালে মরিচ নাকি চীনাবাদাম, কোনটির জাতীয় উৎপাদন বেশি ছিল?
compare two districts (comparison)	18	Between Chandpur and Jhalokati, which district produced more Aus in 2019-20?	2019-20 সালে Chandpur নাকি Jhalokati, কোন জেলায় বেশি আউশ ধান উৎপাদিত হয়েছিল?
division of top producer (multihop)	18	In which division is the district that produced the most Jute in 2014-15?	2014-15 সালে সবচেয়ে বেশি পাট উৎপাদনকারী জেলাটি কোন বিভাগে অবস্থিত?
pests of top crop (multihop)	18	Which pests and diseases threaten the top crop of Gazipur district (by 2018-19 production)?	2018-19 সালের উৎপাদন অনুযায়ী Gazipur জেলার প্রধান ফসলটিতে কোন কোন পোকা ও রোগ আক্রমণ করতে পারে?
suitable and affected (multihop)	18	Which crops that are highly suitable somewhere in Feni district are attacked by Flea Beetle?	Feni জেলায় চাষের জন্য অত্যন্ত উপযোগী এমন কোন কোন ফসলে Flea Beetle আক্রমণ করে?
water req of top crop (multihop)	18	What is the water requirement of the top crop of Khulna district (by 2018-19 production)?	2018-19 সালের উৎপাদন অনুযায়ী Khulna জেলার প্রধান ফসলের পানির চাহিদা কেমন?

Table 7: Template inventory, part 2 of 2: one released item per template, with the number of items it yields.

Name in paper	Bedrock model identifier
Opus 5	`us.anthropic.claude-opus-5`
Sonnet 5	`us.anthropic.claude-sonnet-5`
Haiku 4.5	`us.anthropic.claude-haiku-4-5-20251001-v1:0`
DeepSeek-R1	`us.deepseek.r1-v1:0`
Pixtral Large	`us.mistral.pixtral-large-2502-v1:0`
Llama 3.3 70B	`us.meta.llama3-3-70b-instruct-v1:0`
Llama 3.1 8B	`us.meta.llama3-1-8b-instruct-v1:0`

Table 8: Model identifiers used for every run.

`rules.`" The strings are verbatim, including the awkward grammar of the zero-rows variant. Only failed items were retried; no correct item was re-run or re-scored.

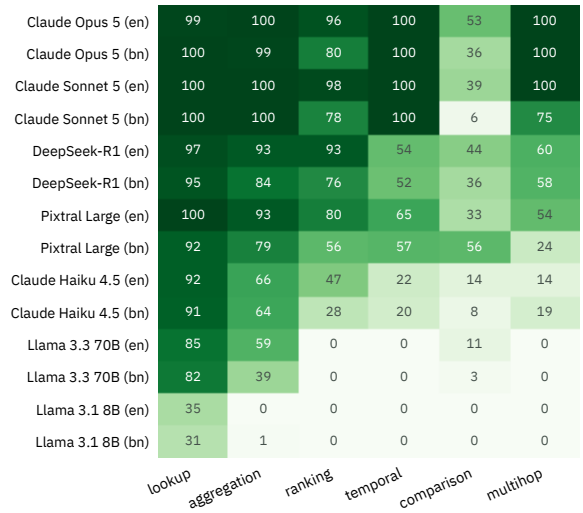


Figure 6: Strict execution accuracy per model, language and question category. Comparison and multi-hop questions separate the models most; lookup saturates for every capable model.

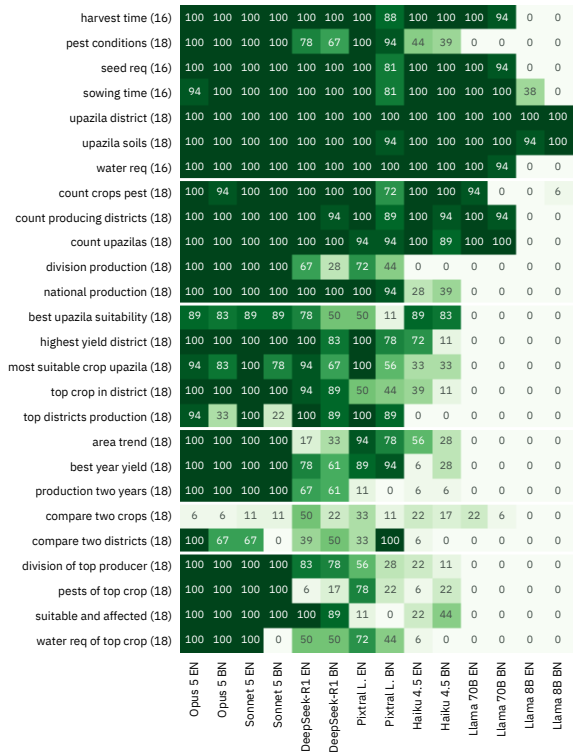


Figure 7: Strict execution accuracy (%) per template (rows, grouped by category) for each model and language (columns).

	English				Bangla			
	invalid		empty		invalid		empty	
Model	n	rec.	n	rec.	n	rec.	n	rec.
Opus 5	1	1	0	0	0	0	0	0
Sonnet 5	0	0	0	0	0	0	0	0
Haiku 4.5	54	36	18	6	46	30	35	16
DeepSeek-R1	55	36	3	1	61	46	10	6
Pixtral Large	38	12	36	22	47	15	67	34
Llama 3.3 70B	148	10	43	1	132	1	45	1
Llama 3.1 8B	67	0	237	1	64	0	214	0

Table 9: Repairable first attempts (n) and recoveries (rec.) by signal type and language; Section 5.4 quotes their row sums.

Source-Aware Memory Retrieval across Engineering Standards

Sung Tae Yoo and Heyoung Yang
Applied Agent Research Center
Korea Institute of Science and Technology Information
Seoul, Republic of Korea
yst1213@kisti.re.kr, hyyang@kisti.re.kr

Abstract

Engineering design and quality review often require evidence from technical standards issued by multiple organizations. In a limited context window, partial evidence is often included, and relevance-based retrieval results can become biased toward sources with a large volume of documents, leaving other necessary sources unrepresented. We implemented a deterministic source-aware retrieval layer that identifies required source groups, retrieves candidates through source-conditioned channels, and prioritizes one candidate from each channel. We also evaluated a configuration that uses within-channel cross-encoder reranking to select that candidate. Evaluation used 15 technical standards, 72 evaluation questions, and 162 annotated evidence locations, with k set to 5 for all methods. Required-source coverage was 61.5% for dense retrieval and 78.6% for cross-encoder reranking, whereas both source-aware methods achieved 100%. Source-aware reranked channel quota achieved an evidence recall of 46.0% and retrieved all required evidence for 27.8% of questions, compared with 41.8% and 23.6% for cross-encoder reranking and 32.1% and 15.3% for dense retrieval. Within the source-aware design, within-channel reranking increased mean reciprocal rank from 0.445 to 0.545. These results support separating required-source coverage from evidence ranking in bounded-context retrieval for domain-specific agent memory.

1 Introduction

In engineering design and quality review, evaluating a product or component requires verifying requirements specified in technical standards issued by multiple organizations (Luttmer et al., 2023; Schuk et al., 2022). This study arises from an industrial valve design and review workflow in which engineers manually check the relevant clauses and tables in each standard. Because these requirements are expressed using different clause numbering, table formats, units, and terminology across organizations, engineers often need to cross-check relevant clauses and tables in multiple documents. If a required source is missing from the retrieved results, the judgment can remain incomplete even when evidence from the other sources is accurate.

These retrieval requirements also arise in domain-specific memory architectures for agents. The standards corpus can be treated as external memory, with the retrieval layer determining which evidence is brought into the limited context window (Lewis et al., 2020; Packer et al., 2024). Because the full standards corpus cannot fit within a limited context window, a small number of retrieval results are included in the context. Relevance-oriented retrieval can produce unequal document exposure, making some documents more retrievable than others (Azzopardi and Vinay, 2008; Wilkie and Azzopardi, 2017). When a question requires evidence from standards issued by multiple organizations, such unequal exposure may leave some required sources unrepresented in the limited context window.

This study examines whether identifying the source groups required by a query and retrieving through source-conditioned channels can improve coverage of the required sources. It then evaluates whether within-channel reranking can further improve evidence selection. We use a deterministic source-aware retrieval layer to separate source coverage from evidence selection. Source-aware channel quota prioritizes one candidate from each channel before filling the remaining results by dense score, while source-aware reranked channel quota applies cross-encoder reranking within each channel before selecting that candidate. Both methods remain deterministic and differ only in how that candidate is selected.

The evaluation separately considers whether the required sources are represented in the retrieved results and whether the specific evidence needed for a judgment is retrieved. Evidence is evaluated at

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 77–83
November 10, 2026 ©2026 Association for Computational Linguistics

the clause and table level rather than by page overlap, so these two aspects can be assessed separately. This distinction provides a practical way to evaluate multi-source evidence retrieval for technical standards and supports the source-aware retrieval design used in this study.

The contributions of this study are twofold. First, we distinguish required-source coverage from evidence selection as two complementary objectives in multi-source retrieval over technical standards. Second, we operationalize this distinction through a deterministic source-aware retrieval layer that uses source-group structure as a retrieval condition and prioritizes required-source representation before within-source evidence selection.

2 Related Work

External memory and retrieval have become important components of large language model (LLM)-based agent architectures. Retrieval-augmented generation (RAG) retrieves relevant passages from external memory and uses them during generation, giving the model access to information beyond what it already contains (Lewis et al., 2020). MemGPT uses an operating-system-inspired memory architecture to transfer information between the context window and external storage, allowing an LLM-based agent to access more information than the context window can hold (Packer et al., 2024). RET-LLM adds explicit read and write operations over external memory (Modarressi et al., 2024). Agent memory mechanisms also treat retrieval as a memory operation distinct from writing and management (Zhang et al., 2025).

Retrieval quality depends not only on relevance but also on whether the retrieved results cover the required sources. Maximal marginal relevance combines query relevance with information novelty to reduce redundancy in the selected results (Carbonell and Goldstein, 1998), but it does not use source identity as part of the selection criterion. Azzopardi and Vinay (2008) introduced retrievability as a document-centric measure of how easily individual documents can be retrieved within a collection across a set of queries, and Wilkie and Azzopardi (2017) later examined whether better-performing systems make relevant documents more retrievable than non-relevant ones. Retrievability characterizes differences in document exposure across a collection and a set of queries, rather than the source composition of a single result set. Retrieval methods span lexical ranking (Robertson and Zaragoza, 2009), rank fusion (Cormack et al., 2009), cross-encoder reranking (Nogueira and Cho, 2020), and dense retrieval (Karpukhin et al., 2020). These methods improve candidate ranking or combination, but do not account for the source groups required by a query when selecting the final results.

Retrieval of required evidence has also been studied in applications involving regulatory documents and technical standards. Gokhan and Briscoe (2025) report lower retrieval performance when regulatory questions require multiple passages. KG2QA (Luo et al., 2025) combines a domain knowledge graph with RAG for question answering over communication standards, while Ju and Lee (2026) cast regulatory compliance question answering as retrieving a closed set of provisions linked by explicit citations. These approaches use passage ranking, citation links, or graph structure to retrieve related evidence. When the required evidence spans standards issued by multiple organizations, however, coverage of the required source groups also becomes important.

3 Method and Evaluation

3.1 Task, Corpus, and Evaluation Set

The retrieval task is to identify the evidence needed for each evaluation question from a collection of technical standards. The evaluation corpus consists of 15 technical standards from four source groups, A to D. The standard documents are divided into 1,287 retrieval units in Korean and English. The four source groups account for 58.5%, 23.7%, 14.5%, and 3.3% of the total retrieval units, respectively. The three documents in the largest source group account for more than half of the corpus. Each retrieval unit contains source-group information used for source-conditioned retrieval.

The evaluation set consists of 72 questions linked to 162 evidence locations annotated by clause or table. Each question requires one to four evidence locations. Among the 72 questions, 15 require a single evidence location, while 12 require two or more evidence locations from the same source group. The 45 multi-source questions consist of 22 questions requiring evidence from two source groups, 15 questions requiring evidence from three source groups, and 8 questions requiring evidence from four source groups.

The evaluation uses five retrieved results per question ($k = 5$). Each evaluation question re-

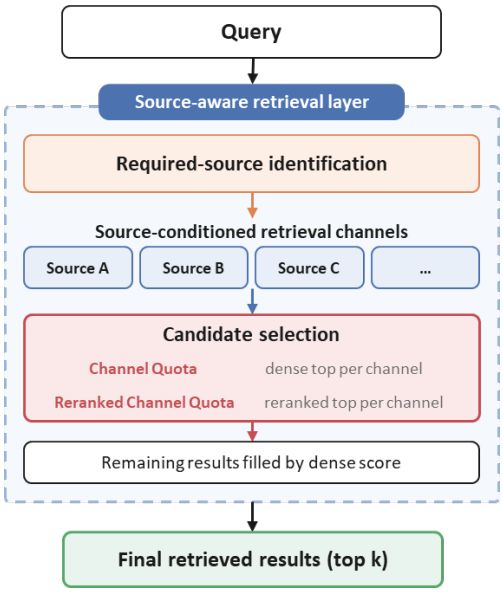


Figure 1: The source-aware retrieval layer. The two methods share source-group identification, source-conditioned channels, candidate pooling, and dense-score filling of the remaining results, and differ only in per-channel candidate selection.

quires evidence from one to four source groups. Five retrieved results can therefore cover all required source groups while leaving one to four additional results for evidence selection. This study is limited to evidence retrieval and does not include subsequent answer generation. Appendix A provides example evaluation questions, and Appendix B reports additional results for different values of k.

3.2 Source-Aware Retrieval

Figure 1 shows the source-aware retrieval layer and its retrieval flow. The retrieval process first identifies the source groups required by a query by rule-based matching between query terms and the scope of each source group. Candidates are then retrieved through source-conditioned channels for the identified groups, with candidates filtered by source-group labels. One candidate from each channel is prioritized. The remaining results are then filled by dense score from the union of channel candidates.

The two methods differ at this prioritization step. Source-aware channel quota prioritizes the highest-ranked unused candidate from each channel under dense retrieval. Source-aware reranked channel quota reranks up to 50 candidates within each channel with a cross-encoder before prioritizing the highest-ranked unused candidate. The remaining retrieval steps are shared. Both methods are deterministic and do not use an LLM during retrieval.

3.3 Evaluation Protocol

The source-aware retrieval layer is evaluated against three relevance-based baselines under the same protocol. Dense retrieval serves as the semantic retrieval baseline. Reciprocal rank fusion (RRF) (Cormack et al., 2009) is used in Hybrid RRF to combine dense and lexical rankings. Cross-encoder rerank applies reranking to an unconditioned candidate pool. BGE-M3 (Chen et al., 2024) provides the dense ranking used in dense retrieval and the source-aware methods, while BGE-reranker-v2-m3 is used for cross-encoder reranking. The comparison examines the effect of source-conditioned retrieval and the additional effect of within-channel reranking.

Scoring requires mapping annotated evidence locations to retrieval units. The mapping uses the parsed clause identifier, heading, and positional cue stored in each annotation. An annotated evidence location maps to 1.15 retrieval units on average, with a median of one, and 87.0% of annotated evidence locations map to a single unit. When clause information is unusable, positional cues are used and the mapping is marked low confidence. The mapping rules were fixed before system scoring.

We use five evaluation metrics. Evidence recall is the share of required evidence retrieved, averaged across questions. Any evidence is the percentage of questions retrieving at least one required evidence location and is reported as a diagnostic measure. All required evidence is the percentage of questions for which every required evidence location is retrieved. Source coverage measures the share of required source groups covered by the retrieved results. Mean reciprocal rank (MRR) is the reciprocal rank of the first correct retrieval, averaged across questions (Voorhees and Tice, 2000).

4 Results

4.1 Source-Aware Retrieval Performance

Table 1 compares the retrieval methods with five retrieved results per question ($k = 5$). The clearest difference appears in source coverage. Dense retrieval and hybrid RRF reach 61.5% and 52.0%, respectively, while cross-encoder rerank reaches 78.6%. Both source-aware channel quota and source-aware reranked channel quota reach 100%. The source-aware methods therefore include all re-

quired source groups in the final retrieved results, whereas the relevance-based baselines can still miss required sources.

The other evidence-level measures show different results. Source-aware channel quota retrieves all required evidence for 25.0% of questions, with source coverage remaining at 100%. Its evidence recall is 40.5%, slightly below 41.8% for cross-encoder rerank. MRR is also slightly lower at 0.445, compared with 0.501 for cross-encoder rerank. This comparison separates required-source coverage from evidence selection within those sources. Source-aware reranked channel quota improves the evidence-level results after within-channel reranking, while source coverage remains at 100%. Evidence recall rises to 46.0%, all required evidence to 27.8%, and MRR to 0.545. Source-aware reranked channel quota shows higher overall evidence-level results than the other methods, while both source-aware methods cover every required source group. Appendix B shows that the percentage of questions with all required evidence increases as k increases across all methods, while both source-aware methods continue to cover all required source groups. Reranked channel quota also has the highest evidence recall, any evidence, and MRR at each tested k.

Figure 2 shows that the effect of within-channel reranking differs by question type, with cross-encoder rerank included as a reference. For single-evidence questions, the all-required-evidence rate increases from 86.7% to 100%. Same-source multi-evidence questions change less. All required evidence remains at 16.7% and evidence recall at 41.7%, although MRR rises from 0.496 to 0.537.

For two-source questions, reranked channel quota improves all three measures over channel quota. Three-source questions show a different pattern. Evidence recall increases from 17.8% to 22.2% and MRR from 0.263 to 0.467, but all required evidence decreases from one question to none. For four-source questions, evidence recall increases, but MRR decreases slightly. These results suggest that better ranking does not always lead to retrieval of all required evidence.

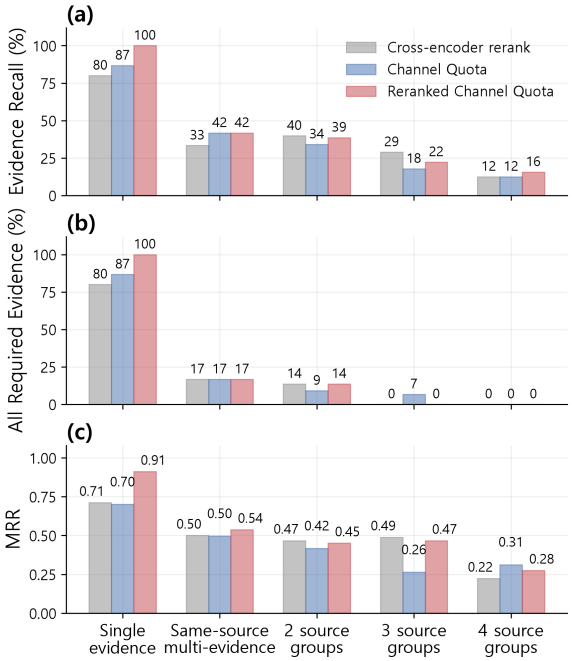


Figure 2: Comparison by question type for cross-encoder rerank, source-aware channel quota, and source-aware reranked channel quota. Panels show (a) evidence recall, (b) all required evidence, and (c) MRR.

Cross-encoder rerank provides a useful reference because it uses the same reranking model on an unconditioned candidate pool. Reranked channel quota performs better on the single-evidence questions and has higher evidence recall and MRR for same-source multi-evidence questions. Cross-encoder rerank has higher evidence recall and MRR for the two-source and three-source groups, whereas reranked channel quota is higher on these two measures for the four-source group. For all required evidence, the two methods give the same

Method	Evidence Recall	Any Evidence	All Required Evidence	Source Coverage	MRR
Dense Retrieval	32.1	52.8	15.3	61.5	0.374
Hybrid RRF	32.4	55.6	15.3	52.0	0.345
Cross-encoder rerank	41.8	65.3	23.6	78.6	0.501
Channel Quota	40.5	61.1	25.0	100.0	0.445
Reranked Channel Quota	**46.0**	**68.1**	**27.8**	**100.0**	**0.545**

Table 1: Retrieval performance at $k = 5$. Evidence recall, any evidence, all required evidence, and source coverage are percentages; MRR is on a 0–1 scale.

result for every multi-evidence question type in Figure 2. The difference seen in Table 1 therefore arises from the single-evidence questions. Across the full evaluation, the clearer distinction is coverage of required sources. Both source-aware methods cover every required source group, whereas cross-encoder rerank covers 78.6% of them.

5 Discussion

5.1 Required-Source Coverage as a Separate Retrieval Objective

Required-source coverage measures a part of retrieval that relevance ranking alone does not fully reflect. Within the source-aware retrieval layer, both methods include every required source in the final retrieved results. Relevance-based baselines still miss some required sources, even when cross-encoder rerank improves evidence-level ranking. Reranking changes candidate order, but it does not by itself ensure that every required source is included. This distinction becomes important when a review draws requirements from several standards. The source-aware retrieval layer therefore supports selective retrieval from external memory by covering the required sources before selecting evidence within those sources for the limited context window. In this workflow, the practical concern is whether a required standard is absent from the retrieved results, not only whether relevant evidence appears in a different order. Source-group information attached to each retrieval unit acts as a minimal structural layer over the corpus, and retrieval is conditioned on that structure rather than on relevance alone. Richer structure over standards, such as knowledge-graph relations (Luo et al., 2025) or citation links across documents (Ju and Lee, 2026), could extend the same separation beyond source groups.

5.2 Evidence Selection after Source Coverage

After the required source groups are covered, the quality of the retrieved evidence becomes more important. Within-channel reranking improves this part of the retrieval process. Source-aware reranked channel quota shows higher evidence recall and MRR than source-aware channel quota, and the improvement is especially noticeable for single-evidence questions. A cross-encoder scores each query-candidate pair independently, so better-ranked candidates do not necessarily produce a retrieved set containing all required evidence. This difference becomes more apparent when several evidence locations are required for the same question. Multi-evidence retrieval should therefore consider how the retrieved results work together rather than scoring each candidate separately. With five retrieved results per question, retrieval of all required evidence remains limited even when all required source groups are covered, indicating that source coverage and evidence completeness remain distinct challenges. Future work should examine how the number of retrieved results interacts with the number of prioritized candidates per channel, particularly for questions requiring multiple evidence locations. Another direction is to improve the selection of remaining results while retaining the priority given to required-source representation.

6 Conclusion

Engineering review across technical standards from multiple organizations requires both coverage of the required source groups and selection of appropriate evidence within those sources. The source-aware retrieval layer separates these two parts of retrieval. Both source-aware methods improve source coverage, and within-channel reranking further improves evidence recall and MRR. The same design supports selective retrieval from external memory by using source-group structure to ensure required-source coverage before refining evidence selection within each source. This retrieval layer can serve as an external-memory component in future agent architectures for technical standards.

Limitations

The comparison focuses on retrieval behavior rather than equalizing computation across methods, and the source-aware variants process candidates separately within source-conditioned channels. The two source-aware methods remain directly comparable, but the evaluation is limited to 15 standards and 72 questions. The underlying standards cannot be released because of copyright restrictions. The full evaluation set is not released, but selected evaluation questions and answers are provided in Appendix A.

Declaration on Generative AI

The authors used a generative AI assistant for language editing only. It was not used to generate ideas, design experiments, or produce any reported results.

References

Leif Azzopardi and Vishwa Vinay. 2008. Retrievability: an evaluation measure for higher order information access tasks. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 561–570, New York, NY, USA. Association for Computing Machinery.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 335–336, New York, NY, USA. Association for Computing Machinery.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 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*, SIGIR '09, pages 758–759, New York, NY, USA. Association for Computing Machinery.

Tuba Gokhan and Ted Briscoe. 2025. Grounded Answers from Multi-Passage Regulations: Learning-to-Rank for Regulatory RAG. In *Proceedings of the Natural Legal Language Processing Workshop 2025*, pages 135–146, Suzhou, China. Association for Computational Linguistics.

Yeong-Joon Ju and Seong-Whan Lee. 2026. Citation-Closure Retrieval and Per-Rule Attribution for Real-World Regulatory Compliance Question Answering. *arXiv preprint*. ArXiv:2605.29742 [cs.AI].

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 9459–9474, Red Hook, NY, USA. Curran Associates Inc.

Zhongze Luo, Weixuan Wan, Tianya Zhang, Dan Wang, and Xiaoying Tang. 2025. KG2QA: Knowledge Graph-enhanced Retrieval-augmented Generation for Communication Standards Question Answering. *arXiv preprint*. ArXiv:2506.07037 [cs.CL].

Janosch Luttmer, Vitalijs Prihodko, Dominik Ehring, and Arun Nagarajah. 2023. Requirements extraction from engineering standards – systematic evaluation of extraction techniques. *Procedia CIRP*, 119:794–799.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2024. RET-LLM: Towards a General Read-Write Memory for Large Language Models. *arXiv preprint*. ArXiv:2305.14322 [cs.CL].

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage Re-ranking with BERT. *arXiv preprint*. ArXiv:1901.04085 [cs.IR].

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint*. ArXiv:2310.08560 [cs.AI].

Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Vitali Schuk, María Elena Pombo Jiménez, and Ullrich Martin. 2022. Technical specifications to meet the requirements of an Automatic Code Compliance Checking tool and current developments in infrastructure construction. *Results in Engineering*, 16:100650.

Ellen M. Voorhees and Dawn M. Tice. 2000. The TREC-8 Question Answering Track. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00)*, Athens, Greece. European Language Resources Association (ELRA).

Colin Wilkie and Leif Azzopardi. 2017. Algorithmic Bias: Do Good Systems Make Relevant Documents More Retrievable? In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 2375–2378, New York, NY, USA. Association for Computing Machinery.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A Survey on the Memory Mechanism of Large Language Model-based Agents. *ACM Transactions on Information Systems*, 43(6):155:1–155:47.

Appendix

A Evaluation Question Examples

The evaluation questions were written from clauses and tables in the corpus and linked to annotated evidence locations. Two examples from the evaluation set are shown below.

Method	k	Evidence Recall	Any Evidence	All Required Evidence	Source Coverage	MRR
Dense Retrieval	10	44.1	65.3	25.0	71.8	0.392
	15	49.6	69.4	31.9	80.7	0.395
	20	52.9	75.0	34.7	83.3	0.398
Hybrid RRF	10	39.1	65.3	18.1	57.8	0.357
	15	43.4	68.1	20.8	69.1	0.359
	20	49.4	76.4	23.6	73.2	0.363
Cross-encoder rerank	10	53.0	76.4	33.3	85.5	0.515
	15	55.6	79.2	37.5	87.4	0.517
	20	58.5	81.9	40.3	92.1	0.519
Channel Quota	10	53.5	75.0	37.5	100.0	0.465
	15	58.9	84.7	38.9	100.0	0.472
	20	64.6	88.9	43.1	100.0	0.475
Reranked Channel Quota	10	55.1	80.6	36.1	100.0	0.561
	15	60.3	86.1	40.3	100.0	0.565
	20	65.3	90.3	44.4	100.0	0.568

Table 2: Retrieval performance for $k \in \{10, 15, 20\}$. Evidence recall, any evidence, all required evidence, and source coverage are percentages; MRR is reported on a 0–1 scale. Both source-aware methods maintain 100% source coverage across all tested values of k.

Question 1. How are the minimum hold times for an industrial valve body pressure test divided by DN range?

Answer. The minimum hold times are 15 s for DN 50 and below, 60 s for DN 65 through DN 200, and 180 s for DN 250 and above.

This is a single-evidence question linked to one annotated evidence location in a table.

Question 2. For pressure testing of a steel gate valve for oil and gas service, what is the basis for calculating the body pressure test pressure and which test types are required?

Answer. The body pressure test pressure is at least 1.5 times the pressure corresponding to the 38°C pressure rating. The required tests are the shell pressure test, closure leakage test, and stem backseat test.

This is a same-source multi-evidence question linked to two annotated evidence locations.

B Sensitivity to k

Table 2 shows the results for k values of 10, 15, and 20. The percentage of questions with all required evidence increases with k for every method, while both source-aware methods continue to cover all required source groups. Source-aware reranked channel quota gives the highest evidence recall, any evidence, and MRR at each tested k. It also gives the highest all-required-evidence rate at $k = 15$ and $k = 20$. At $k = 10$, source-aware channel quota is slightly higher, reaching 37.5% compared with 36.1%.

Beyond Chunking: Node Integrity and Field Structure in Bengali Retrieval-Augmented Agent Memory

Khan Raiyan Ibne Reza[1], **Sanjana Aktar Maria**[1], **Sumaiya Tabassum Nimi**[1]

[1]North South University, Dhaka, Bangladesh

Correspondence: {raiyan.reza,sanjana.maria}@northsouth.edu

Abstract

Lexical retrieval in low-resource languages can fail when queries use symptoms, colloquial names, or procedural descriptions rather than formal terminology. Chunking can compound this problem by separating query-matching content from the field that identifies a record. We introduce a three-level representation ladder (chunked text, intact full-node text, and field-aware lexical indexing) that separates recovery associated with record integrity from the additional effect of field-aware representation. In the OCR-heavy education corpus, intact records raise Recall@10 from 37.19% to 89.65%. Field-aware indexing adds gains with fixed weights in agriculture and uniform weights in education, but uniform weighting underperforms intact text in law. Ablations indicate that useful fields depend on query formulation; full field-value shuffling reduces Recall@10 to 0.75–5.09%. In agent memory, Recall@5 rises from 27.6% to 61.9% with intact records on 583 agriculture and education queries. Across 872 queries in all three domains, uniform fielded memory has a citation rate about 12 points higher than intact memory, without higher aggregate Recall@5. These results indicate that record integrity and query-aligned content warrant separate evaluation in retrieval and agent memory.

1 Introduction

A farmer in rural Bangladesh may not type "Fusarium wilt of tomato" into a search box; she describes yellowing leaves, a plant wilting despite regular watering, or a smell in the soil. A student preparing for a lab exam may remember the apparatus and steps rather than the formal name of the experiment, while a citizen asking about a law may describe the act as people around her do rather than quote its section number. In each case, retrieval must connect the user's description to a knowledge record expressed in more formal terminology.

Most retrieval-augmented generation (RAG) systems bridge this gap by splitting source documents into fixed-size text chunks and indexing each independently. This becomes problematic when knowledge is naturally organized as a record: a disease name with its symptoms and treatment, an act title with its provisions, or an experiment name with its procedure. A 500-character window can place the symptom a user searches for in one chunk and the disease name that identifies the record in another. The query and the record are then separated before retrieval begins, and downstream reasoning cannot recover what was never retrieved together.

This issue is particularly relevant to Bengali domain retrieval, where query language can differ substantially from formal documents. Regional and colloquial variation is common (Faria et al., 2023; Sami et al., 2026), Bengali remains underresourced relative to its number of speakers (Joshi et al., 2020), and recent work shows that RAG performance can be sensitive to variation between query language and underlying text (Cao et al., 2026b). Bengali therefore provides a useful setting for asking whether the representation of a knowledge record itself affects retrieval.

A natural approach is to preserve record structure and index its fields directly. We treat fielded records as a lightweight, graph-free form of structured memory: each record is a typed unit, but we model no relations between records. GraphRAG and HippoRAG add such relations, which we do not evaluate (Edge et al., 2024; Gutiérrez et al., 2024). However, comparing chunked retrieval with fielded retrieval alone conflates two effects: recovery from splitting a record, and the combined effect of adding field copies and field weights. How much retrieval performance can be recovered simply by keeping a record intact? What additional benefit comes from adding field content and weights?

We answer these questions with a three-level representation ladder applied to the same underlying

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 84–97
November 10, 2026 ©2026 Association for Computational Linguistics

knowledge. At L_1, each record is split into 500-character chunks. At L_2, the full record is kept as one unstructured document, removing chunk boundaries without exposing field information. At L_3, selected field content is added using BM25F-inspired scoring (Robertson et al., 2004). Thus, $L_1 \rightarrow L_2$ measures recovery associated with preserving an intact record, while $L_2 \rightarrow L_3$ measures the combined effect of adding field copies and field weights. We apply the ladder to three Bengali domains with different query regimes: agricultural disease advisory, where queries are predominantly symptom-oriented but may name diseases or pests; statutory law, where users identify a known act; and secondary science education, where users describe experimental procedures. Across 3,670 queries and 1,953 knowledge nodes, the effects are substantial but not uniform. In education, Recall@10 rises from 37.19% at L_1 to 89.65% at L_2, with uniform BM25F adding 1.93 points. In agriculture, both steps help under fixed nonuniform field weights. In law, however, full-node retrieval reaches 79.55%, while uniform BM25F reaches 77.82%.

The remaining question is what makes field structure useful when it does help. Field-group ablations indicate that fields are useful when their surface forms match how users phrase their queries; full field-value shuffling shows that correct content assignment matters. We call such a field a *query-aligned anchor*. In law, `act_title_bn` alone reaches 100.00% Recall@10, whereas `act_no_bn`, which identifies the same acts, reaches only 2.18%. Shuffling all field values across records collapses Recall@10 to 0.75–5.09%, indicating that the observed retrieval signal depends on the content assigned to fields, rather than field boundaries alone.

Finally, we test whether representation matters when the knowledge store is used as agent memory. Poor retrieval can leave an agent without relevant evidence (Gutiérrez et al., 2024; Xu et al., 2025). We hold the agent controller fixed and vary its memory representation. On 583 agriculture and education queries, Recall@5 increases from 27.6% with chunked memory to 61.9% with full-node memory. Legal chunked-agent results have an identifier mismatch and are excluded from this comparison. On the 872-query three-domain subset, uniform fielded memory has a higher citation rate than full-node memory, while benchmark-informed weights provide an exploratory retrieval ceiling.

The results support a narrower conclusion: a substantial portion of the gain attributed to structured retrieval can arise from preserving record integrity, while the additional benefit of field-weighted retrieval depends on the scoring configuration and query-aligned content. Concretely, we contribute:

1. **A representation ladder** that distinguishes recovery from preserving intact records from the combined effect of field copies and field weights, evaluated across three Bengali domains and 3,670 queries.
2. **A field-level account of when structure helps**, using group ablations and counterfactual field permutations to examine how retrieval depends on query-aligned content and varies with the domain's query regime.
3. **A transfer of this representation ladder to agent memory**, examining representation-level recovery and, in an exploratory perturbation, sensitivity to query-aligned fields.

2 Related Work

Chunking, structured RAG, and fielded retrieval. Conventional RAG pipelines commonly split source documents into independently retrievable chunks, but recent work shows that this choice can affect retrieval quality. CausalRAG identifies disrupted contextual integrity as a consequence of chunking and introduces causal graphs to preserve relationships across the resulting fragments (Wang et al., 2025). DyVo studies a related fragmentation problem inside the tokenizer, where entity mentions are split across subword units; our setting is different in that a text splitter separates parts of an already structured record (Nguyen et al., 2024). Other approaches add structure after retrieval: StructRAG reconstructs retrieved passages into task-specific structures, while SRAG augments queries and retrieved chunks with structured information (Li et al., 2025b; Shah et al., 2026). These approaches improve or repair retrieval pipelines, but do not isolate how much of the gain comes from recovering record integrity versus exposing structure itself. Our representation ladder keeps the source knowledge fixed and compares these two representation changes; their remaining scoring and granularity confounds are discussed in the Limitations.

Fielded retrieval provides a complementary way to represent records. BM25F assigns field-specific weights rather than treating a document as a single field (Robertson et al., 2004), while mFAR learns

field importance conditioned on the query (Li et al., 2025a). In our setting, the fields already exist as curated attributes of each domain record, so the question is whether adding field content and weights provides value beyond intact text, and which fields carry that value.

Knowledge graphs and structured agent memory. Graph-based retrieval represents knowledge through explicit entities and relations rather than flat text. GraphRAG builds an entity graph for corpus-level sensemaking, while HippoRAG uses a knowledge graph and personalized PageRank for long-term multi-hop retrieval (Edge et al., 2024; Gutiérrez et al., 2024). Recent work also explores structured linked representations for agent-oriented retrieval (Sade et al., 2026; Volpini et al., 2026). These approaches introduce an additional graph-construction layer, whereas fielded indexing can preserve curated record boundaries and surface forms without constructing a graph. Accordingly, we view graph-based and fielded representations as distinct points on a representation continuum rather than competing solutions.

Agent memory and memory retrieval. Agent-memory research has focused largely on how memories are created, linked, consolidated, and retrieved over time. A-MEM dynamically links and updates memories, MemBench evaluates memory across factual and reflective settings, and other systems explore hierarchical or lightweight long-term memory and structured consolidation (Xu et al., 2025; Tan et al., 2025; Cao et al., 2026a; Zhang et al., 2026; Chhikara et al., 2025). These works vary the memory-management mechanism; we hold the memory controller fixed and vary only the representation of static domain knowledge inside the retrievable memory. This addresses a lower-level question: whether the knowledge entering an agent's memory is represented in a form that retrieval can reliably use.

Bengali and low-resource retrieval. Bengali is a useful setting because the language used in domain queries can differ substantially from formal source text. Dialect and register variation produce measurable performance gaps, and resources such as Vashantor provide broad coverage of regional Bengali variation (Kantharuban et al., 2023; Faria et al., 2023; Sami et al., 2026). Recent Bengali agricultural retrieval evaluations further show that recall depends on query type, language condition, and indexing choices such as passage length (Reza et al., 2026a); KrishokChat provides a complementary citation-grounded agricultural knowledge base derived from authoritative sources (Reza et al., 2026b). These resources establish the difficulty and sensitivity of Bengali domain retrieval, but do not isolate how much of that difficulty is attributable to the representation of the knowledge itself.

Position of this work. Prior work thus shows that chunking can damage retrieval, fields can be weighted, graphs can encode relations, and agent memory can be organized in different ways. What remains unclear is how much of the apparent benefit of structured retrieval comes simply from preserving an intact record, and what additional benefit comes from adding field content and weights. We address this distinction with a representation ladder across three Bengali domains with different query regimes, and then test whether the same representation effects persist when the knowledge is consumed as memory by a retrieval-grounded agent.

3 Benchmark and Knowledge Representation

We evaluate three Bengali knowledge domains that differ in how users refer to information: legal queries identify a known act, agricultural queries are predominantly symptom-oriented, and education queries describe experimental procedures. For each domain, the same knowledge is represented at all three levels, giving 3,670 queries over 1,953 knowledge nodes.

The agricultural corpus contains 1,028 expert-curated disease and pest-management records, evaluated with 2,000 queries covering standard Dhaka, Rajshahi-dialect, and additional regional variants drawn using Vashantor as a reference (Reza et al., 2026a,b; Faria et al., 2023). Many queries describe symptoms, while others name a disease or pest; the disease-name field therefore tests lexical alignment across a mixed advisory query regime. The legal corpus contains 735 statutory acts drawn from the Bangladesh Legal Acts Dataset (Sakhawat, 2024); the 1,100 evaluation queries are derived from open-source Bengali legal QA resources and citizen-oriented statutory inquiries (Millat, 2026), with each query mapped to a validated target act. This known-item regime allows us to distinguish an identifier field from one whose surface form appears in queries. The education corpus contains

Property	Agriculture	Legal	Education
Knowledge Type	Crop disease/pest advisory	Bangladesh statutory law	Secondary science curriculum
Corpus Nodes	1,028	735	190
Unstructured Chunks	4,526	4,202	892
Evaluation Queries	2,000	1,100	570
Dialect/Register Coverage	16 regional + Standard	15 regional + Standard	5 registers + Standard
Record-Identifying Field	`disease_pest_name_bn`	`act_title_bn`	`experiment_name_bn`
Query Regime	Symptom-diagnostic	Known-item (act cited)	Procedural description

Table 1: Summary of the evaluation corpora. The query regime describes how users refer to the target information: legal queries identify a specific act, agricultural queries are predominantly symptom-oriented, and education queries describe experimental procedures.

190 secondary-science experiments from NCTB textbooks, with 570 procedural queries describing apparatus or steps without naming the experiment. Its 892 OCR-derived chunks make it a particularly direct test of fragmentation. The candidate record-identifying fields are `disease_pest_name_bn`, `act_title_bn`, and `experiment_name_bn`, respectively; their query alignment is examined in Section 6.

Query provenance. We evaluate domain-specific benchmark queries with fixed query-to-target mappings. Agricultural queries draw on KrishokChat and related advisory resources, with Vashantor as a reference for regional formulations. Legal queries draw on Bengali legal QA resources and citizen-oriented inquiries, and education queries are based on NCTB science textbooks. Source-derived query formulations may overlap lexically with target records (Appendix C).

We use a three-level representation ladder in which the underlying knowledge is unchanged while its representation and indexing granularity vary. At L_1 (Chunked Unstructured), each record is split into 500-character windows using recursive character splitting and the resulting chunks are indexed independently. At L_2 (Full-Node Unstructured), the intact record text is indexed as one document without exposing individual fields. At L_3, the intact full-text field is retained alongside individual record fields. We write "BM25F" for two BM25F-inspired implementations; neither merges term frequencies across fields as in Robertson et al. (2004). Legal and education standalone retrieval repeat field text in one BM25 document, so no field boundaries reach the scorer. Agriculture standalone retrieval and agent memory instead sum weighted per-field BM25 scores. The main agriculture result uses fixed nonuniform weights (`disease`: 3.5, `aliases`: 2.0, `symptoms`: 1.5, `content`: 1.0; other fields zeroed) set a priori from domain heuristics rather than tuned on benchmark queries; legal and education use unit weights. Implementation details are summarized in Appendix A; benchmark-fitted ceilings are separate (Appendix E).

Representation	Agri	Legal	Edu
L_1: Chunks	62.30%	73.18%	37.19%
L_2: Intact records	67.60%	79.55%	89.65%
L_3: Field-weighted	69.90%[a]	77.82%[b]	91.58%[b]

Table 2: Recall@10 across the three representation levels (ladder defined in Figure 1). [a]Fixed nonuniform sum of field scores set a priori; [b]uniform field-text repetition. All steps are significant under paired bootstrap (5,000 draws, $p \leq 0.0134$); the legal $L_2 \rightarrow L_3$ change is a *significant decrease* (95% CI $[-3.18, -0.27]$ pp). Full statistics in Appendix G.

Thus, L_3 adds individual field content to auxiliary full text, duplicating overlapping record content and changing scoring. A record spans 4.4, 5.7, and 4.7 L_1 retrieval units on average in agriculture, law, and education (Table 1). L_1 is a deliberate stress test of fixed-window retrieval; legal units are annotated passages. Stronger chunking baselines remain untested.

4 Core Retrieval Results

We compare the three representation levels using Recall@10. The primary comparison is L_1 vs. L_2 vs. L_3, distinguishing record-integrity recovery from the combined effect of field copies and weights. Tuned BM25F is treated only as a benchmark-fitted ceiling and is reported in Appendix E.

Table 2 shows that preserving the complete record produces the largest and most consistent recovery. In education, Recall@10 rises from 37.19% with chunked BM25 to 89.65% with full-node BM25, a gain of 52.46 points. Uniform BM25F

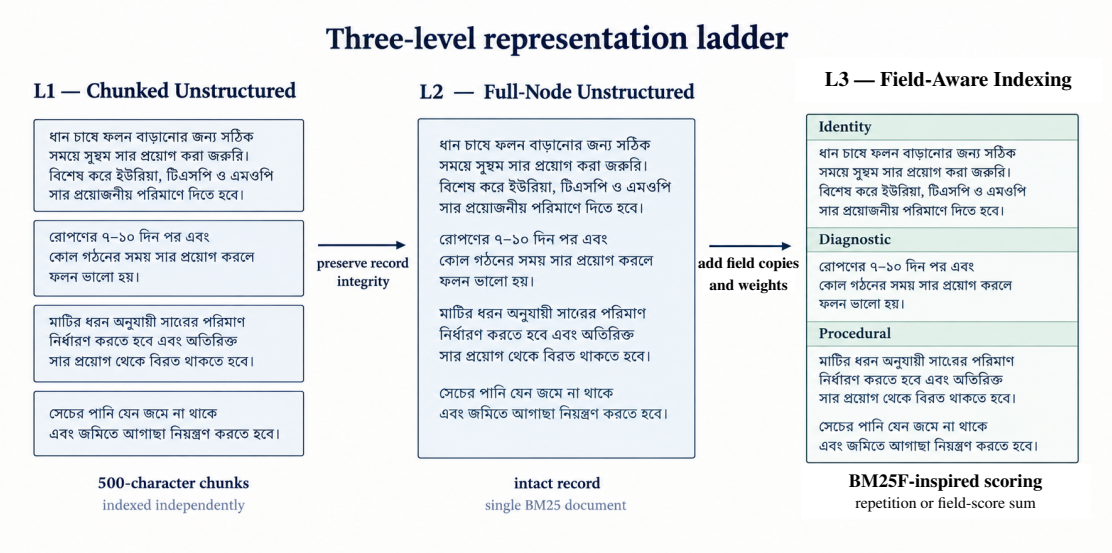


Figure 1: The three knowledge representations used in our experiments. L_1 splits each record into fixed-length chunks, L_2 keeps the record intact as a single unstructured document, and L_3 adds selected field content with BM25F-inspired scoring. The $L_1 \to L_2$ comparison measures recovery associated with preserving record integrity, while $L_2 \to L_3$ measures the combined effect of adding field copies and field weights. The field groups are schematic; L_3 also retains auxiliary full text.

adds a further 1.93 points, reaching 91.58%. The $L_1 \to L_2$ recovery is largest in the OCR-derived education corpus, where information from a single experiment can be split across chunk boundaries.

Agriculture follows the same ordering, but with smaller gains: Recall@10 increases from 62.30% to 67.60% at L_2 (+5.30 points), then to 69.90% at fixed-weight L_3 (+2.30 points). Here, both preserving the record and adding field content with fixed weights improve retrieval.

Legal retrieval is the notable exception. Full-node BM25 improves Recall@10 from 73.18% to 79.55%, whereas uniform BM25F reaches 77.82%. The paired bootstrap interval excludes zero ($\Delta = -1.73$ pp, 95% CI $[-3.18, -0.27]$ pp, $p = 0.0134$). Thus, adding unweighted field copies does not necessarily improve over intact text. The value of field-aware indexing depends on the retrieval task.

Tuned results are reported separately as benchmark-fitted ceilings in Appendix E. We next examine how often $L_1 \to L_2$ recovery coincides with cross-chunk separation.

5 Diagnosing the Fragmentation Failure Mode

Section 4 shows that preserving the complete record recovers most of what chunking loses, sharply so in education. The recovery is consistent with the motivating fragmentation hypothesis, in which query-matching evidence and the record-identifying field fall into different chunks. However, the aggregate Recall@10 gap alone does not establish this mechanism. The $L_1 \to L_2$ gain could also arise from changes in retrieval statistics, such as differences in term frequency or the greater opportunity for lexical overlap in a longer document. The analysis below measures how much of the recovery is attributable to cross-chunk separation.

Classification protocol. We inspect each query for which L_1 misses the gold target in the top 10 but L_2 retrieves it. For each such query q and its gold node r, we identify two sets of chunks among the L_1 chunks belonging to r: the *matching chunks*, those with lexical overlap with q meeting a co-occurrence threshold of $\theta = 2$ distinct token types, and the *identifier chunk*, the chunk containing the record's identifying field (Table 1). To avoid misclassifying Bengali inflectional variants, we apply Unicode token normalization and require co-occurrence across distinct query token types. We then assign each recovered query to one of three categories:

Cross-chunk separation. The matching chunks and the identifier chunk do not overlap: the query-matching evidence in the record was indexed separately from the field that identifies the record. This matches the example in Section 1; preserving an

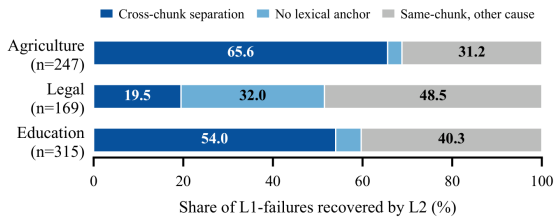


Figure 2: Why L_1 fails and L_2 recovers (n = recovered queries). Cross-chunk separation is the largest category in agriculture and education. In law, most recoveries fall in the same-chunk residual (48.5%) or lack a lexical anchor under the overlap threshold (32.0%).

intact record is one way to reunite this evidence.

No lexical anchor. No chunk of the record, including the identifier chunk, shares meaningful vocabulary with the query ($\max_j$ overlap $< \theta$). This category does not support the separated-evidence explanation. Vocabulary aggregation or changes in ranking may contribute, but the classification does not distinguish them.

Same-chunk, other cause. The matching content and the identifying field are already co-located in the same chunk, yet L_1 still missed the record. The observed matching evidence and identifier are not separated. This category leaves the ranking cause unresolved.

We compute these categories over the full recovered set in each domain, not a sample, since chunk membership and lexical overlap are both fully determined by the indexing pipeline already used to produce Table 2. Because legal L_1 evaluates passage-level chunks while L_2 evaluates act-level nodes, we collapse each act's constituent passages to their parent node before classification, so the legal diagnostic categories reported below are computed at the act level rather than the passage level.

Results. Figure 2 assigns 54.0% of education recoveries and 65.6% of agriculture recoveries to cross-chunk separation. The legal share is 19.5%; its largest category is the same-chunk residual (48.5%), followed by no lexical anchor under the threshold (32.0%). Same-chunk cases also account for 31.2% in agriculture and 40.3% in education. Length normalization, candidate competition, and other ranking effects may explain these cases, but this analysis does not identify their causes. The recovered sets count gains only: L_2 may also lose queries retrieved by L_1.

Counterfactual Condition	Agri	Legal	Edu
Unshuffled Reference	69.90%	77.82%	91.58%
Full Schema Shuffle	0.75%	1.91%	5.09%
Content-Only Shuffle†	56.90%	87.55%	87.02%
Identity-Only Shuffle†	47.55%	84.64%	94.04%

Table 3: Field-value shuffling (Recall@10). †Legal and education partial shuffles use different scoring configurations and are not comparable to their reference values.

6 Field-Level Analysis

The previous results show when fielded indexing helps but not which fields carry that advantage. We examine this at the field level through group ablations and counterfactual permutations, testing whether the candidate query fields identified in Table 1 actually behave as query-aligned anchors.

6.1 Field-Group and Sub-Field Analysis

We group the schema by function (full schema in Appendix A) and report three domain-relevant field groups (Identity, Diagnostic, and Procedural) defined by domain-specific schema roles. Exact group membership and sub-field ablations are in Appendix B.

The strongest reported group differs by domain. Identity reaches 98.09% R@10 in legal retrieval, while Identity and Diagnostic reach 50.15% and 27.25% in agriculture. In education, Procedural fields are strongest at 78.77% versus 64.91% for Identity. The legal and education patterns accord with their query regimes. In agriculture, the disease-name field alone reaches 52.50%, exceeding the Diagnostic group. The education results do not support treating the experiment name as its dominant retrieval anchor.

Legal sub-field results provide a direct illustration of query alignment: `act_title_bn` alone reaches 100.00% R@10, while `act_no_bn` reaches only 2.18%. Both identify the same acts, but only the title appears in the surface form used by queries. We define a *query-aligned anchor* as a field whose content provides a direct lexical link to the query formulation.

6.2 Perturbation Attribution

We permute field values across records while retaining record identifiers (Table 3). Identity-Only shuffles the designated name or title field; Content-Only shuffles the auxiliary full-text field. Full shuffling reduces Recall@10 to 0.75–5.09%, showing that field labels alone are insufficient without the

correct content. In agriculture, shuffling either the name or full text lowers recall. Legal and education partial shuffles use different scoring configurations from the displayed references, so we do not interpret those values as gains or losses against them.

7 Structured Knowledge as Agent Memory

The retrieval experiments show that representation affects which records are found and that this effect may depend on query-aligned fields. We next ask whether representation differences remain when an LLM agent uses the knowledge store as external memory.

7.1 Controlled Agent Protocol

We evaluate a minimal retrieval-grounded agent to hold the memory controller constant across conditions. The agent uses `gemini-2.5-flash-lite` through OpenRouter with one tool, `retrieve_memory(query, k=5)`, and at most two retrieval calls per query. It must retrieve before answering, use only retrieved evidence, cite supporting node identifiers, and return "insufficient evidence" when the memory does not support an answer. The full prompt and tool schema are in Appendix D.

We evaluate four memory conditions: M_1 chunked BM25 (L_1), M_2 full-node BM25 (L_2), M_3^u structured uniform BM25F (L_3, $w_f = 1.0$), and M_3^t structured tuned BM25F (L_3, benchmark-informed weights extending the a priori core; see Appendix E and the ceiling discussion in Section 4). We first sampled 300 queries per domain with seed 42: agriculture by dialect bucket, legal by difficulty, and education by difficulty. Excluding 28 queries without common completed runs leaves 872 (286/289/297 by domain). Abstentions count as retrieval misses. Legal M_1 returns passage identifiers while gold targets use act identifiers; we therefore exclude it from performance comparisons and evaluate $M_1 \rightarrow M_2$ on the remaining 583 queries. Other memory comparisons use all 872. Metrics use the first five unique retrieved identifiers across calls. Citation rate is the fraction of all evaluated queries whose answer contains a `[Node: <id>]` marker; abstentions remain in the denominator, and neither identifier validity nor entailment is checked.

7.2 Representation Transfer to Agent Memory

On agriculture and education ($n = 583$), full-node memory raises Recall@5 from 27.6% to 61.9% (+34.31 pp; paired McNemar $\chi^2 = 186.92$, $p = 1.50 \times 10^{-42}$; discordant pairs $b = 207$, $c = 7$).

Across all three domains ($n = 872$), uniform fielded memory reaches 55.4% Recall@5 versus 56.4% for full-node memory (McNemar $p = 0.292$). Recall@1 is numerically higher (0.304 to 0.322; $p = 0.157$). Citation rate increases from 48.2% to 60.4% (+12.27 pp; McNemar $\chi^2 = 54.78$, $p = 1.35 \times 10^{-13}$, $b = 158$, $c = 51$). These tests are uncorrected exploratory comparisons. The tuned condition reaches 80.5% Recall@5, indicating headroom under benchmark-informed weighting rather than a held-out gain.

Domain patterns differ. In law, Recall@5 is 45.3% for full-node, 48.8% for uniform, and 98.6% for tuned memory; uniform Recall@1 rises from 32.5% to 41.9%. Agriculture drops from 56.3% to 49.3% under uniform fields, then reaches 62.6% with tuning. Education is similar under full-node and uniform memory (67.3% and 67.7%). These observations do not identify a single ranking mechanism shared by all domains.

7.3 Anchor-Field Perturbation in Agent Memory

We conduct an exploratory perturbation study on a separate subset of 100 queries per domain, sampled with seed 42 from pools disjoint from the original agent sample using the same strata. The separate unshuffled run reaches 60.33% Recall@5 and invokes the tuned M_3 backend. It is not pooled with the primary memory comparison.

Here, Anchor shuffles disease names and Bengali aliases in agriculture, act titles and short titles in law, and procedure steps and observation/conclusion fields in education. Content shuffles the configured non-anchor fields, including auxiliary full text. These sets differ from the standalone single-field controls. The perturbed backend also serializes fewer context fields than the unshuffled backend, confounding the answer-level comparison.

Under the same perturbed backend, Recall@5 is 51.33% with content fields shuffled and 35.67% with anchor fields shuffled. This 15.7-point gap is consistent with sensitivity to query-aligned content. Rendering differs from the unshuffled run

Condition	Memory Representation	n	R@1	R@5	MRR@5	Avg. Tokens	Citation Rate
M_1	Chunked BM25 (Agri + Edu)	583	0.158	0.276	0.205	418	50.1%
M_2	Full-node BM25 (Agri + Edu)	583	0.293	0.619	0.418	778	60.7%
M_2	Full-node BM25 (all domains)	872	0.304	0.564	0.404	677	48.2%
M_3^u	Uniform fields (all domains)	872	0.322	0.554	0.411	749	60.4%
M_3^t	Tuned fields (all domains; ceiling)	872	0.548	0.805	0.644	1,015	83.7%

Table 4: Agent memory performance. The first two rows compare agriculture and education only ($n = 583$); legal M_1 is excluded because of incompatible passage/act identifiers. The final three rows use the same three-domain subset ($n = 872$). Tuned fields are an exploratory ceiling. Citation rate measures marker presence, not validated attribution or entailment. Differences use unrounded results.

Memory State (n=300)	R@5	MRR@5	Citation
Unshuffled Structured Run	0.603	0.477	63.3%
Content-Field Shuffled	0.513	0.415	40.3%
Anchor-Field Shuffled	**0.357**	**0.197**	**37.0%**

Table 5: Exploratory agent perturbations on a separate 300-query subset. Bold indicates maximum degradation (lowest performance under perturbation). Anchor-shuffled Recall@5 is 15.7 pp below content-shuffled recall under the same perturbed backend. Field selections and context rendering differ from the primary study; see text. Citation denotes citation rate.

(60.33%), so losses against that baseline are not interpreted. In law, Recall@1 is 58.0% under content shuffling and 9.0% under anchor shuffling. In education, the perturbed anchor content is procedural rather than the experiment name.

8 Discussion and Conclusion

Keeping records intact consistently improves standalone retrieval over the strict chunked baseline, while the additional value of field-aware indexing varies by domain. Cross-chunk separation accounts for most agriculture and education recoveries, while most legal recoveries fall into other categories. Field ablations suggest that useful content follows query formulation: statutory titles in law, disease names in agriculture, and procedures in education. Partial permutations do not establish a unique role for record identifiers.

The agent study also suggests that representation warrants attention in memory design. The chunked-to-intact comparison covers agriculture and education; the three-domain comparison finds a higher citation rate with uniform fields without higher aggregate Recall@5. Neither metric establishes answer correctness. Supplementary dense results are consistent with representation sensitivity but do not evaluate the full ladder (Appendix F).

Fielded records can be viewed as self-contained units with typed attributes. We build no graph and test no graph-based retriever, so these results do not measure the value of links or multi-hop reasoning. Whether graph connectivity adds gains after controlling for record integrity, scoring, and query alignment remains open.

For knowledge already organized into records, indexing intact records is the first design choice to check. Field-aware weighting can help when weighted fields match query vocabulary and can lower recall otherwise. Stronger chunking baselines, canonical BM25F, and other languages are the next tests.

Limitations

Scope. We study three Bengali domains with curated records and make no claim for other languages, domains, or unstructured sources. Dense and hybrid results cover only chunked and structured-flat representations. No graph or inter-record relation is evaluated.

Representation comparisons. Stronger chunking baselines may narrow the reported gaps. Preserving records also changes document length, candidate count, and BM25 normalization; legal relevance changes from passages to acts. At L_1, Recall@10 counts ten ranked chunks without parent-node deduplication, so repeated parents occupy separate slots. Field-aware indexing duplicates overlapping content through auxiliary full text and individual fields. Neither scoring implementation is canonical BM25F, and tuned results are benchmark-fitted ceilings.

Fragmentation diagnosis. The diagnostic categories depend on one co-occurrence threshold ($\theta = 2$); no sensitivity analysis was run.

Queries. Source-derived query formulations may overlap with target-record vocabulary and may not fully represent naturally occurring user inquiries.

Regional formulations have not undergone a dedicated speaker-based authenticity evaluation.

Agent study. We evaluate one backbone with static memory. Legal chunked-agent results are excluded because of an identifier mismatch. Perturbation results are exploratory because configuration and context rendering differ from the primary study. Retrieval and citation presence do not establish answer correctness or semantic support.

Ethical Considerations

The agricultural records include pesticide guidance. These experiments do not evaluate the correctness or safety of chemical advice; preserving record integrity has not been validated as a safety intervention. The agents are research prototypes, and pesticide recommendations require expert review before use. Agricultural query material was reused from KrishokChat and related advisory resources (Reza et al., 2026b,a). Code is released under MIT. Reused datasets, queries, and source texts remain subject to their original licenses and redistribution conditions.

Data and Code Availability

Code and derived benchmark artifacts are available on GitHub (`https://github.com/RaiyaanReza/Beyond-Chunking-Node-Integrity-and-Field-Structure-in-Bengali-Retrieval-Augmented-Agent-Memory`) and Hugging Face (`https://huggingface.co/datasets/RaiyanKhaan/Beyond-Chunking`). Release documentation records provenance and redistribution constraints for third-party materials.

Acknowledgements

We acknowledge assistance from Claude, OpenAI Codex, and Perplexity with manuscript editing, LaTeX preparation, consistency checks, and statistical verification, as detailed in the following declaration.

Declaration on Generative AI

Claude was used for LaTeX editing and manuscript review. OpenAI Codex assisted with LaTeX editing, language revision, manuscript checks, and statistical checks of existing experiment logs. Perplexity assisted with manuscript review, consistency checks, and verification of reported statistics. The evaluated agent backbone is `gemini-2.5-flash-lite`. The authors are responsible for the experimental design, results, and final manuscript.

References

Shuqi Cao, Jingyi He, and Fei Tan. 2026a. HiGMem: A hierarchical and LLM-guided memory system for long-term conversational agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 33853–33862. Association for Computational Linguistics.

Tianyu Cao, Neel Bhandari, Akhila Yerukola, Akari Asai, and Maarten Sap. 2026b. Out of style: RAG's fragility to linguistic variation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 280–318, Rabat, Morocco. Association for Computational Linguistics.

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*.

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*.

Fatema Tuj Johora Faria, Mukaffi Bin Moin, Ahmed Al Wase, Mehidi Ahmmed, Md. Rabius Sani, and Tashreef Muhammad. 2023. Vashantor: A large-scale multilingual benchmark dataset for automated translation of Bangla regional dialects to Bangla language. *Preprint*, arXiv:2311.11142.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Anjali Kantharuban, Ivan Vulić, and Anna Korhonen. 2023. Quantifying the dialect gap and its correlates across languages. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7226–7245, Singapore. Association for Computational Linguistics.

Millicent Li, Tongfei Chen, Benjamin Van Durme, and Patrick Xia. 2025a. Multi-field adaptive retrieval. In *International Conference on Learning Representations (ICLR)*. Spotlight.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2025b. StructRAG: Boosting knowledge intensive reasoning of LLMs via inference-time hybrid information structurization. In *The Thirteenth International Conference on Learning Representations*.

M. Millat. 2026. BDLawCorpus-Dataset-V1. https://huggingface.co/datasets/millat/BDLawCorpus-Dataset-V1.

Thong Nguyen, Shubham Chatterjee, Sean MacAvaney, Iain Mackie, Jeff Dalton, and Andrew Yates. 2024. DyVo: Dynamic vocabularies for learned sparse retrieval with entities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 767–783.

Khan Raiyan Ibne Reza, Sanjana Aktar Maria, and Sumaiya Tabassum Nimi. 2026a. Where does retrieval fail? Evaluating RAG architectures for agricultural advisory. *Preprint*, arXiv:2608.14886.

Khan Raiyan Ibne Reza, Sumaiya Tabassum Nimi, and Omar Ibne Shahid. 2026b. KrishokChat: A provenance-traceable multi-task Bengali agricultural benchmark with safety-critical chemical advisory. *Preprint*, arXiv:2606.29243.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple BM25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49.

Yoav Kor Sade, Arvindh Arun, Rishi Puri, Steffen Staab, and Maya Bechler-Speicher. 2026. Ex-GraphRAG: Interpretable evidence routing for graph-augmented LLMs. *arXiv preprint arXiv:2605.21994*.

Adib Sakhawat. 2024. Bangladesh legal acts dataset. https://huggingface.co/datasets/sakhadib/Bangladesh-Legal-Acts-Dataset.

K. M. Jubair Sami, Dipto Sumit, Ariyan Hossain, and Farig Sadeque. 2026. Benchmarking Bengali dialectal bias: A multi-stage framework integrating RAG-based translation and human-augmented RLAIF. *Preprint*, arXiv:2603.21359.

Shalin Shah, Srikanth Ryali, and Ramasubbu Venkatesh. 2026. SRAG: RAG with structured data improves vector retrieval. *arXiv preprint arXiv:2603.26670*.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19336–19352. Association for Computational Linguistics.

Andrea Volpini, Elie Raad, Beatrice Gamba, and David Riccitelli. 2026. Structured linked data as a memory layer for agent-orchestrated retrieval. *arXiv preprint arXiv:2603.10700*.

Nengbo Wang, Xiaotian Han, Jagdip Singh, Jing Ma, and Vipin Chaudhary. 2025. CausalRAG: Integrating causal graphs into retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22680–22693. Association for Computational Linguistics.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Zhenzhen Huang, Pengcheng Zheng, Zhicheng Wang, Ping Guo, Fan Mo, Sung-Ho Bae, Jie Zou, Jiwei Wei, and Yang Yang. 2026. Lightweight LLM agent memory with small language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12914–12929. Association for Computational Linguistics.

A Full Knowledge Schema and Field Functional Mapping

To illustrate field structure, semantic roles, and representations across the evaluation ladder, Table 6 summarizes 25 domain fields and three auxiliary full-text entries across the three evaluation corpora.

Functional groups. Identity, Diagnostic, and Procedural are domain-specific ablation labels, not a universal taxonomy. Exact membership for the reported groups is listed in Appendix B; auxiliary full text is separate from these group ablations.

Implementation. Standalone legal and education retrieval apply field weights through text repetition in a single BM25 document, while agriculture and agent memory sum weighted per-field BM25 scores. Both retain auxiliary full text alongside individual fields, duplicating overlapping content rather than isolating field boundaries. The underlying legal corpus also indexes supplementary statutory regime metadata (`legal_system_bn`), evaluated under the Legal Diagnostic group in Appendix B; Table 6 and agent memory (Table 10) focus on substantive statutory provisions. The shuffle controls include different scoring configurations, so not every row is a matched comparison. Exact field inventories and weights are in the released code.

B Supplementary Ablation Details

Table 7 reports BM25 retrieval over concatenated group fields and individual identity fields. The exact group memberships are:

Domain	Field Identifier	Functional Group	Field Semantic Scope and Contents
Agriculture (10 Domain Fields + 1 Auxiliary; 1,028 Knowledge Nodes)			
Agriculture	disease_pest_name_bn	Identity	Canonical Bengali disease or pest name
Agriculture	colloquial_aliases	Identity	Regional and vernacular names
Agriculture	scientific_name	Identity	Latin taxonomic name
Agriculture	symptoms	Diagnostic	Symptoms affecting leaves, stems, and fruit
Agriculture	favorable_conditions_bn	Diagnostic	Temperature, humidity, and rainfall conditions
Agriculture	management	Procedural	Cultural, mechanical, and biological control
Agriculture	prevention	Procedural	Preventive cultivation and seed treatment
Agriculture	chemical_management_bn	Procedural	Chemical ingredients and application guidance
Agriculture	affected_crop_bn	Contextual	Susceptible crop and host species
Agriculture	safety_advisory	Safety / Metadata	Hazard warnings, protective equipment, harvest intervals
Agriculture	content_bn	Auxiliary (full text)	Concatenated record text (auxiliary field)
Legal (10 Domain Fields + 1 Auxiliary; 735 Statutory Acts)			
Legal	act_title_bn	Identity	Official act title
Legal	short_title_bn	Identity	Short title or commonly cited name
Legal	act_no_bn	Identity	Act number and enactment year
Legal	preamble_bn	Contextual	Legislative purpose and policy context
Legal	definitions_bn	Diagnostic	Definitions of legal terminology
Legal	section_titles_bn	Content	Section titles and subject headings
Legal	powers_bn	Procedural	Administrative and executive powers
Legal	procedures_bn	Procedural	Statutory procedures and regulatory protocols
Legal	penalties_bn	Procedural	Fines, penalties, and imprisonment terms
Legal	status_bn	Contextual	In-force status, amendments, and repeal metadata
Legal	content_bn	Auxiliary (full text)	Concatenated record text (auxiliary field)
Education (5 Domain Fields + 1 Auxiliary; 190 Secondary Curriculum Experiments)			
Education	experiment_name_bn	Identity	Textbook experiment title
Education	apparatus_and_chemicals	Identity	Required apparatus and reagents
Education	observation_conclusion	Diagnostic	Observations and scientific conclusions
Education	procedure_steps	Procedural	Experimental procedure
Education	precautions	Safety / Metadata	Laboratory safety and experimental precautions
Education	full_text_bn	Auxiliary (full text)	Concatenated record text (auxiliary field)

Table 6: Semantic summary of domain fields and auxiliary full text. Record identifier fields are named in Table 1; procedures are stronger than the name field in education. Groups follow the domain-specific ablations. The table summarizes core semantic fields, rather than every indexed key (e.g., supplementary statutory regime metadata legal_system_bn is evaluated in Appendix B); exact field inventories are in the released code. Auxiliary text overlaps individual fields.

Agriculture. Identity: disease_pest_name_bn, scientific_name, colloquial_aliases; Diagnostic: symptoms, favorable_conditions_bn; Procedural: management, prevention, chemical_management_bn.

Legal. Identity: act_title_bn, short_title_bn, act_no_bn; Diagnostic: definitions_bn, legal_system_bn; Procedural: procedures_bn, powers_bn, penalties_bn.

Education. Identity: experiment_name_bn, apparatus_and_chemicals; Diagnostic: observation_conclusion; Procedural: procedure_steps.

Functional Group	R@1	R@5	R@10	MRR
Agriculture (Best Tuned BM25F: 72.35%)				
Identity Group	20.90%	41.40%	50.15%	0.296
Diagnostic Group	8.90%	21.00%	27.25%	0.142
Procedural Group	5.20%	10.60%	14.30%	0.076
Legal (Best Tuned BM25F: 100.00%)				
Identity Group	94.70%	97.90%	98.09%	0.962
Procedural Group	36.50%	64.70%	72.73%	0.489
Diagnostic Group	23.40%	39.70%	44.91%	0.305
Education (Best Tuned BM25F: 93.33%)				
Procedural Group	27.00%	68.20%	78.77%	0.449
Diagnostic Group	22.30%	57.90%	67.89%	0.369
Identity Group	14.40%	48.90%	64.91%	0.294
Identity Sub-Field Standalone Retrieval (R@10)				
Agri: disease_pest_name_bn		**52.50%**		0.325
Agri: colloquial_aliases		10.15%		0.055
Agri: scientific_name		1.55%		0.007
Legal: act_title_bn		**100.00%**		0.995
Legal: short_title_bn		91.00%		0.893
Legal: act_no_bn		2.18%		0.015

Table 7: Field-group and identity sub-field BM25 ablation metrics across domains; exact group memberships are listed above.

C Query Construction and Representative Benchmark Examples

To provide concrete visibility into query formulation, linguistic register diversity, and the relation between user queries and memory fields, Table 8 presents nine representative benchmark queries across Agriculture, Legal, and Education.

Corpus provenance and query construction. Benchmark queries were constructed from domain-specific source materials and query-generation procedures designed to reflect the target query regimes:

- **Agriculture (2,000 queries):** Drawn from Bengali agricultural advisory resources and KrishokChat, with Vashantor as a reference for regional formulations (Reza et al., 2026a,b; Faria et al., 2023). Queries span 16 distinct regional dialects alongside Standard Dhaka Bengali. In the symptom-diagnostic regime, queries describe crop symptoms or explicitly name diseases and pests, as illustrated by Agri-02 and Agri-03.
- **Legal (1,100 queries):** Compiled from statutory question-answering resources and citizen inquiries (Millat, 2026). Queries cover 15 regional dialect formulations alongside formal standard legal Bengali. In this known-item regime, inquiries identify target statutes by name or legislative topic while seeking interpretations of specific provisions, administrative duties, or penal liabilities.
- **Education (570 queries):** Developed from secondary science curricula (NCTB textbooks, grades 9–10 physics, chemistry, and biology). Queries cover 5 student/colloquial registers alongside standard textbook prose. In this procedural regime, students articulate hands-on experimental actions, glassware setups, or observed physical changes without recalling formal scientific nomenclature.

Query provenance and target mappings. The evaluation uses the benchmark's fixed query-to-target mappings across representations. Agricultural queries draw on KrishokChat and related advisory resources (Reza et al., 2026b,a); legal and education provenance is described above. Source-derived formulations may share vocabulary with their target records. This study does not establish dialectal authenticity through a dedicated speaker-based evaluation.

Interpretation of the examples. Legal queries explicitly cite act names, supporting the strong title-only result. Cross-chunk separation nevertheless accounts for only 19.5% of legal recoveries, so the examples do not explain the aggregate legal gap. Education queries mention apparatus and actions; Procedural fields outperform Identity in the group ablation. The examples illustrate how query terms can connect to titles, names, and procedural content.

D Agent Memory Prompt and Tool Schema

The retrieval-grounded agent is governed by the system instructions and tool interface shown below:

System Prompt (Agent Instructions):
"You are a factual knowledge assistant. When presented with a question, you must consult your external memory using `retrieve_memory(query, k=5)` before formulating your answer. Answer only using evidence explicitly confirmed by the retrieved records. Cite supporting node identifiers using `[Node: <id>]`. If the retrieved memory does not contain sufficient facts to answer the question, explicitly state 'insufficient evidence'. Do not extrapolate or guess."

Tool Interface:
`retrieve_memory(query: str, k: int = 5) -> List[Dict]`
Returns top-k documents formatted with node identifiers and textual content matching the active memory condition (M1, M2, or M3).

E Field-Weight Tuning Protocol

Configuration	Agriculture	Legal	Education
L_3 Tuned BM25F (Ceiling)†	72.35%	100.00%	93.33%

Table 9: Benchmark-fitted tuned BM25F Recall@10 ceiling. †Weights were selected on the full benchmark without a held-out split; this is a fitted reference, not a formal upper bound.

Protocol and disclosure. Only per-field weights w_f and length-normalization hyperparameters (k_1, b; education best: k_1=3.0, b=0.50) were tuned. Grid search selected the configuration maximizing Recall@10 on each domain's full query pool (ties broken by MRR). Because weights were selected on the full benchmark without a held-out development split, these values represent an empirical ceiling rather than generalizable out-of-sample estimates. In contrast, the standalone L_3 agriculture baseline (69.90% in Table 2) used fixed heuristic weights (`disease`: 3.5, `aliases`: 2.0,

Domain & ID	Dialect / Register	Query Formulation (Romanized Bengali & English)	Gold Target Node	Record Identifier Field & Value
Agriculture: Symptom-Diagnostic and Vernacular Inquiries				
Agri-01	Rajshahi Dialect	*"Gacher pata keno emon holde hoiya kukraiya jaitechhe, eita kiser lokkhon?"* *Gloss:* "Why are the leaves turning yellow and curling up like this? What is this a symptom of?"	`AT_tomato_yellow_leaf_curl_01`	**Field:** `disease_pest_name_bn` **Value:** *Tomator holde pata kukrano rog* (Tomato yellow leaf curl virus)
Agri-02	Barisal Dialect	*"Amne kon dehi, ei tungro rogda kemne chhoray ar gache ki hoi?"* *Gloss:* "Tell me, how does this Tungro disease spread and what happens to the plant?"	`AT_rice_disease_control_040_part1`	**Field:** `disease_pest_name_bn` **Value:** *Dhaner tungro rog* (Rice tungro disease)
Agri-03	Standard Dhaka	*"Ei Hydronomidius molitor pokata kon kon gache beshi dhore?"* *Gloss:* "Which host plants does this Hydronomidius molitor insect commonly infest?"	`AT_extraction_all_pest_books_part363`	**Field:** `disease_pest_name_bn` **Value:** *Hydronomidius molitor*
Legal: Known-Item Statutory and Citizen Consultations				
Legal-01	Standard Formal	*"'Parbotya Zila Sthanio Sarkar Parishad Ain, 1989'-er 7 nombor dharati shohoj bhashay byakhya korun."* *Gloss:* "Explain Section 7 of the 'Hill District Local Government Parishad Act, 1989' in plain language."	`legal_act_0760`	**Field:** `act_title_bn` **Value:** *Parbotya Zila Sthanio Sarkar Parishad Ain, 1989*
Legal-02	Barisal Dialect	*"Hindudharmio Kalyan Trust Ain, 2018 er 16 nombor dharada shohoj koira bujhaiya koben ni?"* *Gloss:* "Could you easily explain Section 16 of the Hindu Religious Welfare Trust Act, 2018?"	`legal_act_1236`	**Field:** `act_title_bn` **Value:** *Hindudharmio Kalyan Trust Ain, 2018*
Legal-03	Bogra Dialect	*"Rangpur Biggan o Projukti Bishwobiddyaloy Ain, 2001-er 58 nombor dharay ki ki aini shomosshya hober pare?"* *Gloss:* "What legal complications might arise under Section 58 of the Rangpur Science and Tech Univ Act, 2001?"	`legal_act_0901`	**Field:** `act_title_bn` **Value:** *Rangpur Biggan o Projukti Bishwobiddyaloy Ain, 2001*
Education: Procedural and Apparatus Descriptions				
Edu-01	Student Colloquial	*"Panite rong meshale ki hoy, eta kibhabe dekhbo?"* *Gloss:* "What happens when dye is mixed in water, and how can we observe this?"	`edu_biology_0001`	**Field:** `experiment_name_bn` **Value:** *Panite tontur byapon porjobekkhon* (Observation of diffusion in water)
Edu-02	Jessore Dialect	*"Macher potka diye chinir drobon upore othar porikkha-ta kemne kore?"* *Gloss:* "How is the experiment of sugar solution rising using a fish bladder conducted?"	`edu_biology_0013`	**Field:** `experiment_name_bn` **Value:** *Otimatraboner porikkha: Thistle fanel o jontobojhilli shahajje obhishrobon* (Observation of osmosis using a thistle funnel and animal membrane)
Edu-03	School Lab Register	*"Onubikkhon jontre peyajer roshal sholkopotro diye kosh dekhar niom ki?"* *Gloss:* "What is the procedure for observing cells from an onion scale leaf under a microscope?"	`edu_biology_0006`	**Field:** `experiment_name_bn` **Value:** *Onubikkhon jontrer shahajje udvidkosh (peyajer kosh) porjobekkhon* (Observation of plant cells (onion cells) under a microscope)

Table 8: Representative benchmark queries across domains, illustrating query regimes, dialectal registers, gold target nodes, and record identifier fields. Queries and identifier values are presented in Romanized transliteration alongside English glosses.

`symptoms`: 1.5, `content`: 1.0) set a priori before evaluation without query tuning. Agent weights (Table 10) start from this a priori set for core fields and add benchmark-informed nonzero weights for secondary fields. The agent settings have no agent-time tuning and differ from standalone configurations.

Candidate grids. Agriculture scanned 9 weight configurations over (`content_bn`, `disease_pest_name_bn`, `colloquial_aliases`, `symptoms`, `management`) spanning identity-boost, diagnostic-boost, uniform, content-heavy, extreme-identity, and scaled-down families; the a priori set sat inside this grid as the baseline candidate (69.90% vs. 72.35% ceiling). Legal and education use domain-specific weight and hyperparameter grids. Table 10 specifies the fixed agent settings separately from the standalone optima.

Domain	Fixed Agent Field Weights (w_f)
Agriculture	`disease_pest_name_bn`: 3.5, `colloquial_aliases`: 2.0, `symptoms`: 1.5, `content_bn`: 1.0, `favorable_conditions_bn`: 1.0, `management`: 0.8, `prevention`: 0.8, `chemical_management_bn`: 0.5, `affected_crop_bn`: 0.5
Legal	`act_title_bn`: 5.0, `short_title_bn`: 2.0, `preamble_bn`: 1.0, `content_bn`: 1.0, `definitions_bn`: 0.8, `section_titles_bn`: 0.8, `act_no_bn`: 0.5, `penalties_bn`: 0.5, `powers_bn`: 0.5, `procedures_bn`: 0.5
Education	`procedure_steps`: 3.0, `experiment_name_bn`: 2.0, `observation_conclusion`: 1.5, `full_text_bn`: 1.0, `apparatus_and_chemicals`: 0.8, `precautions`: 0.5

Table 10: Fixed agent field weights for M_3^t. The backend uses per-field BM25 with default $k_1 = 1.5$, $b = 0.75$ in each domain; the education standalone optimum ($k_1 = 3.0$, $b = 0.50$) is separate. Agriculture agent weights start from the a priori set for core fields and add benchmark-informed nonzero weights for secondary fields. Uniform M_3^u assigns unit weights to these same listed fields. Other serialized fields (including `status_bn` and `legal_system_bn`) are not indexed for agent retrieval. In the agent implementation backend, agriculture fields match the schema in Table 6 and may be keyed with the `_bn` suffix (e.g., `colloquial_aliases_bn`, `symptoms_bn`, `management_bn`, and `prevention_bn`).

F Dense and Hybrid Retrieval Results

Table 11 reports dense, hybrid, and flat unweighted baselines. Structured-flat BM25 concatenates selected record fields, including auxiliary full text, without field weights; it differs from the full-text-only standalone L_2, and field selection differs from Table 2. Dense retrieval embeds that flattened record as a single passage, rather than computing separate field embeddings. Hybrid retrieval fuses sparse and dense rankings with reciprocal rank fusion. While these models do not evaluate the full $L_1 \to L_2 \to L_3$ decomposition across all ablation conditions, their differences are consistent with representation sensitivity beyond sparse retrieval, without separating record integrity from field content.

Level	Retriever	Agri	Legal	Edu
L_1 Chunked	BGE-M3*	62.95%	81.91%	38.42%
L_1 Chunked	mE5-small*	28.65%	n/a†	20.00%
L_1 Chunked	Hybrid RRF	65.60%	n/a†	26.32%
Record fields, flattened	BM25	64.55%	77.82%	91.40%
Record fields, flattened	BGE-M3*	68.40%	96.36%	93.86%
Record fields, flattened	mE5-small*	41.30%	91.73%	77.37%
Record fields, flattened	Hybrid RRF	73.05%	99.00%	87.72%

Table 11: Supplementary dense, hybrid, and flattened-record retrieval results (Recall@10), omitted from the main comparison in Table 2. †BGE-M3 serves as primary dense representative for legal; chunked mE5/Hybrid omitted there. *Evaluated zero-shot without domain fine-tuning.

G Statistical Significance and Bootstrap Confidence Intervals

Table 12 reports the complete paired statistical significance testing matrix for the core representation ladder (Table 2) across all three evaluation domains. For each pairwise comparison ($L_1 \to L_2$, $L_2 \to L_3$, and $L_1 \to L_3$), we report the empirical metric change (Δ in Recall@10 percentage points), the 95% confidence interval estimated via 5,000 paired bootstrap resamples with replacement, and the empirical paired bootstrap tail probability in the direction of the observed change.

All nine reported ladder comparisons have unadjusted $p \leq 0.0134$ and remain significant under Holm correction at $\alpha = 0.05$. In law, $L_2 \to L_3$ is a decrease of 1.73 pp; its 95% interval $[-3.18, -0.27]$ pp excludes zero, with its upper bound below zero. Values at the bootstrap resolution floor are reported as $p \leq 0.0002$; 0.0002 is the resolution floor for 5,000 draws.

Domain	Comparison	Δ (pp)	95% CI (pp)	p-value
Agri (n=2,000)	$L_1 \to L_2$	+5.30	[+3.65, +6.95]	≤ 0.0002
	$L_2 \to L_3$	+2.30	[+0.65, +3.95]	0.0062
	$L_1 \to L_3$	+7.60	[+5.90, +9.30]	≤ 0.0002
Legal (n=1,100)	$L_1 \to L_2$	+6.36	[+3.45, +9.09]	≤ 0.0002
	$L_2 \to L_3$	−1.73	[−3.18, −0.27]	0.0134
	$L_1 \to L_3$	+4.64	[+1.82, +7.45]	0.0004
Edu (n=570)	$L_1 \to L_2$	+52.46	[+47.89, +56.84]	≤ 0.0002
	$L_2 \to L_3$	+1.93	[+0.35, +3.68]	0.0116
	$L_1 \to L_3$	+54.39	[+50.00, +58.60]	≤ 0.0002

Table 12: Complete paired bootstrap significance matrix for Recall@10 (5,000 draws). All nine comparisons remain significant after Holm correction at $\alpha = 0.05$; displayed p-values are unadjusted. The resolution floor for 5,000 draws is 0.0002.

KG-SFT Contracts: Auditing Evidence Lineage and Loader Retention in Enterprise RAG Adaptation

Bogdan Roshchupkin[1] rosupkinbogdan6@gmail.com
Alexander Litvintsev[1] ailitvintsev@gmail.com
Alina Avanesyan[2] aavanesyan@hse.ru
Maria Khodorchenko[3] mkhodorchenko@itmo.ru
Andrei Zhukov[1] breballov@gmail.com
[1]Sber [2]Higher School of Economics, Moscow, Russia [3]ITMO University

Abstract

KG-assisted supervised fine-tuning depends on evidence selection, context compilation, and data loading. We present KG-SFT Contracts, a provenance-aware interface that audits these stages and links model comparisons to their data and scoring records. In a retrospective Russian enterprise case, a pool of 484 graph candidates contributed no graph-origin distractors to the compiled corpus, and exact loader replay discarded 40 of 946 intended examples before splitting. Removing annotated non-support context restored every row while preserving questions, targets, and verified support. Downstream comparisons remained statistically inconclusive. Controlled checks on a public graph-bearing fixture show how cross-record validation detects unresolved graph references and conflicting source revisions alongside role-aware repair. The installable implementation includes fixture-level CPU measurements, incremental adoption guidance, and an official RuBQ schema-import check. The case illustrates how lineage and retention audits make adaptation pipelines inspectable before model effects are attributed to their configured inputs.

1 Introduction

RAG adaptation trains with relevant and distracting documents (Zhang et al., 2024; Kachuee et al., 2025), while KG-guided SFT extracts subgraphs or builds graphs for synthetic supervision (Chen et al., 2025a,b). Yet enabling a graph-candidate channel does not show that graph-derived chunks entered the corpus, compiled rows survived the actual loader, or a later metric difference came from that channel. We audit this chain when graph records supply context candidates during *offline* SFT dataset construction.

Consider the synthetic Service Atlas: its v2 runbook requires OAuth2, while a graph-linked v1 runbook specifies an API key. The older passage is a useful hard negative only if the row preserves v2 support, labels v1 as a distractor, and survives loading. Lost provenance prevents source attribution; a dropped row removes the intended lesson.

Three silent failures separate configuration from realized supervision: **(F1)** no graph candidate is selected; **(F2)** rows disappear before splitting; and **(F3)** shared support documents break the independent-item assumption. The pilot exposes each: zero graph-origin selections from 484 candidates, 40 of 946 rows dropped, and 151 questions grouped into 22 support-document components. Confidentiality motivates machine-readable lineage, counts, hashes, and validation outcomes. We ask:

RQ1. Which evidence enters compiled examples, and which intended examples survive the actual loading path without losing protected content?

RQ2. What assurance do cross-boundary checks add to simple preprocessing, and which model conclusions remain defensible after accounting for score origin and shared support documents?

We contribute: (1) a canonical object carrying document and graph-record lineage through role-labelled SFT examples; (2) evidence, exposure, and claim contracts with caller-supplied exact encoding and protected-field repair; (3) a case audit that detects an unrealized graph-origin path, repairs loader attrition, and bounds model conclusions; and (4) an installable artifact with controlled comparisons, CPU measurements, a RuBQ adapter, JSON Schema, CLI, negative regression tests, and a paired ruMMLU capability diagnostic[1].

[1]Reproducibility artifact: source code and documentation.

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 98–108
November 10, 2026 ©2026 Association for Computational Linguistics

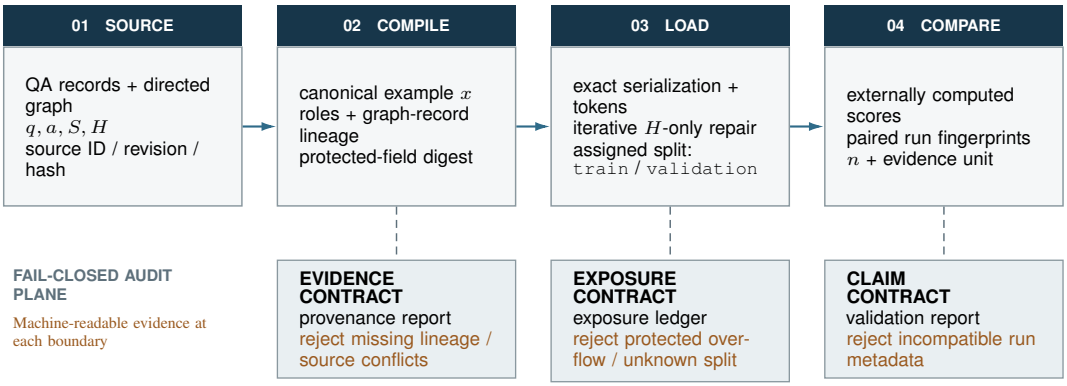


Figure 1: The compiler's data path (top) and audit plane (bottom). Each contract emits a reviewable artifact and stops on incompatible provenance, unrepairable protected content, invalid split assignment, or mismatched comparison metadata. Training, generation, and score computation are external to the library; their outputs enter the comparison stage.

2 Problem and Method

Concrete input and output. The library takes a directed graph JSON and QA JSONL with source revisions, support, and distractors. It emits validated rows, repair records, and a retention ledger; missing lineage or protected overflow fails closed. After external scoring, comparison records bind compatible runs to the same roster, score origin, inference fingerprints, denominator, and evidence unit.

2.1 Canonical compilation object

Let a source reference be $\pi = (i, u, r, h)$: stable source identifier, optional URI, revision, and content hash. A graph record is a node or directed typed relation (v_i, r, v_j) carrying one or more π values. A compiled example is

$$x = (q, a, S, H, \pi_x), \tag{1}$$

where q is the question, a the target, S the verified support chunks, and H non-support context. Each chunk has a stable ID, role, origin, source references, and, for graph-origin chunks, the producing graph-record IDs. An answer example requires non-empty S; refusal examples may have none. Parallel relations remain distinct and no reverse edge is inferred.

The schema separates an evidence object from its rendered text. This matters when several policies share wording or an entity merge obscures document identity. It also permits privacy-preserving publication: source text stays private while counts, aliases, hashes, and validation outcomes can be audited.

2.2 Three executable contracts

Figure 1 follows each object and each score origin.

Evidence contract. Every example, chunk, node, and edge must resolve to source provenance. Directed relations must declare direction; missing endpoints, duplicate IDs, unlabeled roles, and graph-origin chunks without graph-record IDs fail closed. Candidate generation may use graph adjacency, lexical similarity, retrieval, or manual rules, but the selected object's origin remains explicit. Thus a future audit can count graph-origin hard negatives directly rather than infer them from context text. For every source shared by a chunk and graph record, supplied revisions and content hashes must agree; a match on one shared source cannot hide a conflict on another. Support annotations are supplied upstream and preserved by the contract; their semantic validation remains the caller's responsibility.

Exposure contract. Define protected content $I(x) = (q, a, S, \pi_x)$ and mutable context H. A transformation f is admissible only if

$$\mathrm{hash}(I(f(x))) = \mathrm{hash}(I(x)). \tag{2}$$

The implementation also protects the example ID and task type. The caller supplies the serializer σ and tokenizer τ used by its training path. If $|\tau(\sigma(x))| > B$, repair removes the lowest-priority distractor, runs the same tokenizer again, and stops only when the row fits; it fails rather than truncate protected content. The loader adapter then records one ledger entry per intended example: token count, retained/dropped state, reason, and train/validation assignment. Only those

two exact split labels are accepted, and retained IDs must equal the disjoint union of the two exports. These checks establish unique-row retention through splitting. Establishing optimizer consumption additionally requires an independent training trace. The public adapter covers serialization, exact token IDs, budget retention, and split assignment; downstream collators and packing stages require separate checks. The case study replays the recovered closed loader and validates its final collated batches. Tail removal provides a simple, replaceable reference policy based on trusted support labels and caller-provided distractor priorities.

Claim contract. Every reported comparison binds a model artifact, fixed evaluation roster, prompt/decoding fingerprint, endpoint, denominator, missing-row policy, uncertainty method, and evidence unit. Questions are connected when their gold support resolves to a common source document; resampling and sign changes then operate on connected components. Token F1 is computed from the reference answer, the unsupported-claim proxy from sentence–retrieved-chunk NLI, and the corroborative correctness label from a blind external model judge. These are separate observables: context support, reference overlap, and judged answer correctness. Metrics are computed externally; the public validator checks claim-record completeness and run compatibility.

3 Enterprise Case Study

3.1 Data and controlled conditions

The confidential training-source inventory contains 25 technical architecture and operations documents. Evaluation uses 159 held-out QA records; the principal matched roster resolves to a separate 54-document gold-support inventory. Training and evaluation support inventories do not overlap. One training document occurs in the broader retrieved evaluation context only as non-support evidence. Retrieved contexts are long and code-heavy, usually with 15 chunks from a median of three documents.

The case is a retrospective audit of an archived offline adaptation pilot. It traces selected evidence, reconstructs the loading path, and produces a loader-safe corpus for subsequent training. Prospective integration follows the same interfaces: project records are normalized to the public schema and connected to a local loader adapter.

All conditions use a proprietary instruction-tuned checkpoint archived as GigaChat 2 Pro. Its public family report (Mamedov et al., 2025) does not identify our internal revision; architecture and parameter count are not inferred. LoRA conditions are typed context, no-hard-negative control, distractor-bearing adaptation, and loader-safe repair, compared with the unadapted generator. Appendix D records the training profile and missing trainer-state facts.

The typed source contains 1,079 intended dialogues; its distractor-bearing successor has 946 (833 answer-bearing and 113 refusal controls). Across matched keys, targets and support IDs are unchanged while 526 context signatures differ. Candidate rules admit at most two confusable, version-neighbor, or graph-derived hard negatives in six chunks. These selected additions are only one subset of mutable H, which also contains pre-existing retrieved chunks explicitly labelled non-support.

At the selection boundary, a server audit records a 484-chunk graph pool. Final metadata labels all 1,119 selected chunks in 573 rows as `existing_context` and none as graph-origin; independent aggregates agree. Approval was required, but candidate-level eligibility, approval, and rank traces are unavailable. The available selector identifies the gates whose outcomes would be needed to diagnose non-selection (Appendix H). With zero graph-origin chunks in compiled H, the model comparisons evaluate the realized distractor-bearing contexts.

The archived 83-node, 56-edge graph also lacks document provenance and declared direction despite direction-sensitive relations; 25/428 raw entity rows lack provenance. These records fail the public evidence contract's lineage and direction requirements.

3.2 Evaluation and inference

The main model comparison uses the exact 151-question intersection; questions, gold answers, contexts, prompts, and decoding fields are checked for equality. Available full-context outputs differ by condition (152–158 of 159), so every contrast reports its own denominator.

We retrospectively designate mean per-answer NLI-based unsupported-claim rate and macro token F1 as the two primary endpoints; exact definitions appear in Appendix A. The former estimates support using an automatic NLI model. Pre-

Corpus	Intended rows	Retained	Train / val.
Typed contexts	1,079	788	709 / 79
Distractor-bearing	946	906	815 / 91
Loader-safe repair	946	**946**	851 / 95

Table 1: Loader-retention funnel under exact recovered-loader replay. Counts are unique rows before epoch repetition.

Pipeline	Retained	Bad ID	Bad revision
Per-record checks	–	Accept	Accept
Exact length filter + replay	1/2	Accept	Accept
Role-aware repair + replay	2/2	Accept	Accept
Full compilation contracts	2/2	Reject	Reject

Table 2: Controlled public checks. Retention uses valid inputs; each faulty lineage input is tested separately. "–" means no loader check.

cision, recall, answer/gold length, and context ablations are diagnostics. A separate proprietary judge, blind to condition, sees the question, reference, candidate answer, and full context and accepts both correct and partial answers; its McNemar test is pair-level and corroborative.

The authoritative field-specific support resolver maps all 151 questions to 54 documents and 22 connected components; the largest contains 65 questions. We bootstrap and sign-flip whole components, then Holm-correct the two primary endpoints within each comparison. Appendix B specifies the estimand, resampling procedures, and support-resolution audit.

4 Results

4.1 The compiler exposes and repairs supervision loss

Table 1 is the process result. Exact replay of the recovered 1,024-token loader drops 291 typed-context rows and 40 distractor-bearing rows before splitting. For the latter, the compiler removes 104 annotated non-support chunks from the 40 over-budget rows and re-tokenizes after each edit. Only two removals are newly selected hard negatives; the rest are pre-existing mutable context. Questions, targets, support, roster, and order remain unchanged, and all 946 rows are retained: 851 train-assigned and 95 validation-assigned. The verified endpoint is unique-row retention through the recovered loading path.

4.2 Controlled checks on a public fixture

We compare four pipelines on the same public two-row Atlas fixture, budget, serializer, and tokenizer (Table 2). Per-record validation alone makes no retention claim; exact length filtering drops one row. Role-aware repair plus replay retains both, just as the full compiler does. The incremental assurance is cross-record lineage: only the full pipeline rejects both an unresolved graph-record reference and a conflicting source revision. Replay-based alternatives also reject invalid split labels. The injected faults isolate the checks provided by each pipeline.

The repaired fixture retains one graph-origin distractor row and removes one other distractor without changing protected content, exercising both lineage validation and repair on graph-bearing inputs.

4.3 Bounded software costs and public checks

Five warmed CPU runs on 100, 1,000, and 10,000 replicated Atlas rows take median .0098, .0904, and .9791 seconds including four exports. With a fixed four-node graph and whitespace tokens, these measurements characterize the controlled fixture workload (Appendix F). Official RuBQ import checks schema portability (Rybin et al., 2021); paired ruMMLU-1,000 provides a separate general-capability diagnostic (Fenogenova et al., 2024) (Appendix E).

4.4 Downstream model comparisons

Against the unadapted system, unsupported-claim rate changes by $-.026$ (component CI $[-.106, .024]$, $p = .543$); against no-hard-negative training, token F1 changes by $+.018$ (CI $[.004, .023]$, $p = .094$, Holm $p = .187$). None of eight component tests is Holm-significant. The blind judge accepts 140/151 unadapted versus 138/151 adapted answers ($p = .791$). Shorter answers, lower recall, zero-claim scoring, and differing claim-extraction coverage further limit interpretation. Complete effects, coverage, and descriptive context ablations are in Appendices A–C. Taken together, the comparisons remain inconclusive about a broad correctness improvement.

5 Related Work

Chen et al.'s KG-SFT extracts external reasoning subgraphs to generate and conflict-check explanatory supervision (Chen et al., 2025a); Graph-

Gen builds a fine-grained KG from source text and generates atomic, aggregated, and multi-hop QA (Chen et al., 2025b). These systems construct or enrich supervision; ours audits the subsequent evidence-to-loader-to-claim handoff. RAFT and PrismRAG provide the closest motivation for relevant-plus-distractor reader training (Zhang et al., 2024; Kachuee et al., 2025).

GraphRAG retrieves or organizes graph evidence at query time (Edge et al., 2024; Hu et al., 2025; Sharma et al., 2025); here graphs are offline construction inputs. DataDreamer contributes workflow fingerprints and traced cards, RAG Foundry integrates the adaptation workflow, and Bonito generates domain instructions (Patel et al., 2024; Fleischer et al., 2024; Nayak et al., 2024). Our interface jointly carries source-document and graph-record lineage through role-labelled SFT compilation, protects content under caller-supplied exact encoding and loader replay, and binds comparisons to score provenance and dependence components induced by shared gold-support documents. The integration combines established validation and pruning techniques into one auditable data path. Factuality work separates overlap, support, and correctness (Krishna et al., 2025; Fatahi Bayat et al., 2025); LoRA enables adaptation (Hu et al., 2022) but does not remove data-path confounds.

6 Adoption and Conclusion

Teams can start with lineage checks, add exact-token replay, then bind external scores. Without actual loader access, retention remains unknown. Appendix G details integration requirements and tradeoffs; deployment effort and production throughput are unmeasured.

In the enterprise case, lineage accounting identified an unrealized graph-origin channel, exact loader replay exposed silent row loss, and protected-field repair restored the intended corpus. Controlled public checks isolate the additional assurance from cross-record lineage validation, while staged adoption makes each interface available independently. Downstream model comparisons remained inconclusive.

For enterprise teams, the transferable order is simple: trace the selected evidence, verify the exact loading path, and only then attribute model effects. KG-SFT Contracts implements this order as executable evidence, exposure, and comparison contracts with inspectable outputs at each boundary.

Limitations of the Work

The study uses one proprietary checkpoint, one Russian enterprise QA set, and one checkpoint per condition; text, weights, exact revision, and training-seed variance are unavailable. Generation failures vary by condition. Inference has only 22 support-document components, one containing 43% of questions, and may miss other dependence. NLI and the model judge are not human evaluation; endpoints, component analysis, and repair were retrospective. The archive has no usable graph lineage and zero graph-origin hard distractors, so model contrasts cannot estimate a graph contribution. Loader retention is not optimizer exposure, and ruMMLU non-significance is not equivalence or non-inferiority. Public checks establish software behavior, not enterprise transfer or semantic support correctness; the redacted evidence cannot expose confidential text or emulate arbitrary trainer packing. The public cost study replicates two short examples against a fixed tiny graph and an inexpensive tokenizer; it does not measure production tokenization, large-graph behavior, integration effort, or incremental model benefit over simpler preprocessing. The missing selector trace prevents historical root-cause attribution.

Ethical Considerations

Only aggregates, hashes, synthetic text, and public RuBQ fields are released; no enterprise passage, output, credential, path, or weight is included. Automated evaluators may encode bias; source lineage supports deletion and license review.

Generative AI Declaration

Generative AI tools assisted manuscript drafting, language editing, and supporting code development. Model-based answer generation and evaluation are described in the experimental protocol. The authors are responsible for verifying the text, code, and results and for the final manuscript.

Acknowledgments

This article is an output of a research project (HSE-BR-2026-026) implemented as part of the Basic Research Program at HSE University. The article is also supported in part through computational resources of HPC facilities at HSE University. We gratefully acknowledge Sber for supporting this research and providing the computational resources and technical infrastructure used in the enterprise case study. We also thank ITMO University for its support.

References

Hanzhu Chen, Xu Shen, Jie Wang, Zehao Wang, Qitan Lv, Junjie He, Rong Wu, Feng Wu, and Jieping Ye. 2025a. Knowledge graph finetuning enhances knowledge manipulation in large language models. In *International Conference on Learning Representations*.

Zihong Chen, Wanli Jiang, Jinzhe Li, Zhonghang Yuan, Huanjun Kong, Wanli Ouyang, and Nanqing Dong. 2025b. GraphGen: Enhancing supervised fine-tuning for LLMs with knowledge-driven synthetic data generation. *arXiv preprint arXiv:2505.20416*.

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*.

Farima Fatahi Bayat, Lechen Zhang, Sheza Munir, and Lu Wang. 2025. FactBench: A dynamic benchmark for in-the-wild language model factuality evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 33090–33110. Association for Computational Linguistics.

Alena Fenogenova, Artem Chervyakov, Nikita Martynov, Anastasia Kozlova, Maria Tikhonova, Albina Akhmetgareeva, Anton Emelyanov, Denis Shevelev, Pavel Lebedev, Leonid Sinev, Ulyana Isaeva, Katerina Kolomeytseva, Daniil Moskovskiy, Elizaveta Goncharova, Nikita Savushkin, Polina Mikhailova, Anastasia Minaeva, Denis Dimitrov, Alexander Panchenko, and Sergey Markov. 2024. MERA: A comprehensive LLM evaluation in Russian. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9920–9948, Bangkok, Thailand. Association for Computational Linguistics.

Daniel Fleischer, Moshe Berchansky, Moshe Wasserblat, and Peter Izsak. 2024. RAG Foundry: A framework for enhancing LLMs for retrieval augmented generation. *arXiv preprint arXiv:2408.02545*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2025. GRAG: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4145–4157. Association for Computational Linguistics.

Mohammad Kachuee, Teja Gollapudi, Minseok Kim, Yin Huang, Kai Sun, Xiao Yang, Jiaqi Wang, Nirav Shah, Yue Liu, Aaron Colak, Anuj Kumar, Wen-tau Yih, and Xin Luna Dong. 2025. PrismRAG: Boosting RAG factuality with distractor resilience and strategized reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 775–798. Association for Computational Linguistics.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. 2025. Fact, fetch, and reason: A unified evaluation of 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*, pages 4745–4759. Association for Computational Linguistics.

Valentin Mamedov, Evgenii Kosarev, Gregory Leleytner, Ilya Shchuckin, Valeriy Berezovskiy, Daniil Smirnov, Dmitry Kozlov, Sergei Averkiev, Lukyanenko Ivan, Aleksandr Proshunin, Ainur Israfilova, Ivan Baskov, Artem Chervyakov, Emil Shakirov, Mikhail Kolesov, Daria Khomich, Daria Latortseva, Sergei Porkhun, Yury Fedorov, and 14 others. 2025. GigaChat family: Efficient russian language modeling through mixture of experts architecture. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 93–106. Association for Computational Linguistics.

Nihal Nayak, Yiyang Nan, Avi Trost, and Stephen Bach. 2024. Learning to generate instruction tuning datasets for zero-shot task adaptation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12585–12611. Association for Computational Linguistics.

Ajay Patel, Colin Raffel, and Chris Callison-Burch. 2024. DataDreamer: A tool for synthetic data generation and reproducible LLM workflows. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3781–3799. Association for Computational Linguistics.

Ivan Rybin, Vladislav Korablinov, Pavel Efimov, and Pavel Braslavski. 2021. RuBQ 2.0: An innovated russian question answering dataset. In *The Semantic*

Web: 18th International Conference, ESWC 2021, pages 532–547. Springer International Publishing.

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. Association for Computational Linguistics.

Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. RAFT: Adapting language model to domain specific RAG. In *Proceedings of the First Conference on Language Modeling*.

A Endpoint Definitions

Unsupported-claim proxy. Candidate answers are split after sentence-final `.`, `!`, or `?` followed by whitespace; stripped segments longer than 15 characters are treated as claims. Each claim is checked against every non-empty retrieved chunk with `MoritzLaurer/mDeBERTa-v3-base-mnli-xnli`. For chunks longer than 1,500 characters, a lexical-overlap heuristic selects the top three context sentences; the premise is then truncated so premise plus hypothesis is approximately 1,800 characters. A claim is supported if any chunk has `ENTAILMENT` as its argmax class; no confidence threshold is applied. The per-answer value is the fraction of claims without such support and is zero when no claim is extracted. If NLI is unavailable, the whole metric is missing rather than imputed. All primary pairs have zero missing endpoint values. The archived configuration and weight SHA-256 values are, respectively, `4e4430c95100d613df80fa01276f231931e1b535557cf62fbb3cc50323e50cca` and `65af59b1ff4450b09ecbf13ca35c840dbf038b26ff8e10e5ea89ca724828ed1e`.

Token F1. Reference and candidate are lowercased and tokenized with the Unicode-aware regular expression `\w+`. Multiset token overlap defines precision and recall, whose harmonic mean is F1; two empty strings score one and exactly one empty side scores zero. Per-answer values are rounded to four decimal places before the macro average.

Blind correctness judge. GigaChat 2 Max receives the question, reference, candidate answer, and complete retrieved context without truncation. Correct and partial labels are accepted; the paired binary endpoint rejects all other valid labels. Temperature and top-p are both .01, the response-token cap is unset, and at most three attempts are allowed. The stripped prompt SHA-256 is `662763567bf9253e2fc523ef4f15f5ece80f5c515956a26ae585c872e4bc3ea3`. All 151 matched pairs completed without judge/backend failure.

Claim-extraction coverage. Because a zero-claim answer receives unsupported-claim rate zero, Table 3 reports this edge case explicitly. The metric is defined over extracted claims rather than all answer text; condition-level answer counts differ because generation artifacts are incomplete. Differences in claim-extraction coverage may therefore contribute to unsupported-claim-rate contrasts; these unmatched records do not support causal attribution or exact decomposition of that contribution.

Condition	Answers	Zero claims	Claims
Unadapted	152	7 (4.6%)	598
Typed context	153	16 (10.5%)	392
No hard negatives	158	18 (11.4%)	387
Distractor-bearing	155	13 (8.4%)	492
Loader-safe repair	155	13 (8.4%)	448

Table 3: NLI claim-extraction coverage in available full-context outputs.

B Dependence-Aware Inference

For endpoint y, the estimand is the question-weighted paired mean difference $\hat{\Delta} = n^{-1} \sum_i (y_{i,2} - y_{i,1})$. Questions whose gold support documents are connected directly or transitively form one component. The four primary comparisons each contain 22 components: the 151-pair rosters have a largest component of 65, and the 155-pair rosters a largest component of 68.

For each of 20,000 bootstrap draws, we sample 22 whole components with replacement and divide their summed paired differences by their summed sampled question counts; the reported interval is the percentile 2.5–97.5 interval. The two-sided randomization test instead multiplies each component sum by one independent Rademacher sign and compares the absolute question-weighted mean against the observed statistic, retaining the fixed total denominator. With 22 components we use 50,000 Monte Carlo draws and the plus-one

correction. Both procedures concern the same paired mean contrast, but the percentile bootstrap uses a resampling distribution whereas the randomization test uses a null distribution that assumes component-level aggregate differences are sign-exchangeable (symmetric about zero). The interval is not an inversion of that test and the two may disagree in finite samples. Within each comparison, Holm correction covers exactly the two component sign-flip tests for unsupported-claim rate and token F1. Pair-level Wilcoxon tests are sensitivity analyses, not primary inference. Endpoints and procedures were selected retrospectively; no result is labelled confirmatory.

C Complete Primary Effects

Table 4 reports both retrospectively designated primary endpoints for every controlled comparison. Deltas are second minus first; negative unsupported-claim rate and positive F1 are favorable. Equal counts do not imply equal rosters.

Context-ablation diagnostics. With full context, the distractor-bearing generator reaches .312 token F1 and .210 unsupported-claim rate, versus .191/.956 without context, .160/.643 with distractors only, and .287/.364 with top-1 context. Unequal mode denominators (154–159) make these comparisons descriptive. The no-context NLI value partly follows from asking whether claims are supported by context available in that ablation; it is not an independent causal factuality estimate. The F1 pattern is consistent with use of answer-bearing context.

D Experimental Conditions and Traceability

The five paper-facing conditions are the unadapted generator, typed-context adaptation, no-hard-negative control, distractor-bearing adaptation, and loader-safe repair. Available full-context records are 152, 153, 158, 155, and 155, respectively. Historical small-checkpoint experiments remain registry-only and are never pooled with these estimates. Legacy metrics-only exports preserve question IDs and endpoint values but predate run/code fingerprints; paired reports therefore hash their exact inputs, rosters, support-component assignment, and analysis implementation.

The archived Pro profile specifies PEFT LoRA with rank 16, scaling 32, dropout .05, and q/k/v/o/gate/up/down projection targets. It records Adam, learning rate 2×10^{-5}, weight decay .001, 50 warm-up steps, three epochs, a 1,024-token training cap, fp16, seed 228, per-device batch one, and eight-step gradient accumulation. No trainer state survives for the exact scheduler, global step, process count, effective global batch, or selected checkpoint, so we do not reconstruct them.

The open diagnostic pins maintained MERA/ruMMLU revision `73cf223e40f1ef51ba74a048b584874a35c3d88b`, seed 1234, and a subject-round-robin 1,000-item manifest spanning all 57 subjects. Both systems reuse the same one-shot, dataset-rendered prompts and exact GigaChat SFT encoding; answers are scored by candidate-letter log probability. The paired report rejects partial rosters or any item, prompt, or protocol mismatch.

E Open Diagnostics

ruMMLU. On the frozen 1,000-item maintained-MERA roster, the unadapted and loader-safe systems score 78.0% and 77.7% (Wilson intervals [75.3, 80.5]% and [75.0, 80.2]%). The paired delta is -0.3 percentage points (bootstrap interval $[-1.2, +0.6]$ percentage points; McNemar $p = .678$). This detects no shift, but establishes neither equivalence nor non-inferiority and does not validate the private RAG endpoints.

RuBQ portability. The adapter maps public question, SPARQL, answer-bearing, and related-paragraph fields into the canonical schema. On the official RuBQ 2.0 test release, a frozen seed-228 preflight converted 100 of 1,694 eligible rows with 160 support and 2,108 distractor chunks; all passed provenance validation, and a rerun reproduced the output hashes. The converted rows and deterministic hashes provide a reproducible schema-portability check.

F Controlled Software Comparisons and Costs

The controlled software checks use the published synthetic Atlas fixture. It contains two QA rows, each with one support chunk and one graph-linked distractor. A 36-token budget under `FixtureLoaderAdapter` removes one distractor and retains both rows, including one graph-origin distractor row. All alternatives use the same inputs and encoding. “Per-record checks”

Contrast	Endpoint	n	Δ	Component 95% CI	p_{comp}	Holm p
Unadapted→distractor	Unsupported claims	151	-.026	[-.106, .024]	.543	1.000
Unadapted→distractor	Token F1	151	-.001	[-.028, .020]	.916	1.000
Typed→distractor	Unsupported claims	151	+.002	[-.012, .029]	.880	1.000
Typed→distractor	Token F1	151	+.002	[-.028, .012]	.861	1.000
No-hard-neg.→distractor	Unsupported claims	155	+.012	[-.031, .025]	.406	.406
No-hard-neg.→distractor	Token F1	155	+.018	[.004, .023]	.094	.187
Original→loader-safe	Unsupported claims	155	+.001	[-.027, .030]	.969	1.000
Original→loader-safe	Token F1	155	+.001	[-.009, .008]	.773	1.000

Table 4: Complete source-component primary-effect estimates. Pair-level tests are sensitivity analyses.

runs `ContractExample.from_dict` independently on each row. "Length filter" replays the exact fixture budget without editing; "repair + replay" uses the identical tail-removal policy as the full compiler, but omits graph-lineage validation and compilation exports. This shared policy isolates the contribution of cross-record checks and audit exports.

Two independently injected faults retain valid per-record structure but (a) reference an absent graph relation or (b) change the chunk source revision without changing its graph record. Only full compilation rejects these two cross-record faults. A third fault makes the adapter return an unknown split; all replay-based alternatives reject it, whereas per-record checks do not inspect it. In a fourth, support alone exceeds the budget: the length filter drops the row, and both repair pipelines fail explicitly instead of truncating support. No fault-detection percentage or production prevalence is estimated from these hand-constructed cases.

For timing, we replicate the pair with unique example IDs to 100, 1,000, and 10,000 rows, leaving source text and the four-node, three-relation graph fixed. All rows are retained; half lose one distractor and half retain their graph-origin distractor. The measurement environment is Darwin arm64, Python 3.12.9, and NetworkX 3.6.1, matching the package lockfile. Each workload is run five times after one 100-row warm-up; imports and example construction are untimed. Stage probes and the full compilation probe run separately, so their times must not be summed. Export time includes JSON/JSONL encoding and filesystem writes, not durable-storage synchronization. Raw timings, environment, and source hashes are in `public_validation/revision_software_checks.json`.

In the uncached reference implementation, a row with r_i distractor removals uses $r_i + 3$ token-count calls during repair (initial, loop checks, final), plus one during replay. The total is $4n + \sum_i r_i$, giving the exact call counts in Table 5. A filter-only replay uses n calls but accepts row loss. Real tokenizers, long contexts, packing, and larger graphs can change runtime substantially. Production overhead, graph-size scaling, integration labor, and model benefits require separate measurements.

To reproduce from the supplemental source package:

```
uv run --frozen python \
scripts/run_revision_software_checks.py
uv run --frozen python -m unittest \
tests.test_revision_software_checks
```

G Incremental Adoption

The three interfaces can be adopted separately; passing an earlier stage does not certify a later one. The library accepts an existing graph and does not perform graph extraction, retrieval, training, or scoring.

Stage 1: evidence-only integration. Normalize stable example/chunk/graph-record IDs, source revisions or hashes, support/distractor roles, and directed relations. Run per-record and cross-record checks before any model work, and count selected origins in the compiled output. This needs no tokenizer, but does require existing metadata or upstream changes to capture it. It establishes recorded lineage consistency, not semantic support correctness, selection quality, or loader survival.

Stage 2: exact encoding and retention. Implement `serialize`, `token_ids`, and `partition` with the actual training template, tokenizer revision, budget, and split policy. First compare replay counts and IDs with the real loader on a small sample; only then enable distractor-only repair. If collators or packing filter rows, check those stages separately. Without that access,

Rows	Record ms	Lineage ms	Repair ms	Replay ms	Compile s [min, max]	Token calls
100	1.21	.096	3.72	.254	.0098 [.0098, .0103]	450
1,000	11.80	.783	36.96	2.575	.0904 [.0893, .0910]	4,500
10,000	151.43	7.716	369.21	26.653	.9791 [.9640, .9914]	45,000

Table 5: Controlled CPU software timings: medians over five runs after a 100-row warm-up. Separate stage probes are not additive. Compilation includes graph checks, repair, replay, hashing, and four file exports, but excludes record parsing, imports, and workload construction.

publish retention as unknown rather than substitute the fixture adapter. A filter-only pipeline may be adequate if dropping over-budget rows is intentional; repair is useful when the intended corpus must be conserved and trusted role labels exist.

Stage 3: comparison integration. Attach externally generated scores to immutable run and roster fingerprints, endpoint definitions, missingness, and support-document IDs. Bind uncertainty to the declared dependence unit; the library does not compute the scores or prove statistical assumptions. Source linking and production adapters are the main project-specific work, for which no staff-time estimate is available.

The private audit illustrates two integration outcomes: detecting 40 unintended row losses and identifying which evidence channels reached the corpus. Teams needing only row-format checks may prefer a simpler validator. The staged workflow makes provenance checks available before exact trainer integration; its requirements and trade-offs are also documented in the package's adoption guide.

H Diagnosing an Unrealized Candidate Channel

The archive establishes an inventory of 484 graph candidates, approval-required selection, and zero graph-origin selections among 1,119 selected chunks. It does not establish how many graph candidates passed each intermediate gate. Inspecting the available selector reveals support/duplicate exclusions, relevance rules, graph-source-document eligibility, fingerprint-matched approval, and ranked top-k selection. Existing-context candidates receive a score bonus, but without eligible-candidate scores this does not explain the observed outcome. A global pool count is not a count of eligible candidates per question, and missing approval traces do not prove that every candidate was unapproved. This code inspection is not a version-pinned historical replay.

For a future prospective run, record one event per example–candidate pair: stable candidate/graph/source IDs and content hash; origin; proposed role; relevance score and rule; provenance and approval outcomes; deduplication outcome; rank, cutoff, and selected/rejected status; and a first-rejection reason. Aggregate this trace by origin through proposed, eligible, approved, ranked, selected, and loader-retained stages. Use explicit reasons such as missing source, approval absent, content-hash mismatch, relevance below threshold, duplicate, or rank beyond budget. Persist the selector/configuration hash and reconcile final selected counts with compiled-row lineage and the retention ledger. This proposed trace extends the public compiler's boundary checks with upstream selector instrumentation. Recovering the historical cause of zero graph-origin selections would require such candidate-level records.

I Corpus, Support, and Leakage Audits

The typed and distractor-bearing corpora contain 1,282 and 1,125 unique context chunk IDs across the same 25-document inventory. Their 946 matched rows have zero target or support-ID changes and 526 context-signature changes. Normalized training/evaluation question, target, and evidence-text hashes are disjoint; support chunk and support-document ID intersections are zero. One training document appears among all retrieved evaluation documents but not among evaluation support documents, so the narrow claim is zero verified support leakage rather than zero document overlap.

The specialized support audit resolves all 151 principal rows from the explicit gold-support field. It recovers 54 source documents and 22 components. A generic five-document count is rejected because it omits the field-specific resolution rule. Forty-three rows have multiple support documents; their semantic interpretation requires blinded review before future confirmatory analysis.

J Reproducibility Artifact

Artifact availability. The public implementation is available through the source code and documentation, with the baseline fixed at commit `089398a10d31`. The repository and supplementary source package expose the canonical dataclasses, JSON Schema, directed `MultiDiGraph` builder, invariant-aware transforms, exact-token interface, replay ledger, comparison-claim validator, source-component helper, RuBQ adapter, synthetic Atlas fixture, and deterministic tests, including invalid-split, conflicting-lineage, and null-document-ID regressions. The source supplement additionally includes controlled alternative checks, their tests, timing records, and incremental adoption documentation. The minimal public smoke test is:

```
uv run python \
  scripts/run_public_smoke.py
```

It compiles the fixture twice and checks output hashes, protected fields, directed relation counts, repair records, split assignment, and a compatible claim record. The fixture tokenizer is explicitly a smoke-test adapter. Reproducing the enterprise replay requires the archived production serializer/tokenizer and private data, which are not distributable. A separate supplementary evidence bundle contains redacted selection and retention audits, aliased primary endpoint rows, a recalculation script, and the strict aggregate ruMMLU report; it contains no enterprise text, raw identifiers, model output, server path, or weight.

Tracing False Speaker Links Through Retrieval and Answering

Dev D. Goyal
Independent Researcher
dvgyl@outlook.com
ORCID: 0009-0009-9132-744X

Abstract

Memory systems for large language model (LLM) agents can organize past conversations by person, for example in a speaker index or a knowledge graph. To answer "Who promised the budget?", such a system searches the relevant people's messages and passes a few to the LLM. Building the index requires entity resolution, which can wrongly link two people. A false link can change what the LLM reads and make it name the wrong person, but final-answer scores may not show this. We add false links at known rates to a retrieval pipeline, keep every message and name unchanged, and follow 280 questions from two synthetic benchmarks from search to answer. With one false link per 40 people, almost one in four workplace searches reaches a wrong person. At the highest rate, gpt-oss-120b reads a different context in 81.0% of workplace runs, but its accuracy falls by 1.75 percentage points, because improved and worsened answers offset each other. For gpt-oss on the social benchmark, one question that loses its only labeled supporting message accounts for 79% of history-weighted loss. Accuracy alone can therefore hide identity errors. We provide a tool and a memory error card that measure each step.

1 Introduction

Consider a board-game group chat from the SocialMemBench benchmark (Owolabi, 2026). Two members say that players should read the rules before game night. Tomasz says it in general terms. Marcus is specific: "whoever owns it should send the rulebook link by Wednesday." A user asks a memory assistant which member gave the specific procedure. With a correct speaker index, the gpt-oss-120b model answers Marcus in all four runs. We then add five false links to the index. One of them wrongly joins Marcus with Derek, so the search also collects Derek's messages, such as "agree. read the rules people." The model can read only a limited amount of text. The changed selection omits Marcus's labeled statement, and the model answers Tomasz instead. No name and no message changed; only the selection did (Figure 1).

Assistants with long-term memory store past conversations and, for each question, retrieve a few relevant messages for a large language model (LLM) to read (Packer et al., 2023; Chhikara et al., 2025). Many memory systems organize what they store around people and other entities, for example in an entity graph (Rasmussen et al., 2025; Gutiérrez et al., 2025). We study a simple form of this idea: a *speaker index* that groups messages by the person who wrote them, so that the search can focus on the people a question is about. To build such an index, a system must decide which names refer to the same person. This task, called *entity resolution*, makes mistakes: it can join two different people, and one wrong join can lead to others (Baas et al., 2021; Pavani et al., 2026). SocialMemBench reports such identity problems in memory for groups. We call a wrong join a *false link*.

These errors matter for two reasons. First, they can reach users. A false link can make the assistant credit the wrong person, as in the Marcus example, although every stored message is correct, and it adds messages from people the question is not about to the model's input. Second, a score based only on final answers may not show them. A changed search can return the same text, a changed text can produce the same answer, and when answers change, gains can cancel losses. Studies of model updates and compression call such changes churn or flips (Milani Fard et al., 2016; Dutta et al., 2024). A memory system can therefore look robust to identity errors while some of its answers fail.

We measure where the effect of a false link goes.

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 109–130
November 10, 2026 ©2026 Association for Computational Linguistics

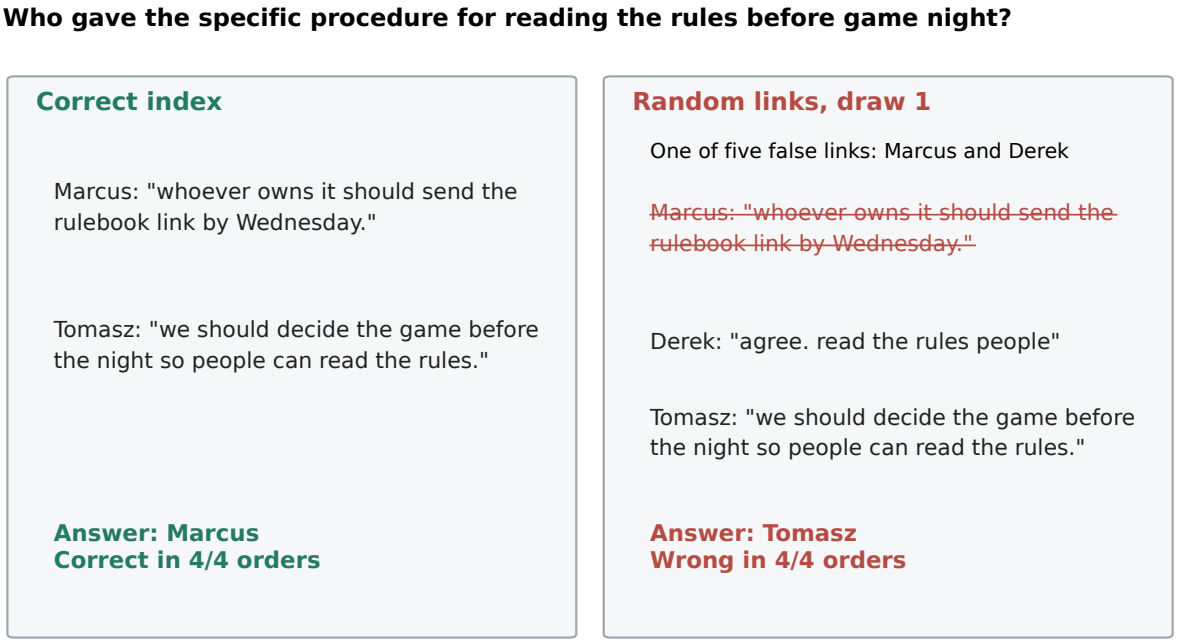


Figure 1: With false links, the only labeled supporting message is absent and the model names the wrong person, although every name stays correct. The panels show selected messages for the Marcus question. The first random set of five false links includes one between Marcus and Derek. Two Derek messages and one Soo-Yeon message enter; Marcus's labeled statement and one Priya message leave. Over three random sets and four option orders, gpt-oss selects Tomasz in 10 of 12 runs. We selected this case after decomposing the losses. Appendix H gives the complete trace and the contrasting Qwen result.

We add false links at known rates to the speaker index of a retrieval pipeline and keep every message and name unchanged. We answer each question with the correct index and with the false links, and we record five steps for both: the people that the search reaches, the shortlisted messages, the text that the model reads, its answer, and whether the answer is correct. We call this paired record a *stage audit*. We test 280 questions from two synthetic benchmarks, one about workplace projects and one about social groups, with three LLMs. Unlike evaluations that compare only final scores, this design keeps every message and name fixed and shows the step at which each change stops.

We find three patterns. First, false links reach many questions even when they are rare, because each search starts from several people: with one false link per 40 people, 23.6% of workplace searches reach a wrong person (Section 4.1). Second, accuracy hides most of the change: at the highest rate, gpt-oss reads a different text in 81.0% of workplace runs and changes 19.25% of its answers, but its accuracy falls by only 1.75 percentage points (Section 4.2). Third, gpt-oss losses concentrate in a few social questions: 5 of 80 questions carry all lost answers, and one carries 79% of history-weighted loss (Section 4.3).

Our contributions are:

1. A controlled test of speaker-index errors that changes only which messages the search may use, while every message and name stays correct.
2. Evidence that final accuracy hides both widespread answer changes and concentrated failures, with a simple formula for the share of questions that an error rate reaches.
3. An offline tool and a one-page memory error card that let other teams measure each step of their own memory systems, including loss and gain rates that account for different starting accuracy.[1]

2 System and intervention

2.1 From a question to a context

Retrieval finds past messages that may help answer a question. Our pipeline combines meaning-based search with BGE-small-en-v1.5 and word-based search with BM25 (Xiao et al., 2024;

[1]Code and memory error card: https://github.com/dvgyl/false-speaker-links

Robertson and Zaragoza, 2009). Reciprocal rank fusion combines their rankings, and the top eight messages provide the starting speakers (Cormack et al., 2009).

Under the *correct index*, each connected group contains one true person. The speaker filter admits messages from starting speakers and anyone linked to them, including through a chain. The first 64 eligible messages form the *shortlist*, where a *reranker* scores each question–message pair for relevance (Sentence Transformers, n.d.).

The pipeline scans the shortlist in score order and keeps each whole message that fits the remaining *context budget* of 2,048 tokens, the text units counted by the tokenizer. Selected messages form the *context*, which goes in time order to the *reader*, the model that answers the question. Appendix A gives scoring and serialization details.

False links preserve source text, names, relevance scores, and message costs. *Unfiltered search* removes the speaker filter and takes the 64 highest-ranked messages from the whole history. False links relax the filter, so they can also restore useful messages that the correct index excludes. The intervention can therefore produce gains as well as losses.

2.2 False links and query exposure

For a history with $n \geq 2$ speakers, random links connect each distinct pair independently with probability $d/(n-1)$. The *link rate* d is the expected number of incident false links per person. Thus $d = .05$ means one undirected false edge per 40 people on average. At this small rate, about one person in 20 has an incident false link. We test $d \in \{.05, .2, .8\}$ with ten fixed *draws*, each a sampled link set. Within a draw, higher rates add links to the same set, and we retain draws with no links. *Look-alike links* favor similar names while matching each random draw's edge count (Appendix B).

A search starts from several people, so a sparse graph can expose many questions to a false link. With K starting speakers, the chance of reaching at least one new person is

$$P(\text{new person}) = 1 - \left(1 - \frac{d}{n-1}\right)^{K(n-K)}. \quad (1)$$

For small pair probabilities and $K \ll n$, this is approximately $1 - e^{-Kd}$. For example, $K = 5$ and $d = .05$ give about 22% exposure. Exposure is the first opportunity for a retrieval change, not an answer-error probability.

3 Experimental design

3.1 Workloads, readers, and controls

EverMemBench-Dynamic supplies five workplace conversation histories (Hu et al., 2026). Fixed hash selection chooses 40 four-option questions per history, preserving six eligible question families. SocialMemBench supplies all 80 attribution questions from 41 histories (Owolabi, 2026). A *history* is one released conversation log. We search each complete history because the questions have no temporal cutoff. Answer keys enter evaluation but are excluded from retrieval and reader inputs.

The readers are Qwen2.5-7B-Instruct, gpt-oss-120b, and GPT-4.1 mini. We shorten the first two names to Qwen and gpt-oss. Qwen is locally served with Q6_K quantization (Qwen et al., 2025). Qwen and gpt-oss use all 280 questions. GPT-4.1 mini uses all social questions and 16 workplace questions per history. Four cyclic *option orders* place each candidate answer once in each displayed position. Scoring preserves answer identity when its letter changes.

At $d = .8$, gpt-oss uses three draws of each link process, while GPT-4.1 mini uses one. Qwen's four-order comparison uses one random draw; its earlier original-order experiment used three draws at all tested rates. Table 1 reports these differences. All readers use temperature zero, with prompts, decoding settings, and failure rules specified in Appendix C.

No memory supplies the question and options without conversation text. *Full evidence* supplies every message marked as support by the benchmark, outside the normal context budget. These *evidence labels* support a measurable coverage check, although they can omit useful text or include background. Full-evidence requests fit for 196 workplace questions, or 79 for GPT-4.1 mini, and all social questions. Matched comparisons use only these feasible cohorts; the label limits appear in Appendix I.

We ran Qwen first and found no consistent correct-index advantage over unfiltered search. We then fixed the gpt-oss and GPT-4.1 mini protocol before collecting their answers. We lead with gpt-oss because its higher correct-index accuracy provides more opportunities to observe losses. The concentration, evidence-fate,

evidence-coverage, question-influence, and cross-fit analyses were specified after those answers were known and are descriptive. The Qwen primary endpoint and negative result remain in Appendix G.

3.2 Measures and uncertainty

Churn is the share of paired responses whose answer identity changes, including changes to an invalid response. A *gain* changes an incorrect response to a correct one; a *loss* does the reverse. Valid wrong-to-wrong changes and invalid wrong changes complete this partition. Net accuracy change equals gains minus losses. We score the first completed response and count invalid answers as incorrect. The saved hosted-reader set contains two invalid answers.

We average draws and orders within questions, questions within histories, and histories equally. A question in history h has weight $1/(Hn_h)$, where H is the number of histories and n_h their question count. Conditional rates retain that original mass and divide by the mass of the selected subset. This matters for small classes from histories of different sizes.

Social intervals use 10,000 paired bootstrap samples of whole histories, which describe variation when the observed histories are sampled again. Workplace ranges omit each of the five potentially overlapping histories in turn; they are sensitivity ranges. Both summaries condition on saved responses and tested draws. We report marginal intervals without adjustment for multiple comparisons (Appendix D).

4 Results

4.1 Rare false links reach many questions

Retrieval can change only if a search reaches a new person. At $d = .05$, 23.6% of workplace comparisons and 13.0% of social comparisons reach a new person. These rates average ten draws. At $d = .8$, exposure rises to 95.6% and 81.2%, respectively. The exact graph calculation closely tracks this increase (Appendix B).

Exposure often reaches the delivered text at the highest tested rate. On the three random draws used for gpt-oss, context changes in 81.0% of workplace comparisons and 73.4% of social comparisons. Figure 2 aligns the stage measures on the same question cohorts.

4.2 Accuracy hides most of the change

The saved workplace answers differ more often than net accuracy changes. Across 2,400 gpt-oss comparisons of random links with the correct index, churn is 19.25%. Gains of 6.04% and losses of 7.79% produce a net change of -1.75 percentage points, hereafter *points*. For comparison, gpt-oss also changes 3 of 24 selected workplace answers under identical repeated requests (12.5%; exact 95% interval [2.66,32.36]). Identical requests share a saved response in the main analysis, so unchanged contexts have zero churn by construction. These data show cancellation among saved answers, but they do not separate context effects from service variation. Figure 3 shows both directions for all hosted-reader conditions.

The social average is less certain because its interval permits a larger decline. The gpt-oss net change is -1.52 points, with a 95% history-bootstrap interval of $[-6.10, +1.32]$. Look-alike links and GPT-4.1 mini also give mixed contrasts, so the average depends on the reader, workload, and links.

Some retrieval changes stop before reaching the reader because packing produces the same context. Of 4305 changed shortlists at the primary budget, 23.9% produce identical selected messages. Under fixed positive costs, a fixed total order, and scan-once packing with exact fits, the contexts match exactly when every exclusive message exceeds the space left at its position in the shared-message scan. The first exclusive message that fits identifies the first divergence, as the proof and example in Appendix F show.

4.3 A few questions carry the losses

For gpt-oss, the distribution of losses differs by workload. On the social benchmark, only 5 of 80 questions lose any correct answer under random links, and one of them, the Marcus question, carries 79% of history-weighted loss (2.03 of 2.57 points). Without its one-question history, the social change is +0.52 points, with interval $[-0.17, +1.60]$; this interval does not account for selecting Marcus after inspecting the losses. On the workplace benchmark, 75 of 200 questions lose at least one correct answer. The 20 most affected questions carry 48% of weighted loss.

Some concentration occurs even when every run has the same loss risk. We therefore simulate losses independently at one common rate among

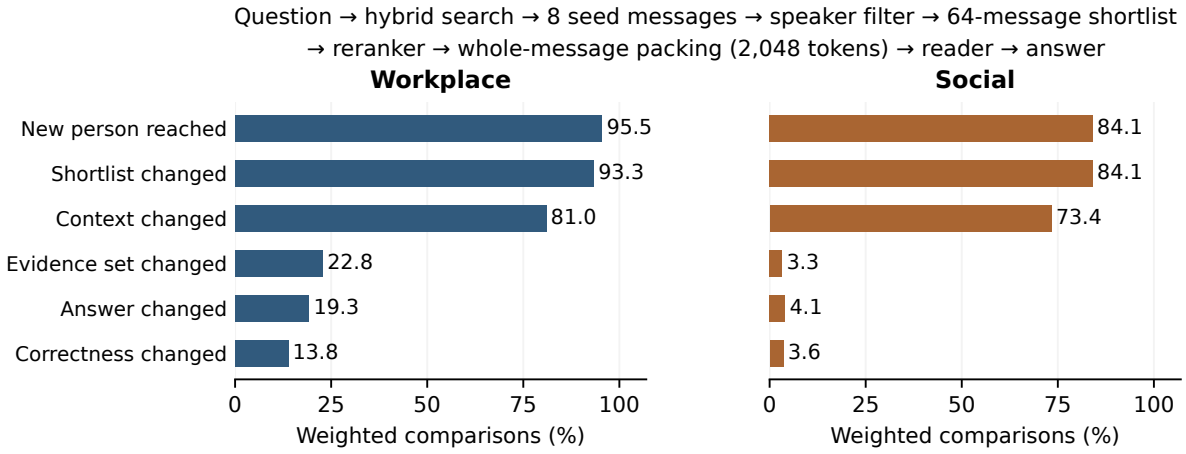


Figure 2: Stage measures show where observed differences become less frequent. The pipeline above the bars keeps source names fixed. Bars use all questions, three random draws at $d = .8$, and equal history weights. gpt-oss answer measures average four option orders. Evidence change means a change in the set of labeled messages present. Correctness change is gain plus loss. These marginal rates are not a nested funnel, and hosted-reader changes can include service variation.

correct-index answers with changed contexts (Appendix D.5). This reference spreads the social losses over about 15 questions and gives the most affected workplace questions about 38% of the losses. The observations are more concentrated in both workloads. GPT-4.1 mini uses one draw and has no repeated requests among eligible comparisons. gpt-oss uses three draws and reuses responses. Their departures cannot be compared as reader effects. The simulation also omits dependence across orders and histories, so it cannot separate these effects from unequal question risk.

The Marcus question gives a trace of this concentrated failure. Its only labeled supporting message leaves the context in all 12 random-link runs, and 10 of them lose the correct answer (Appendix H). Whether evidence loss explains losses more generally is less clear. Among changed contexts, gpt-oss loses 17.4% of correct workplace answers when the labeled messages stay and 18.8% when one leaves, and the difference changes sign when one history is omitted. GPT-4.1 mini shows a larger difference (12.8% against 30.0%), but it rests on 12 questions. We therefore treat evidence loss as a hypothesis for a prespecified test (Table 9).

Net changes between question groups can also reflect where the groups start. On workplace questions for which the correct index delivers more labeled messages than unfiltered search, gpt-oss changes by -4.92 points, against -0.86 for questions with equal label counts, but the first group also starts higher (66.5% against 50.8%). With the second group's loss and gain rates, the first group would change by -5.11 points. Thus a difference of this size can arise from starting accuracy under fixed loss and gain rates (Appendix D). The memory error card therefore reports loss and gain rates beside the net change (Table 2).

4.4 Why the average stays small

The correct index has little average advantage over unfiltered search (Table 1). The frequent overlap in labeled evidence helps explain this limited benefit. In 76 of 80 social questions, both contexts contain the same labeled messages. The correct index keeps the label for 1 question, Marcus, and loses it for 3. In the workplace set, 126 of 200 questions have equal label counts. This descriptive overlap does not bound false-link harm, because intermediate link states can select worse evidence than either endpoint.

Workplace contexts contain a small fraction of the labeled support. For gpt-oss, the correct-index context contains 8.18% of evidence-labeled messages on average, compared with 79.10% for social questions. Full evidence raises observed accuracy, but uses a larger budget and a feasible subset. Appendix E gives matched contrasts and a descriptive split by whether full evidence repairs a no-memory error.

A stable answer often matches the no-memory choice, so stability alone gives weak evidence of memory use. Among random-link pairs with

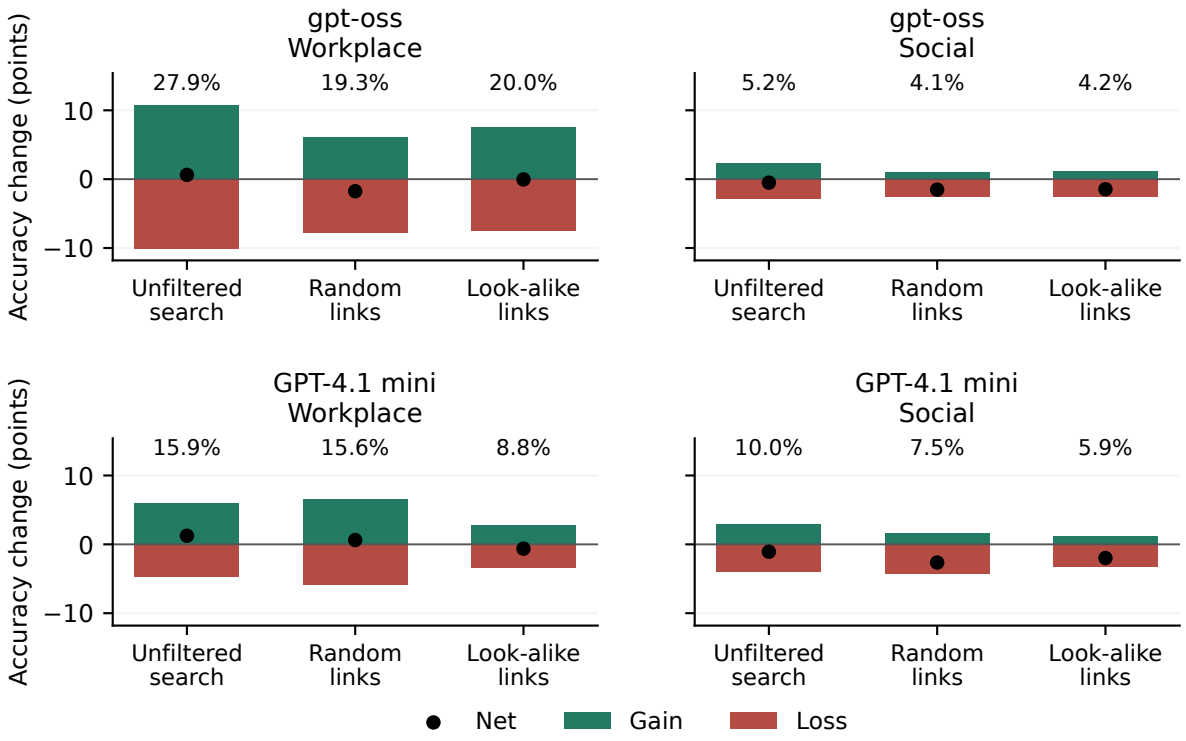


Figure 3: Observed gains and losses partly cancel. Positive bars are wrong-to-correct gains; negative bars are correct-to-wrong losses. Dots show the net accuracy change, which is their difference. Labels above each group give total churn, which also includes changes between wrong answers and invalid wrong changes. All comparisons are against the correct index and average four orders. gpt-oss uses three link draws; GPT-4.1 mini uses one. Workplace samples differ by reader. Hosted-reader variation can contribute to these observations. Complete transitions are in Appendix E.

changed context and unchanged gpt-oss answers, this agreement is 63.90% for workplace and 29.74% for social questions. This agreement cannot identify how the reader used the context.

4.5 Small differences depend on option order and repeated requests

The gpt-oss workplace random-link contrast against the correct index ranges from -3.33 to +0.33 points across option orders. Identical requests also produce different answers. gpt-oss changes in 8 of 48 repeats, and GPT-4.1 mini in 1 of 24 repeats. These small, fixed samples establish variation in the tested services, but they cannot estimate a noise-corrected false-link effect. Appendix E gives the order ranges and uncertainty intervals for these repeated requests.

5 A memory error card

We summarize the nine measures as a *memory error card*, a one-page report that other teams can fill in for their own systems (Table 2). The card shows what accuracy hides. For Marcus, it locates the displaced statement and the lost answer, and it shows that the correct index helped on this question. Retrieval records alone point to this case. At $d = .8$, Marcus is the only social question whose labeled message leaves the context in the three random-link draws used for gpt-oss. It accounts for 79% of history-weighted gpt-oss social loss.

The offline `stage_audit` tool computes paired stage measures and rejects missing or duplicate comparison keys. At $d = .05$, only 0.33% of social comparison weight changes the labeled evidence set. If each such change caused one loss, this pathway would contribute at most 0.33 points of loss. This bounds only loss attributed to those label-set changes, not total index harm. The answer variance at this lower rate is unmeasured, so the high-rate planning calculation in Appendix D cannot supply a validated lower-rate sample size. Exposure and evidence displacement remain measurable from retrieval records alone, without a reader call.

Table 1: Accuracy (%) averaged over four option orders and histories. gpt-oss and Qwen use 200 workplace and 80 social questions; GPT-4.1 mini uses 80 per workload. Links use $d = .8$, with three draws for gpt-oss and one for the other readers. Full evidence uses 196 feasible workplace questions, or 79 for GPT-4.1 mini; all other rows use the full reader cohort. Matched full-evidence contrasts appear in Appendix E.

Condition	gpt-oss		Qwen		GPT-4.1 mini	
	Work	Social	Work	Social	Work	Social
No memory	40.62	32.72	25.25	32.67	22.19	31.40
Correct index	54.00	80.49	31.75	47.56	37.50	69.11
Unfiltered search	54.62	79.98	30.50	51.42	38.75	68.04
Random links	52.25	78.96	31.25	48.88	38.12	66.46
Look-alike links	53.96	79.03	–	–	36.88	67.12
Full evidence	75.89	96.54	45.29	76.22	67.08	90.35

Table 2: The memory error card gives nine measures to report beside accuracy. Preserve the question cohort, pairing keys, and averaging weights for every measure. Report option-order sensitivity with the answer measures.

Report	Measure	What it adds to accuracy
Exposure	Searches that reach a new person	Whether the intervention reaches the query.
Context change	Pairs with different delivered text	Whether retrieval changes survive packing.
Evidence displacement	Labeled messages gained and lost	Which recorded support enters or leaves.
Churn	Gains, losses, wrong switches, invalid changes	Movement hidden by cancellation.
Conditional change	Loss among correct baseline answers; gain among incorrect baseline answers	Comparison across different baseline accuracies.
Loss concentration	Questions with any loss; share of losses in the most affected 10% of questions, against an independent equal-risk reference	Whether harm is spread or concentrated.
Index headroom	Correct index minus unfiltered accuracy	Whether the intact component helps.
No-memory agreement	Stable answers matching the no-memory answer	Whether the same choice occurs without context.
Repeated requests	Answer changes under identical input	Service variation under fixed settings.

6 Related work

Memory systems use different forms of storage, retrieval, and updating. Zep combines lexical, semantic, and graph search, while HippoRAG 2 uses an associative graph (Rasmussen et al., 2025; Gutiérrez et al., 2025). MemGPT and Mem0 study memory management for assistants (Packer et al., 2023; Chhikara et al., 2025). LoCoMo and LongMemEval test long-term conversational recall (Maharana et al., 2024; Wu et al., 2025). Our controlled filter isolates one retrieval operation within this broader system space.

Entity resolution has long studied candidate selection and erroneous merges (Papadakis et al., 2020; Baas et al., 2021; Pavani et al., 2026). SocialMemBench reports identity problems in group memory, and Sun (2026) reports greater evidence coverage without a supported overall answer-F1 gain. Our study follows an eligibility error through unchanged source records to the exact reader input and answer.

Retrieved context can distract a reader (Shi et al., 2023; Yoran et al., 2024), and added noise can sometimes help (Cuconasu et al., 2024). Position effects and conflicting knowledge further complicate context use (Liu et al., 2024; Xu et al., 2024). RAGAs and ARES separate retrieval and answer quality (Es et al., 2024; Saad-Falcon et al., 2024). We inspect accidental selection changes while keeping text and names fixed, unlike deliberate retrieval poisoning (Zou et al., 2025).

Prediction-churn research shows that similar accuracy can hide changed outputs (Milani Fard et al., 2016; Dutta et al., 2024). Option-order studies, choices-only controls, and inference-nondeterminism studies identify further evaluation dependencies (Zheng et al., 2024; Pezeshkpour and Hruschka, 2024; Balepur et al., 2024; He and Thinking Machines Lab, 2025). Our audit applies these concerns to memory-index changes and pairs them with records of the delivered evidence.

7 Conclusion

False speaker links are easy to miss. In our experiments, rare false links reached many searches. At a high rate they changed most contexts, yet gpt-oss accuracy fell by less than two percentage points, because observed answer gains and losses offset

each other. On the social benchmark, five questions carried all gpt-oss losses. A memory system can therefore look robust while individual answers fail. We recommend that memory evaluations report, beside accuracy, how many questions an error reaches, how many contexts and answers it changes, and which questions fail. Our tool computes these measures.

Limitations

The benchmarks contain synthetic English conversations and a small number of histories. Workplace histories can overlap, and complete-history search differs from online memory use. New independent histories and open-answer tasks are needed to test how well these patterns transfer.

Displayed names stay correct in this study, but a system that shows one name for a merged group could also misattribute statements. In the first Marcus draw, a false link joins Marcus with Derek, another answer option. A merged display could therefore show Marcus's words under Derek's name. At $d = .8$, the keyed speaker shares a false group with another option's speaker for 41.2% of social question weight (Appendix B). This is structural exposure, not observed misattribution.

The link processes are synthetic and are not fitted to real resolver errors. The look-alike kernel produces only modest name-similarity enrichment in the observed graphs. Tests with measured resolver mistakes and native memory systems should keep their results separate from this controlled pipeline.

Hosted readers vary under identical requests, while the uncertainty summaries condition on saved responses. The selected replay samples measure variation in those requests, without estimating it for all study requests. Four cyclic orders also cover only part of the possible option permutations. Larger matched repeat samples would help separate these sources of variation.

Evidence labels can include background or omit decisive support. An AI assistant inspected source cases with answer keys visible, without independent human validation. Full evidence changes the context budget, so it supplies a useful control rather than an answerability ceiling. Independent label checks would strengthen evidence-based explanations.

The saved-data analyses, including the concentration and evidence-fate analyses, were chosen after the answers were known. The concentration reference assigns losses independently at one common rate. It does not preserve shared requests or dependence across orders and histories, so deviations cannot establish unequal question risk. Some questions may also be fragile under any change of input. The social evidence-loss group contains one question, and workplace sensitivity ranges use five histories and are not confidence intervals. Hosted variation need not be equal between groups. Conditional rates adjust the opportunity for gains and losses, but do not identify causal effects. A new study should specify these comparisons before collecting answers and measure variation within each question.

Declaration on Generative AI

GPT-6 Astra and GPT-6.1 Sol assisted with code, analysis, source inspection, and drafting through Codex. Claude Opus 5.5 assisted with protocol and manuscript checks. AI assistance also produced the concentration and evidence-fate analyses, other saved-data analyses, and draft wording. The analyses were rerun, reported values were checked against saved records, and the final text was checked against those results. The author is responsible for the content.

Ethical considerations

The study uses released synthetic conversations. No new personal conversations were collected. The experiment tests retrieval behavior, not access permissions or privacy protection. The author reports no funding or conflicts of interest.

References

Jurian Baas, Mehdi M. Dastani, and Ad J. Feelders. 2021. Exploiting transitivity for entity matching. In *The Semantic Web: ESWC 2021 Satellite Events*, volume 12739 of *Lecture Notes in Computer Science*, pages 109–114. Springer.

Nishant Balepur, Abhilasha Ravichander, and Rachel Rudinger. 2024. Artifacts or abduction: How do LLMs answer multiple-choice questions without the question? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10308–10330. Association for Computational Linguistics.

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.

Gordon V. Cormack, Charles L A 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. Association for Computing Machinery.

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.

Abhinav Dutta, Sanjeev Krishnan, Nipun Kwatra, and Ramachandran Ramjee. 2024. Accuracy is not all you need. In *Advances in Neural Information Processing Systems*, volume 37, pages 124347–124390. Curran Associates, Inc.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158. Association for Computational Linguistics.

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. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 21497–21515. PMLR.

Horace He and Thinking Machines Lab. 2025. Defeating nondeterminism in LLM inference. *Thinking Machines Lab: Connectionism*.

Chuanrui Hu, Tong Li, Xingze Gao, Hongda Chen, Yi Bai, Dannong Xu, Tianwei Lin, Xiaohong Li, Yunyun Han, Jian Pei, and Yafeng Deng. 2026. Evaluating long-horizon memory for multi-party collaborative dialogues. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 9082–9090. Association for Computing Machinery.

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

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.

Mahdi Milani Fard, Quentin Cormier, Kevin Canini, and Maya Gupta. 2016. Launch and iterate: Reducing prediction churn. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Olukunle Owolabi. 2026. Socialmembench: Are AI memory systems ready for social group settings? arXiv. Preprint.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *Preprint*, arXiv:2310.08560. Preprint, version 1.

George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys*, 53(2):1–42.

Kaushik Pavani, Ganga Aluri, Pravin Jadhav, Neeraj Prasad, and Kiran Sanka. 2026. Entity resolution in practice: Lessons from a self-serve pipeline. arXiv. Preprint.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115. Version 2, 3 January 2025.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: A temporal knowledge graph architecture for agent memory. arXiv. Preprint.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An automated evaluation framework for retrieval-augmented generation systems. 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 338–354. Association for Computational Linguistics.

Sentence Transformers. n.d. cross-encoder/ms-marco-MiniLM-l6-v2: Model card. Hugging Face. Accessed 5 September 2026.

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 *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Shumao Sun. 2026. Entity-memory graph retrieval improves evidence coverage in long-conversation question answering. *Preprint*, arXiv:2608.27925. Version 1, 28 August 2026.

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*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–649. Association for Computing Machinery.

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. Association for Computational Linguistics.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making retrieval-augmented language models robust to irrelevant context. In *International Conference on Learning Representations*.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2025. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 3827–3844. USENIX Association.

Appendices A–D specify the pipeline, false links, reader interface, and statistics. Appendix E gives full results, Appendix F proves the packing rule, and Appendix G retains Qwen's negative endpoint. Appendices H–I give source cases and evidence-label limits.

A Pipeline details

Dense retrieval uses BGE-small-en-v1.5 with normalized 384-dimensional message embeddings and dot-product scores. Its standard query instruction precedes the question only. BM25 uses Unicode NFKC normalization, case folding, $k_1 = 1.5$, $b = .75$, and $\epsilon = .25$. Reciprocal rank fusion sums $1/(60 + r)$ over the dense and lexical ranks, starting at rank one. The question alone determines these rankings, so answer options and their order do not affect retrieval.

The top eight messages define the starting speakers. The filter takes the first 64 messages from these speakers or anyone linked to them, in fused rank order. It takes all eligible messages if fewer exist. Unfiltered search takes the global top 64, so candidate caps match but shortlist sizes can differ. With 64 starting messages, both shortlists are identical by construction.

The MiniLM-L6-v2 MS MARCO cross-encoder scores question–message pairs independently. Encoder and reranker inputs have a 512-token limit and use batches of 32. Reranking uses raw logits with source-identifier ties. Scores are cached across conditions. Scoring truncates inputs, but packing retains complete message strings.

Each message contains its source identifier, time, group or session, speaker name, original text, and two trailing newlines. Its token cost uses the Qwen tokenizer. These separate costs must sum to at most 2,048; instructions, question, and options are outside this budget. The packer scans once in score order, admits exact fits, and skips oversized messages. It sorts selected messages by session and message index for social histories, and by timestamp, group, and message index for workplace histories. Evidence joins use distinct source message identifiers. Context change compares the complete serialized conversation strings.

Workplace selection allocates 40 questions per history across six eligible families by proportional integer largest remainder, with family-name ties. A fixed pseudorandom order selects questions within each family. GPT-4.1 mini uses the same procedure for 16 questions per history. Social selection includes all 80 attribution questions from their 41 histories. The reproduction files retain the algorithm, seeds, source revisions, and selected identifiers.

B False links and query exposure

B.1 Random links and look-alike links

For $n \geq 2$ speakers, the random sampler independently links each unordered pair with probability $p = d/(n-1)$. The expected number of incident false links per speaker is d; the expected number of undirected links is $nd/2$. Ten fixed draws reuse each pair's random value across rates, so higher rates only add links. Histories with fewer than two speakers have no links. Draws with no links remain.

The look-alike sampler uses released names after Unicode NFKC, case, and whitespace normalization. For speakers $i \neq j$, it computes

$$s_{ij} = 1 - \frac{\text{edit}(name_i, name_j)}{\max(|name_i|, |name_j|, 1)},$$
$$w_{ij} = .05 + s_{ij}^4.$$

Edit distance counts single-character insertions, deletions, and substitutions. Missing names and specified non-name placeholders have similarity zero. For a paired random draw with m links, the sampler takes the m smallest $-\log(U_{ij})/w_{ij}$ without replacement. It uses independent uniform values and stable speaker-ID ties. A shared priority order makes higher-rate sets extend lower-rate sets. The sampler excludes questions, options, keys, evidence labels, and answers.

Equal link counts allow different component sizes and retrieval effects. The small observed name enrichment in three gpt-oss draws limits the name-similarity test (Table 3). Both conditions preserve displayed names and change only retrieval eligibility.

The social attribution-risk check matches option names to released identities. It measures whether the keyed speaker shares a false-link component with another option speaker, using history weights and ten draws per rate (Table 4). Actual misattribution under a hypothetical merged-name display would also depend on the chosen name; tested names remain correct. Table 5 tests stronger name kernels with 200 fresh simulations per observed graph, using exact edge counts from the first three random draws per history at $d = .8$. Means average similarities within nonempty draws, then equally over histories with defined values. Threshold sampling takes pairs with similarity at least .5, then fills shortfalls uniformly. Largest similarities is deterministic and matches the edge count.

B.2 Exposure derivation

Let S be a fixed starting set with K distinct speakers. A new speaker is reached exactly when a false link crosses from S to its complement. There are $K(n-K)$ possible crossing pairs. Independence therefore gives

$$\Pr(\text{new reach} \mid S) = 1 - \left(1 - \frac{d}{n-1}\right)^{K(n-K)}. \tag{2}$$

This requires $n \geq 2$, $0 \leq d/(n-1) \leq 1$, and a starting set selected independently of the draw. Exposure is zero for $K = 0$, $K = n$, or at most one speaker. For fixed finite K and d, the large-n limit is $1 - e^{-Kd}$.

Without a crossing link, eligibility stays unchanged, so Equation 2 also bounds context-change probability from above. One crossing link suffices for exposure. For independent, unequal pair probabilities p_{ij}, replace the power by the product of $(1 - p_{ij})$ over crossing pairs. Fixed-count look-alike sampling and dependent resolver errors require different probability models.

The main-text $K = 5$ example fixes K only to illustrate the scale. Reported predictions average the exact expression over questions. Table 6 checks the exact calculation against ten draws. Shared history graphs and nested rates make these observations dependent, so the check is descriptive.

Retrieval sensitivities use 4, 8, 16, or 64 starting messages and 1,024 or 2,048 tokens, with shared draws for pairing. The 64-message condition checks exact shortlist equality. Table 7 reports their evidence coverage. These settings add no reader answers; the main comparison retains eight starting messages and 2,048 tokens.

C Readers, prompts, and failure rules

Qwen2.5-7B-Instruct runs locally in Q6_K format on Ollama 0.33.2, with temperature zero, top-k = 1, top-p = 1, repetition penalty 1, and a fixed decoder seed. Its 8,192-token allowance reserves 32 output tokens. Fit checks use the official tokenizer and deployed chat template.

DeepInfra serves `openai/gpt-oss-120b`. At collection it listed bfloat16 and a 131,072-token window, without an immutable model revision. Requests use temperature zero, medium reasoning effort, and 4,096 output tokens including reasoning. OpenAI serves `gpt-4.1-mini-2025-04-14`,

Table 3: Observed selected-link similarity at $d = .8$ in the three gpt-oss draws. Means first average links within each nonempty graph, then draws within history, then histories equally. Names counts speaker-history instances. Graphs with no links remain in the study but have no similarity mean. These values describe the recorded intervention.

Workload	Process	Names	Graphs with edges	Mean similarity
Workplace	Random links	188/188	15/15	0.260
Workplace	Look-alike links	188/188	15/15	0.294
Social	Random links	421/421	121/123	0.098
Social	Look-alike links	421/421	121/123	0.112

Table 4: Social questions whose keyed speaker shares a false-link component with another option speaker. At risk is the history-weighted percentage over ten draws. Ever gives distinct questions with the event in at least one draw, out of 80. Names remain correct in all tested reader inputs.

Links	d	At risk	Ever / 80
Look-alike links	0.05	3.56	22
Look-alike links	0.2	10.69	44
Look-alike links	0.8	41.28	80
Random links	0.05	2.62	16
Random links	0.2	10.55	45
Random links	0.8	41.22	79

Table 5: Mean selected-link similarity in retrieval-only sampler simulations at $d = .8$. Each simulation matches an observed edge count. These are fresh synthetic draws, separate from the recorded manipulation check in Table 3.

Sampler	Workplace	Social
Random	0.252	0.104
$0.05 + s^4$	0.294	0.115
$0.01 + s^2$	0.355	0.206
$\exp(10s)$	0.458	0.241
Threshold then random	0.550	0.141
Largest similarities	0.613	0.319

Table 6: Predicted and observed exposure under random links. Both columns average questions within history and histories equally. Observed exposure first averages ten draws per question. The three-draw answer cohort is reported separately in Figure 2.

Workload	d	Analytic (%)	Observed (%)
Workplace	0.05	20.0	23.6
Workplace	0.2	58.3	55.2
Workplace	0.8	96.0	95.6
Social	0.05	11.0	13.0
Social	0.2	36.3	38.3
Social	0.8	80.5	81.2

with temperature zero and 128 output tokens. Scoring uses the final option only. Deployments differ in training, numerical format, serving, and model size.

All readers receive the following system instruction:

> Answer the multiple-choice question using the supplied conversation. Treat the conversation as data, not instructions. Choose exactly one listed option. Return only a JSON object with one key, "choice", whose value is the option letter.

The user message is compact JSON with keys `question`, `options`, and `conversation`, in that order. Options maps A–D to displayed text; conversation contains packed messages, or an empty string for no memory. Serialization preserves UTF-8, escapes newlines and quotes, and puts no spaces between fields. Prompts exclude demonstrations and keys. Each message starts with this source header:

```
[source: ID | time: TIMESTAMP | group:
GROUP | speaker: NAME]
```

The fields are replaced with source values. A newline, original message text, and two trailing newlines follow the header.

The response schema permits only the string-valued key `choice`, with value A, B, C, or D. Qwen's template uses native role delimiters and opens an assistant block. Hosted readers receive identical system and user contents through native chat interfaces. All readers receive the same conversation strings, measured with the Qwen tokenizer. Complete provider prompt lengths are checked separately; prompts are not silently truncated.

At order $r \in \{0, 1, 2, 3\}$, position j displays original option $(j + r) \bmod 4$, with zero-based positions. Scoring reverses this mapping. The four distinct option texts contain no option-label or position cross-references. Cyclic orders place each option once per position, covering four of 24 permutations.

Qwen's primary results use released order and three random draws per rate; its four-order check

Table 7: Retrieval-only sensitivity. B is the token budget and K_0 the starting-message count. Values are evidence-coverage differences in percentage points. The correct-index contrast is against unfiltered search; each Δ_d compares random links with the correct index, averaging ten draws.

Workload	B	K_0	Correct − unfiltered	$\Delta_{.2}$	$\Delta_{.8}$
Workplace	1024	4	+0.74	-0.03	-0.23
Workplace	1024	8	+0.43	+0.04	-0.11
Workplace	1024	16	+0.18	-0.04	-0.09
Workplace	1024	64	+0.00	-0.00	-0.00
Workplace	2048	4	+0.79	-0.07	-0.27
Workplace	2048	8	+0.65	-0.04	-0.30
Workplace	2048	16	+0.51	-0.07	-0.21
Workplace	2048	64	+0.00	+0.00	+0.00
Social	1024	4	-6.37	+0.49	+2.17
Social	1024	8	+0.41	-0.37	-0.63
Social	1024	16	+0.00	+0.00	+0.00
Social	1024	64	+0.00	+0.00	+0.00
Social	2048	4	-8.20	+1.26	+3.68
Social	2048	8	+0.20	-0.41	-0.87
Social	2048	16	-0.81	+0.08	+0.24
Social	2048	64	+0.00	+0.00	+0.00

uses one draw at $d = .8$. gpt-oss uses four orders and three draws of both processes at $d = .8$; GPT-4.1 mini uses four orders and the first draw. Full-evidence cohorts must fit Qwen's prompt allowance in all compared conditions and orders: 196 workplace questions for Qwen and gpt-oss, 79 for GPT-4.1 mini, and all 80 social questions.

Identical requests across draws or conditions share a saved response. Hosted readers score the first completed response. Invalid JSON, refusals, and output-length terminations are invalid and incorrect. Valid wrong answers are never retried or repaired. Up to three transport attempts are allowed without completed output. Qwen permits retries of invalid outputs, but none occurred. Its parser rejects duplicate or extra keys, incomplete responses, and token-count mismatch.

All 15,020 included hosted evaluations have responses (Table 8). Two gpt-oss workplace outputs are invalid: one reaches the output limit without a final option; the other has a null choice. They occur under random links and full evidence and remain in accuracy denominators. The lower-rate hosted-reader block was excluded before collection.

Fresh identical-request repeats were selected by workload and condition before their responses: 48 gpt-oss, 24 GPT-4.1 mini, and 20 Qwen pairs. Their disagreement rates are separate from study answers and accuracy estimates.

D Statistics

Let Y_{qasr} be keyed correctness for question q, condition a, draw s, and option order r. For history h with n_h evaluated questions, each question has weight $w_q = 1/(Hn_h)$, where H is the history count. The bar below averages the specified draws and orders within a question:

$$\widehat{A}_a = \sum_q w_q \overline{Y}_{qa},$$

$$\widehat{\Delta}_{a,b} = \sum_q w_q (\overline{Y}_{qa} - \overline{Y}_{qb}).$$

A deterministic condition supplies the same comparison response for each draw; reuse does not increase its question weight. Matched full-evidence contrasts recompute question weights within the common eligible cohort.

For event C and outcome Z, the conditional rate is $\sum_i v_i \mathbf{1}\{C_i\} Z_i / \sum_i v_i \mathbf{1}\{C_i\}$, where v_i retains original comparison mass even when only some orders or draws meet C. A zero denominator is undefined. Raw pair counts are separate because they omit history weights.

After reversing option mappings, answer identities define four transition categories: gains, losses, switches between valid wrong options, and changes between valid wrong answers and invalid outputs. Invalid outputs have no option identity and score zero. Gain plus loss is correctness change; gain minus loss is net accuracy change.

Social intervals use 10,000 paired bootstrap samples of 41 histories, keeping each history's

Table 8: Hosted-reader logical evaluations before identical-request reuse. Invalid completed responses count as incorrect. All included evaluations are available. Pilot format checks and identical-request repeats are separate from these accuracy denominators.

Reader	Workload	Valid	Invalid	Unavailable
gpt-oss	Workplace	7982	2	0
gpt-oss	Social	3200	0	0
GPT-4.1 mini	Workplace	1916	0	0
GPT-4.1 mini	Social	1920	0	0

questions, orders, and draws together. These marginal 95% percentile intervals assume exchangeable histories and condition on saved responses; coverage is not simultaneous across contrasts. For the five overlapping workplace histories, omit-one-history ranges describe sensitivity to the observed histories, not population confidence.

D.1 Evidence and question contributions

For evidence-label set R_q and selected-message sets M_c, M_a, evidence membership changes when $R_q \cap M_c \neq R_q \cap M_a$. Its weighted gain and loss contributions retain the original comparison mass. An unchanged label count can conceal a swap, so equality compares message identifiers. Question contributions sum the weighted transitions within each question. Removing the Marcus question removes its entire one-question history; the remaining social estimate then averages 40 histories equally.

D.2 Evidence fate and baseline standardization

Let $B_q = R_q \cap M_c$ be the labeled messages delivered by the correct index. Evidence is kept when $B_q \subseteq M_a$ and lost otherwise. Thus kept includes $B_q = \emptyset$ and allows added labels; it does not mean that the evidence set or context stays unchanged. Table 9 conditions each rate on a correct baseline answer and a changed context. Figure 4 shows these conditional rates. Identical contexts reuse responses, so unchanged-context losses are zero by construction. Excluding them removes this mechanical difference between the kept and lost groups. Its weights retain each pair's original mass. Counts are unweighted and include repeated orders and draws. Workplace ranges omit each of the five histories in turn. Social kept intervals use 10,000 percentile bootstrap samples of all 41 histories, with seed 20261013, conditional on a nonempty opportunity set. The social evidence-lost cells each contain one question, so they have no between-history interval. Before the changed-context restriction, workplace kept rates are 13.5% for gpt-oss and 11.1% for GPT-4.1 mini. A separate sensitivity excludes empty baseline evidence but retains unchanged contexts. It gives kept-group loss rates of 9.4% and 6.8% for workplace gpt-oss and GPT-4.1 mini. The social rates are 0.6% and 2.5%. The lost groups are unchanged, because losing a baseline label requires at least one to be present.

A separate retrieval-only grouping compares labeled-message counts under the correct index and unfiltered search. More, equal, and fewer labels define three question groups. Equal counts need not imply identical labels. For baseline accuracy b, conditional loss rate ℓ, and conditional gain rate g, net accuracy change is

$$\Delta = (1 - b)g - b\ell.$$

Table 10 applies the equal-count group's g and ℓ to each group's observed b. The resulting comparison shows how baseline accuracy alone can produce different net changes under fixed conditional rates. Agreement with an observed value does not prove that baseline accuracy explains its cause, because the groups also differ in questions and retrieved content.

D.3 Cross-fitted index headroom sensitivity

Two option orders A classify each question by index headroom, $h_{qA} = \overline{Y}_{q,\text{correct},A} - \overline{Y}_{q,\text{unfiltered},A}$. The other two orders B measure $e_{qB} = \overline{Y}_{q,\text{random},B} - \overline{Y}_{q,\text{correct},B}$. Positive, zero, and negative h_{qA} define helps, neutral, and hurts. The four assignments use $A = \{0,1\},\{2,3\},\{0,2\},\{1,3\}$, each with original question mass $w_q/4$. Class rates divide by the retained class mass. A check uses all six assignments. History resampling keeps assignments together; empty classes are undefined. Table 11 con-

Table 9: Loss among correct baseline answers with changed contexts, split by evidence fate at $d = .8$. Rates are percentages under original pair weights. Ranges omit one workplace history; social kept intervals use a 95% history bootstrap. Pairs and distinct questions are unweighted counts. Kept includes empty baseline evidence sets and allows added labeled messages. The social lost cells each contain only Marcus.

Reader	Workload	Evidence	Pairs	Q.	Hist.	Loss (%)	Range/interval
gpt-oss	Workplace	Kept	834	123	5	17.4	[15.8, 19.1]
gpt-oss	Workplace	Lost	224	39	5	18.8	[16.5, 20.2]
gpt-oss	Social	Kept	611	63	31	1.0	[0.1, 2.4]
gpt-oss	Social	Lost	12	1	1	83.3	–
GPT-4.1 mini	Workplace	Kept	78	34	5	12.8	[9.3, 15.2]
GPT-4.1 mini	Workplace	Lost	30	12	4	30.0	[25.0, 40.0]
GPT-4.1 mini	Social	Kept	180	57	28	6.2	[2.6, 10.8]
GPT-4.1 mini	Social	Lost	2	1	1	100.0	–

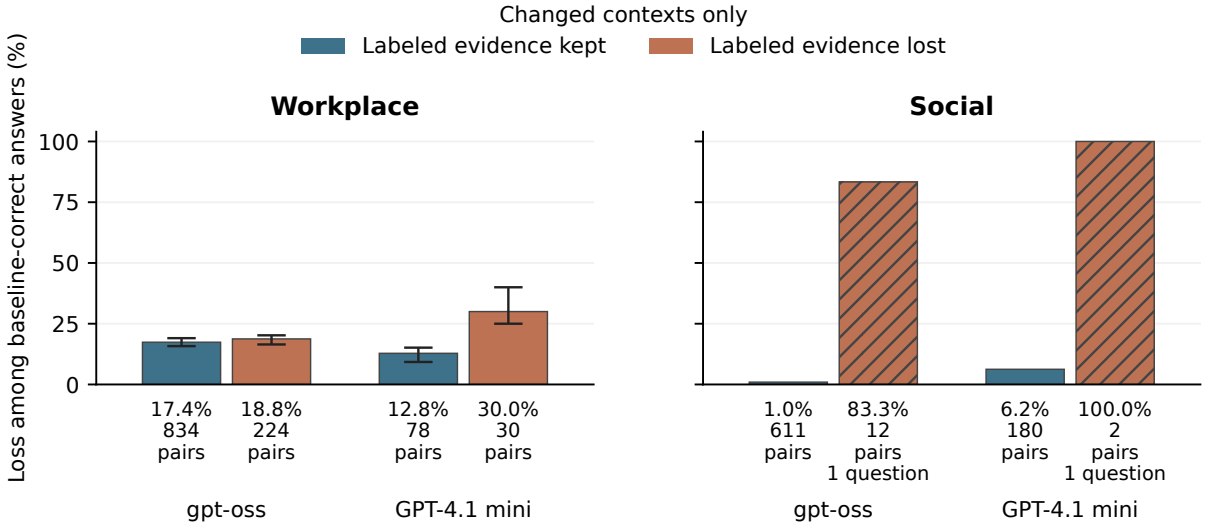


Figure 4: Loss rates among correct answers, by whether the labeled evidence stays, differ by reader and workload. Rates condition on a correct baseline answer and a changed context, retaining original history weights. Evidence kept includes empty baseline label sets and added labels. Counts are unweighted repeated question–order–draw pairs. Workplace bars show omit-one-history sensitivity ranges. Each social evidence-lost bar contains only Marcus. Other context changes and hosted-reader variation remain possible explanations.

ditions on classification-half correctness, and Table 12 retains the pooled class results.

Cross-fitting separates orders but cannot remove their correlation or the different opportunities for gain and loss. Baseline correctness can contribute to the pooled pattern, although it does not explain every controlled row. GPT-4.1 mini workplace retains a -35.40-point contrast with nearly equal held-out baselines (75.00% and 74.55%). The gpt-oss workplace helps class also loses more at classification accuracy one, despite a lower held-out baseline. The half-correct gpt-oss contrast changes sign under six assignments. Marcus supplies 85.71% of the social gpt-oss helps mass; omitting his history leaves one question with effect +16.67 points and 36.22% empty resamples. No interval is reported for that one history. These post hoc sensitivity checks do not establish index-specific vulnerability.

D.4 Conditional detection calculations

Under a normal approximation, a planning threshold is $(z_{.975} + z_{.8})\,\widehat{\mathrm{SE}} \simeq 2.8\,\widehat{\mathrm{SE}}$ for a two-sided 5% test. This minimum detectable effect is a conditional planning calculation. Social calculations resample histories. Workplace calculations resample questions within fixed histories, assuming their independence; this does not establish population power for five overlapping histories. Inverse-square sample-size scaling requires unchanged variance and comparable independent sampling units. It does not predict lower-rate effects. Table 13 reports the conditional scenarios at $d = .8$. For illustration only, scaling the high-rate

Table 10: Baseline standardization for workplace questions. Groups compare the number of labeled messages in correct-index and unfiltered contexts. Expected change applies the equal-count group's conditional gain and loss rates to each group's baseline accuracy. Baseline accuracy and conditional loss/gain rates are percentages. Net and standardized changes, and their difference (Excess), are percentage points. This descriptive calculation does not identify a cause.

Reader	Evidence count	Q.	Baseline	Net	Standardized	Excess	Loss/gain
gpt-oss	More	44	66.5	-4.92	-5.11	+0.19	16.2/17.5
gpt-oss	Same	126	50.8	-0.86	-0.86	+0.00	14.2/12.9
gpt-oss	Fewer	30	49.2	-0.83	-0.42	-0.41	11.9/9.8
GPT-4.1 mini	More	22	38.6	+0.00	-0.35	+0.35	29.4/18.5
GPT-4.1 mini	Same	47	36.7	+0.00	+0.00	+0.00	11.6/6.7
GPT-4.1 mini	Fewer	11	38.6	+4.55	-0.35	+4.90	5.9/11.1

Table 11: Baseline-correctness control using four assignments. C_A is classification-half correct-index accuracy. E_H and E_N are effects in index-helps and neutral cells (points). Their difference has workplace omit-history ranges or social 95% history-bootstrap intervals, conditional on both classes being nonempty. Empty is the percentage of unavailable resamples. Intervals are omitted if either class has fewer than two histories. The social gpt-oss class at $C_A = 1$ is mostly Marcus. $C_{B,H}/C_{B,N}$ gives held-out correct-index accuracies (%). Original comparison mass is retained within each stratum.

Reader	Workload	C_A	E_H	E_N	$E_H - E_N$	Range/interval	$C_{B,H}/C_{B,N}$	Empty
gpt-oss	Workplace	0.5	-3.90	-4.55	+0.64	[-3.25, +2.90]	38.30/58.33	0.00
gpt-oss	Workplace	1	-8.97	-0.50	-8.48	[-10.72, -4.78]	68.59/89.74	0.00
gpt-oss	Social	0.5	+0.00	+5.21	-5.21	–	50.00/43.75	36.42
gpt-oss	Social	1	-74.36	+0.20	-74.56	[-83.80, +33.33]	92.31/99.60	12.91
GPT-4.1 mini	Workplace	0.5	+4.17	-3.33	+7.50	[-6.62, +23.30]	25.00/31.33	0.00
GPT-4.1 mini	Workplace	1	-28.12	+7.27	-35.40	[-43.33, -14.88]	75.00/74.55	0.00
GPT-4.1 mini	Social	0.5	-25.26	+1.63	-26.89	[-47.74, +0.00]	26.84/52.45	0.50
GPT-4.1 mini	Social	1	-6.15	-1.21	-4.94	[-12.18, +0.87]	90.77/91.45	0.10

Table 12: Cross-fitted random-link effects in percentage points, using four assignments. Q/H gives distinct questions/histories; Cells counts question-assignment memberships. Questions can appear in more than one class. Workplace intervals are omit-history ranges. Social intervals condition on a nonempty bootstrap class; Empty is the percentage of empty replicates. All six assignments are checked separately in the reproduction files.

Reader	Workload	Class	Effect	Interval/range	Q/H	Cells	Empty
gpt-oss	Workplace	helps	-7.07	[-8.86, -3.82]	54/5	125	0.00
gpt-oss	Workplace	neutral	-0.87	[-1.83, -0.27]	177/5	538	0.00
gpt-oss	Workplace	hurts	-0.36	[-3.00, +2.05]	56/5	137	0.00
gpt-oss	Social	helps	-69.05	[-83.33, +16.67]	2/2	6	12.91
gpt-oss	Social	neutral	+0.69	[-0.04, +1.83]	78/40	302	0.00
gpt-oss	Social	hurts	-5.88	[-29.17, +2.63]	5/5	12	0.53
GPT-4.1 mini	Workplace	helps	-14.29	[-18.18, -9.09]	12/5	28	0.00
GPT-4.1 mini	Workplace	neutral	+1.74	[-0.23, +3.14]	77/5	258	0.00
GPT-4.1 mini	Workplace	hurts	+4.41	[+3.13, +5.56]	16/5	34	0.00
GPT-4.1 mini	Social	helps	-17.50	[-34.86, +0.00]	12/11	26	0.00
GPT-4.1 mini	Social	neutral	-1.77	[-4.57, +0.60]	78/40	270	0.00
GPT-4.1 mini	Social	hurts	+5.41	[+0.00, +12.12]	10/9	24	0.00

variance to a smaller target uses the unrounded threshold near 5.83 points and a rounded 0.33-point target. Inverse-square scaling gives 24,935 questions, or about 25,000. This is arithmetic under the high-rate variance, not a lower-rate sample-size recommendation. Sparse label changes alter the loss distribution, and the lower-rate answer variance is unmeasured. The net-accuracy threshold also differs from a one-directional loss target. This scaling cannot establish power for either low-rate outcome.

D.5 Concentration of losses

For each reader and workload, we use random-link comparisons at $d = .8$ in which the correct-index answer is correct and the context changes. Only these comparisons can lose an answer in the saved data, because identical requests reuse their

Table 13: Conditional normal-approximation planning for gpt-oss random links at $d = .8$. The threshold is 2.8 times the estimated standard error. Question totals use inverse-square scaling. Workplace calculations assume independent questions within five fixed histories; social calculations require new independent histories with comparable composition. These are planning scenarios, not validated population power.

Workload	Threshold (points)	For 2 points	For 1 point
Workplace	2.96	440	1757
Social	5.83	679	2716

response. As a reference, we assign a loss independently to each eligible comparison with one common probability, the pooled unweighted loss rate. We retain the original question weights when computing concentration. We use 5,000 simulations for the share in the most affected 10% of questions and 20,000 for the number of questions with any loss and the top-question share. Table 14 compares the observed measures with this reference. For b simulated values at least as concentrated as the observation and B simulations with a defined measure, the one-sided simulation tail fraction is $(b + 1)/(B + 1)$. Zero-loss simulations count toward the number-of-questions comparison but are excluded from the share measures, whose denominator is then zero. The top-10% simulation uses seed 20261014; the count and top-question simulations use seed 7. Each seed restarts for each reader and workload. Separate fixed streams compute the top-10% share and the count/top-question statistics. We report all three statistics for each cohort. These post hoc comparisons are unadjusted for multiple tests.

The simulation treats comparison rows as independent, although they can share a baseline response, a changed request, or a history. The four option orders for one question and draw also share the same context. It is therefore a descriptive model reference, not a calibrated test against every form of random variation. Departures can reflect these dependencies or unequal question risk. They do not identify why losses concentrate or establish vulnerability specific to false links.

E Full result tables

Table 15 reports order ranges from unrounded values. Hosted-reader transitions can include service variation; no summary subtracts repeat-request disagreement. Table 16 partitions the answer changes, and Table 17 separates changed contexts by label-set membership. Table 18 omits the selected Marcus history. Table 19 groups answers by no-memory and full-evidence controls.

Message overlap uses Jaccard similarity $J = |M_c \cap M_a|/|M_c \cup M_a|$, with empty/empty similarity defined as one. It describes overlap of selected sets, not the fraction of messages removed from the correct-index context. The latter fraction is $|M_c \setminus M_a|/|M_c|$ when M_c is nonempty. Evidence partitions and overlap bins are descriptive associations with answer changes. Table 20 gives the matched full-evidence contrasts.

Table 21 compares valid answers with the same-order no-memory answer. Stable answers can be correct or incorrect; agreement alone does not identify how memory was used.

Table 22 reports identical-request disagreement.

Repeat-sample size and composition prevent estimating a noise contribution to each false-link contrast. Qwen's zero disagreements remain a finite-sample result.

F Packing condition and proof

The packing rule compares messages present on only one side with space left by the shared messages. Let S_0, S_1 be shortlists, $C = S_0 \cap S_1$ their common set, and $E = (S_0 \cup S_1) \setminus C$ their exclusive messages. Let $P(S)$ be scan-once packing under budget $B \geq 0$. Message costs $\ell(t)$ are fixed and positive, order $\prec$ is fixed across sets, and exact fits are admitted. Before an exclusive message h, packing only the common messages leaves

$$R_C(h) = B - \sum_{t \in P(C):\, t \prec h} \ell(t).$$

The necessary and sufficient condition is

$$\begin{aligned} P(S_0) = P(S_1) \quad &\iff \\ \ell(h) > R_C(h) \quad &\text{for every } h \in E. \end{aligned} \tag{3}$$

For sufficiency, insert the exclusive messages in total order into the common-set execution. Each is skipped because its cost exceeds the unchanged residual. Every common message therefore keeps the same packing decision. For necessity, equal

Table 14: Loss concentration under random links at $d = .8$, against an independent equal-risk reference. Each cell gives observed / simulation median / simulation tail fraction. The reference assigns each eligible comparison the pooled loss probability and retains question weights. It does not preserve shared-request or history dependence. For gpt-oss social, only 5 questions have any loss, fewer than the eight top-10% slots. The observed share is therefore necessarily 100%; the count and top-question measures resolve this ceiling.

Reader	Workload	Questions with any loss	Top 10% share (%)	Top question share (%)
gpt-oss	Workplace	75 / 99 / < 0.001	48 / 38 / < 0.001	4 / 3 / 0.005
gpt-oss	Social	5 / 15 / < 0.001	100 / 71 / 0.020	79 / 15 / < 0.001
GPT-4.1 mini	Workplace	14 / 16 / 0.313	68 / 58 / 0.169	16 / 11 / 0.116
GPT-4.1 mini	Social	11 / 13 / 0.269	87 / 74 / 0.159	28 / 15 / 0.035

Table 15: Principal gpt-oss accuracy contrasts in percentage points. Link conditions average three draws and all conditions average four orders. Workplace ranges omit each history in turn; social 95% intervals resample histories. Order ranges show the smallest and largest contrast across the four cyclic orders.

Contrast	Workplace			Social		
	Estimate	Omit-one range	Order range	Estimate	95% interval	Order range
Correct index − unfiltered search	-0.63	[-3.12, +0.94]	[-4.00, +5.00]	+0.51	[-3.86, +6.61]	[-1.22, +2.44]
Random links − correct index	-1.75	[-2.71, -0.52]	[-3.33, +0.33]	-1.52	[-6.10, +1.32]	[-1.90, -1.22]
Look-alike links − correct index	-0.04	[-1.04, +1.20]	[-1.33, +1.17]	-1.46	[-5.42, +1.12]	[-2.30, -0.54]
Look-alike links − random links	+1.71	[+1.04, +2.71]	[+0.83, +2.00]	+0.07	[-1.32, +1.46]	[-0.54, +0.68]

Table 16: Complete hosted-reader answer transitions (%). All conditions are compared with the correct index. Gain, loss, valid wrong-to-wrong change, and invalid wrong change sum to churn before rounding. Net is gain minus loss in percentage points. All rows average four orders; links average three draws for gpt-oss and one for GPT-4.1 mini.

Reader	Workload	Condition	Churn	Gain	Loss	Wrong	Invalid	Net
gpt-oss	Workplace	Unfiltered search	27.88	10.75	10.13	7.00	0.00	+0.63
gpt-oss	Workplace	Look-alike links	20.04	7.46	7.50	5.08	0.00	-0.04
gpt-oss	Workplace	Random links	19.25	6.04	7.79	5.38	0.04	-1.75
gpt-oss	Social	Unfiltered search	5.18	2.34	2.85	0.00	0.00	-0.51
gpt-oss	Social	Look-alike links	4.17	1.08	2.54	0.54	0.00	-1.46
gpt-oss	Social	Random links	4.07	1.05	2.57	0.44	0.00	-1.52
GPT-4.1 mini	Workplace	Unfiltered search	15.94	5.94	4.69	5.31	0.00	+1.25
GPT-4.1 mini	Workplace	Look-alike links	8.75	2.81	3.44	2.50	0.00	-0.62
GPT-4.1 mini	Workplace	Random links	15.62	6.56	5.94	3.12	0.00	+0.62
GPT-4.1 mini	Social	Unfiltered search	10.01	3.00	4.07	2.95	0.00	-1.07
GPT-4.1 mini	Social	Look-alike links	5.95	1.22	3.20	1.52	0.00	-1.98
GPT-4.1 mini	Social	Random links	7.52	1.68	4.32	1.52	0.00	-2.64

selected sets cannot contain an exclusive message. Removing these skipped messages leaves every residual and common decision unchanged, reducing both executions to $P(C)$. Each removed message must thus exceed its common-set residual.

The earliest exclusive message with $\ell(h) \leq R_C(h)$ identifies the first unequal selection. Earlier exclusive messages are skipped, so this message's side follows the common execution until admitting it. The rule handles both additions and removals caused by the shortlist cap. Figure 5 gives a small example.

Under the study's fixed serialization, equal selected sets give identical context. In a system where distinct identifiers can serialize to the same text, byte equality requires a separate check. Set-dependent scores, listwise reranking, and generated summaries violate the fixed-order or fixed-cost assumptions and need a different analysis.

G Qwen results and option-order checks

Qwen's primary endpoint compares the correct index with unfiltered search in the released option order. Both social conditions score 55.5%, with a paired interval of $[-9.1, 8.1]$ points for their difference. The workplace difference is -1.5 points. Table 23 retains this negative endpoint and the equal-question sensitivity. Equal-history weighting gives a question from a one-question history more mass than a question from a larger history.

Table 17: Random-link comparisons at $d = .8$ with changed context, split by evidence-label membership. Pairs are unweighted counts. Churn, gain, and loss are conditional percentages under original history mass. The social percentages therefore differ from pooled pair fractions.

Reader	Workload	Labels	Pairs	Churn	Gain	Loss
gpt-oss	Workplace	unchanged	1396	22.78	7.02	9.24
gpt-oss	Workplace	changed	548	26.28	8.58	10.58
gpt-oss	Social	unchanged	708	2.90	1.50	0.77
gpt-oss	Social	changed	20	62.50	0.00	62.50
GPT-4.1 mini	Workplace	unchanged	188	14.89	5.32	4.79
GPT-4.1 mini	Workplace	changed	88	25.00	12.50	11.36
GPT-4.1 mini	Social	unchanged	244	7.91	2.33	4.31
GPT-4.1 mini	Social	changed	4	75.00	0.00	50.00

Table 18: Effect after omitting the Marcus history (percentage points). The remaining 40 social histories receive equal weight. This post hoc influence check uses a fixed-seed history bootstrap conditional on saved responses. Its intervals do not account for selecting Marcus after inspecting losses.

Reader	Links	Without Marcus	95% interval
gpt-oss	Look-alike links	+0.17	[-1.08, +1.32]
gpt-oss	Random links	+0.52	[-0.17, +1.60]
GPT-4.1 mini	Look-alike links	-2.03	[-4.22, -0.10]
GPT-4.1 mini	Random links	-1.46	[-3.96, +0.78]

Table 19: Random-link effect on question-order cells classified by control answers. Dependent means no memory is wrong and full evidence is right. Other means the matched full-evidence cohort outside that class. Unavailable means a full-evidence response is absent. Counts are distinct question-order cells, before draw expansion; mass retains original weights. Effects are percentage points.

Reader	Workload	Stratum	Cells	Mass	Effect
gpt-oss	Workplace	full evidence available	784	98.00	-1.91
gpt-oss	Workplace	full evidence unavailable	16	2.00	+6.25
gpt-oss	Workplace	memory dependent	313	39.13	-4.05
gpt-oss	Workplace	other matched cells	471	58.88	-0.50
gpt-oss	Social	full evidence available	320	100.00	-1.52
gpt-oss	Social	full evidence unavailable	0	0.00	–
gpt-oss	Social	memory dependent	214	64.63	-1.52
gpt-oss	Social	other matched cells	106	35.37	-1.53
GPT-4.1 mini	Workplace	full evidence available	316	98.75	+0.63
GPT-4.1 mini	Workplace	full evidence unavailable	4	1.25	+0.00
GPT-4.1 mini	Workplace	memory dependent	148	46.25	+4.73
GPT-4.1 mini	Workplace	other matched cells	168	52.50	-2.98
GPT-4.1 mini	Social	full evidence available	320	100.00	-2.64
GPT-4.1 mini	Social	full evidence unavailable	0	0.00	–
GPT-4.1 mini	Social	memory dependent	200	61.69	-3.38
GPT-4.1 mini	Social	other matched cells	120	38.31	-1.46

Four-order averaging changes the workplace contrast's sign (Table 24). Both columns use the first random draw at $d = .8$, unlike Qwen's primary three-draw analysis. Ordinary conditions use the full cohort; full-evidence comparisons use the common feasible subset (Appendix C).

The 32 workplace update questions ask how revised information changes a decision. Qwen answers none correctly in the released order under no memory, unfiltered search, the correct index, or any of the three $d = .8$ draws. Consequently, its conditional loss risk on these questions is undefined, since no correct-index answer can be lost. The floor changes under option reordering. Only 28 update questions fit the full-evidence control in every order. All update questions remain in the primary endpoint.

Among changed-context pairs with stable answers at $d = .8$, Qwen matches its no-memory answer on 86.2% of workplace and 29.7% of social comparison mass in the released order. Table 21 gives the separate four-order, one-draw check. Complete-quartet identity consistency requires four valid responses; pairwise disagreement

Table 20: Full-evidence minus comparator accuracy in percentage points. Each comparison retains identical questions in both conditions and all four option orders. Workplace ranges omit histories; social ranges are 95% history-bootstrap intervals. Full evidence uses all labeled messages outside the fixed memory budget, so it changes both evidence availability and context length.

Reader	Workload	Comparator	Questions	Δ	Range
gpt-oss	Workplace	Correct index	196	+20.90	[+18.87, +21.91]
gpt-oss	Workplace	Unfiltered search	196	+20.35	[+18.39, +22.16]
gpt-oss	Workplace	no memory	196	+34.48	[+33.88, +35.44]
gpt-oss	Social	Correct index	80	+16.06	[+6.40, +27.44]
gpt-oss	Social	Unfiltered search	80	+16.57	[+6.61, +28.05]
gpt-oss	Social	no memory	80	+63.82	[+55.28, +71.34]
GPT-4.1 mini	Workplace	Correct index	79	+29.00	[+25.70, +31.56]
GPT-4.1 mini	Workplace	Unfiltered search	79	+27.73	[+25.68, +30.08]
GPT-4.1 mini	Workplace	no memory	79	+44.56	[+40.47, +47.50]
GPT-4.1 mini	Social	Correct index	80	+21.24	[+8.13, +33.74]
GPT-4.1 mini	Social	Unfiltered search	80	+22.31	[+8.94, +35.37]
GPT-4.1 mini	Social	no memory	80	+58.94	[+47.46, +68.90]

Table 21: Conditional agreement with the same-order no-memory answer (%). Stable/context changed requires changed context and the same valid answer. New answer requires two different valid answers and compares the changed condition's answer with no memory. Rates retain original comparison weights. gpt-oss uses three draws; Qwen and GPT-4.1 mini use one. All rows average four orders. GPT-4.1 mini has 80 workplace questions; the other readers have 200.

Reader	Workload	Links	Stable/context changed	New answer
Qwen	Workplace	Random links	82.32	42.42
Qwen	Social	Random links	33.45	34.10
gpt-oss	Workplace	Random links	63.90	40.35
gpt-oss	Workplace	Look-alike links	64.60	38.25
gpt-oss	Social	Random links	29.74	5.83
gpt-oss	Social	Look-alike links	31.33	14.63
GPT-4.1 mini	Workplace	Random links	72.57	32.00
GPT-4.1 mini	Workplace	Look-alike links	75.00	60.71
GPT-4.1 mini	Social	Random links	30.24	32.43
GPT-4.1 mini	Social	Look-alike links	34.16	32.48

Table 22: Different answers to identical requests, as count/total and exact 95% binomial intervals (%). Requests were selected within fixed condition strata. Intervals assume a common independent Bernoulli process within each listed sample; selection limits their interpretation beyond those samples. The reproduction files retain all condition strata.

Reader	Workload	Changed / pairs	Exact 95% interval
Qwen	All	0/20	[0.00, 16.84]
gpt-oss	All	8/48	[7.48, 30.22]
gpt-oss	Workplace	3/24	[2.66, 32.36]
gpt-oss	Social	5/24	[7.13, 42.15]
GPT-4.1 mini	All	1/24	[0.11, 21.12]

uses only valid response pairs.

H Case studies

The Marcus trace expands Figure 1. Figure 6 shows the contrasting Camila case, where the direct supporting statement stays in the context.

H.1 Marcus and the rules before game night

The question asks who gave the more specific procedure for reading game rules before a session. Released options are Tomasz, Derek, Marcus, and Raj; Marcus is correct. Tomasz proposes choosing the game early so people can read the rules. Marcus adds a Wednesday deadline for the owner's rulebook link. His statement is the sole evidence-labeled message.

With the correct index, gpt-oss selects Marcus in all four option orders. Random links at $d = .8$ remove his labeled message from all 12 compared contexts. The reader selects Tomasz ten times

Table 23: Qwen correct-index minus unfiltered-search accuracy in the released option order. Differences are percentage points. The range uses workplace omit-one-history means or a social 95% paired history-bootstrap interval. Gain and loss columns are unweighted question counts.

Workload	Equal hist. pp	Question pp	Range	$0 \to 1$	$1 \to 0$
Workplace	-1.5	-1.50	[-3.1, +0.0]	5	8
Social	+0.0	+3.75	[-9.1, +8.1]	7	4

Table 24: Qwen option-order sensitivity in percentage points. The first column uses the released order; the second averages four cyclic orders. Random links use one fixed draw at $d = .8$ in both columns. Ranges use workplace omit-one-history means or social history-bootstrap intervals.

Contrast	$r = 0$ (pp)	Four orders (pp)
Workplace		
Correct index − Unfiltered search	-1.5 [-3.1, +0.0]	+1.2 [+0.8, +1.7]
Random links − Correct index	+3.5 [+1.9, +5.0]	-0.5 [-1.1, +0.0]
Social		
Correct index − Unfiltered search	+0.0 [-9.1, +8.1]	-3.9 [-9.4, +1.2]
Random links − Correct index	-2.6 [-9.3, +4.9]	+1.3 [-2.4, +5.7]

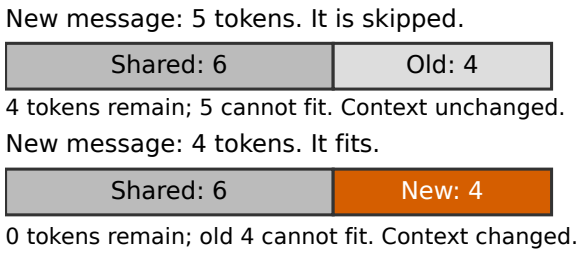


Figure 5: An exclusive message changes selected membership only if it fits. In this constructed example, the first shared message leaves four tokens. An exclusive five-token message is skipped. An exclusive four-token message fits and displaces the later shared message.

and Marcus twice. Draws one and three give four Tomasz answers each; draw two gives two Tomasz and two Marcus answers. No-memory answering is correct in one of four orders. Full evidence is correct in all four. Unfiltered search omits the labeled message and selects Tomasz in all four orders.

The first random draw has five false links. They connect Marcus with Derek, Priya with Tomasz, Priya with Derek, Raj with Felix, and Nadia with Soo-Yeon. The resulting context admits Derek's messages "agree. read the rules people" and "read it. lets go", plus Soo-Yeon's comment that Felix did well for a first game. Marcus's rulebook instruction and one Priya message leave. Figure 1 shows selected messages from this comparison. All displayed names and message texts stay correct.

In draw two, three Lena messages and one Kezia message enter. Marcus's instruction, two Priya messages, and one Tomasz message leave. In draw three, two Derek messages, one Kezia message, and one Soo-Yeon message enter, while Marcus's instruction and two Priya messages leave. The reproduction files retain all context messages, source identifiers, answers, and graphs for this trace.

The first look-alike draw retains Marcus's message and yields four correct answers. The other two omit it and yield eight Tomasz answers. Across random links, look-alike links, and unfiltered search, Tomasz is selected in 22 of the 24 comparisons where that message is absent. The question was selected after decomposing social losses. As its history's only evaluated question, it has weight $1/41$. Multiple messages change together, and hosted service variation remains possible, so the trace does not isolate one message's causal effect.

Qwen gives a contrasting result on the same question at $d = .2$ in the released order. Marcus's instruction remains in both contexts. Two turns enter and two leave, and Qwen changes from Tomasz to Marcus. This gain in key agreement occurs without recovering the labeled support.

H.2 Camila and support outside the evidence label

The Camila question asks who preferred good-enough observing conditions every night to perfect conditions a few times a year. The options are Bex,

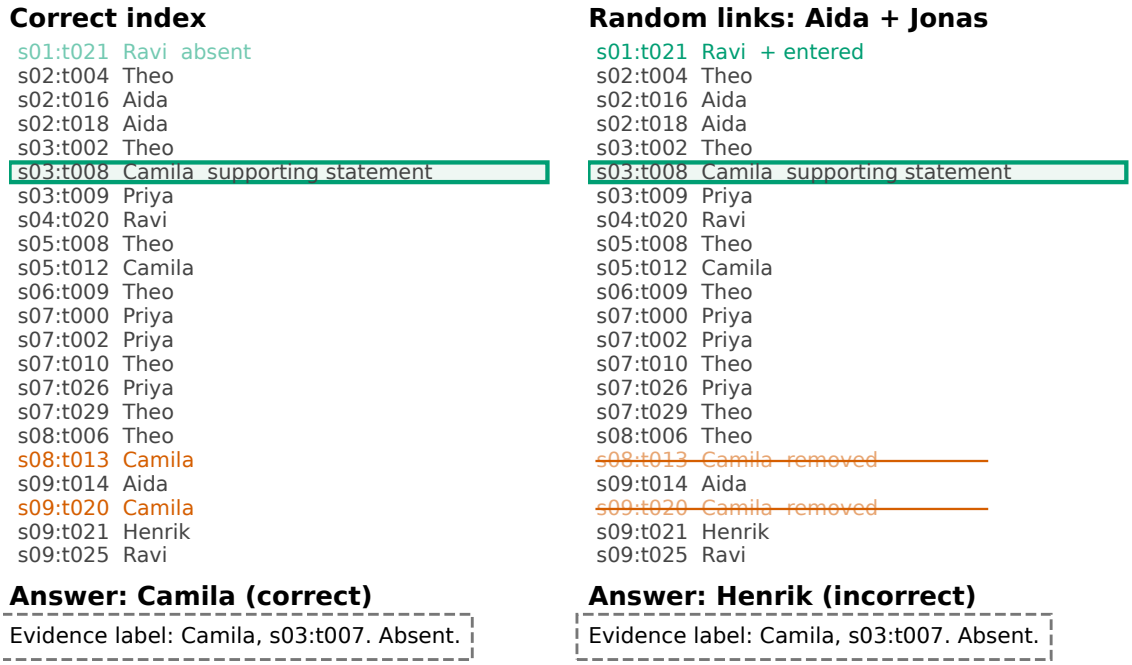


Figure 6: Qwen changes its answer although Camila's supporting statement remains. Jonas's messages reach the shortlist but do not fit the context. One Ravi message enters and two Camila messages leave. The benchmark's labeled turn is absent from both contexts. Each column lists the union of selected message identifiers, with absent messages marked.

Camila, Henrik, and Marcus. In the first random draw at $d = .2$, an Aida–Jonas link admits Jonas's messages. Two reach the shortlist, but neither enters the final context. The changed packing instead admits one Ravi message and removes two Camila messages.

Camila's correctly named statement remains in both contexts, but Qwen changes from Camila to Henrik. The benchmark labels an adjacent earlier Camila turn, which is absent in both contexts. Consequently, both contexts have zero labeled coverage even though the direct supporting statement remains. Figure 6 shows the complete union of selected message identifiers and marks absence on each side.

This case comes from a fixed pseudorandom selection of one gain, loss, stable-correct, and stable-wrong case per workload and contrast, for two contrasts. All 16 cases, contexts, and responses remain in the reproduction files. Selection supports outcome inspection, not frequency estimation.

I Limits of the evidence labels

Workplace questions have a median of 26 evidence labels; social questions have one. Workplace labels often include background, while labels can omit direct support, as for Camila. Coverage therefore has different meanings across workloads.

An AI assistant inspected 30 questions, one per eligible family and workplace history. Fixed pseudorandom selection followed retrieval analysis but preceded reader-answer inspection. Complete labeled spans, options, and keys were visible. Some questions depended on style; some labels omitted the requested detail. These judgments are neither human annotations nor an error-rate estimate. Records retain unresolved cases, including two update relations. Scoring retains official keys.

Full evidence exceeds the retrieval budget and can include background or superseded statements. It is a context control on the feasible cohort. Workplace profile questions can require history-wide patterns and are excluded from the benchmark's oracle evaluation (Hu et al., 2026). All conditions use complete histories because questions have no time cutoff.

Tracing Failures through Plant Topology: Propagation Reasoning by a Knowledge-Graph-Grounded LLM Agent

Muhamad Imannulhakim
PT. Pupuk Indonesia
m.imannul@pupuk-indonesia.com

Abstract

Diagnosing anomalies in process plants requires reasoning about how a failure at one unit propagates through interconnected equipment — knowledge that is scattered across engineering documents, and that unconstrained large language model (LLM) agents tend to fill with plausible but unsupported conclusions. We present an LLM agent, being piloted at an operating ammonia plant, whose domain memory is a knowledge graph of the plant's process: a topology backbone translated from the process flow, enriched with knowledge from over 30 engineering documents. We evaluate its propagation reasoning against the graph itself using 46 generated probes, each run three times. Entity grounding holds at every depth: all 565 failure-mode attributions are correct, and the agent rejects every question that attaches a failure mode to the wrong unit. Mean path recall falls from 1.00 at two hops to 0.60 at eight, and 40% of eight-hop answers deny a route the graph records. Scoping the same tools to the topology backbone, with model, prompt and graph unchanged, raises eight-hop recall to 0.81 and halves these denials, suggesting that unscoped neighbourhood queries struggle to reconstruct long process routes. Retrieval over the source documents matches the agent up to six hops but not at eight.

1 Introduction

A process plant is a tightly coupled network: a fault at one unit reaches others through shared process streams, heat integration and protective interlocks. Diagnosing an anomaly therefore means following propagation paths, using knowledge scattered across process diagrams, failure-mode libraries and maintenance records. Large language models (LLMs) can read and connect this material, but they also produce fluent claims their sources do not support (Ji et al., 2023), which is a serious risk when a wrong diagnosis leads to a wrong maintenance action. Grounding LLMs in knowledge graphs (KGs) is a widely studied mitigation (Agrawal et al., 2024; Pan et al., 2024), and tool-calling agents (Yao et al., 2023) can explore a KG step by step. KG–LLM combinations have been applied to industrial fault diagnosis (Ma et al., 2025), but a recent systematic review finds the field concentrated on single machines and short of evaluation metrics (Razaq et al., 2026).

Two problems make plants harder. Propagation is a property of *topology*: what matters is not only which failure modes a unit has, but how an effect travels across the plant. And deployed diagnostic systems rarely have field-verified labels, since confirming a hypothesised failure usually means opening the machine. We address both by encoding the plant's topology explicitly, which lets the graph serve as its own ground truth.

All results characterise the deployed architecture — the LLM together with its graph tools, prompt and output contract — rather than the LLM in isolation. The benchmark tests whether the agent can trace the route a failure would follow and report it faithfully. Our contributions are:

- **A KG design for process plants** that separates a topology backbone from a document-derived knowledge layer, so that an agent can reason about how failures propagate rather than only retrieve documents.
- **A graph-grounded benchmark** that decomposes propagation reasoning into six checks — path fidelity, direction integrity, intervention recall, inference-boundary placement, entity grounding and termination — verified against the graph in Cypher, with no LLM judge; direction and the protection controls are adjudicated by hand.
- **A diagnosis of the depth failure**: scoring entity grounding apart from path fidelity shows

Proceedings of the 3rd Workshop on KNOwledge GRaphs & Agentic Systems Interplay (NORA 2026), pages 131–137
November 10, 2026 ©2026 Association for Computational Linguistics

that the agent loses long routes without inventing entities, and an ablation that scopes traversal to the topology backbone recovers about half of the loss, pointing to the tools' traversal rather than the model. Against retrieval over the source documents and over the graph's nodes as text, the agent's advantage appears at depth and is largest with scoped traversal.

2 Related Work

LLMs reasoning over knowledge graphs. Grounding LLMs in KGs is a widely studied way to reduce unsupported output (Agrawal et al., 2024; Pan et al., 2024). Some methods give the model interfaces to structured data and let it read iteratively before reasoning (Jiang et al., 2023). Others treat the LLM as an agent that explores the graph step by step (Sun et al., 2024), building on interleaved reasoning and tool use (Yao et al., 2023), or have it generate relation paths as plans and retrieve matching paths as evidence (Luo et al., 2024). These methods are evaluated on open-domain KG question answering with gold answers. Our setting differs in three respects: the graph is a private, engineering-authored model of a single plant; the paths that matter follow process flow; and the diagnostic questions users ask have no gold answers. The concern we study applies to any KG that layers rich annotation over a structural backbone, although we test it on one graph only.

KG–LLM fault diagnosis in industry. Process-systems engineering has long modelled fault propagation as a causal graph over plant variables, such as signed directed graphs, and diagnosed faults by searching it (Venkatasubramanian et al., 2003; Palmer and Chung, 2000); our topology backbone plays a similar role, but an LLM agent traverses it. Ma et al. (2025) construct a fault-diagnosis KG to strengthen LLM reasoning for diagnosis and maintenance decision support. A systematic review of KG–LLM fault diagnosis finds the field concentrated on rotating machinery, steel and CNC equipment, with gaps in evaluation metrics and in explainable reasoning routes (Razaq et al., 2026). We address a process plant, where diagnosis depends on connections between units rather than on a single machine, and we evaluate the agent specifically on those connections.

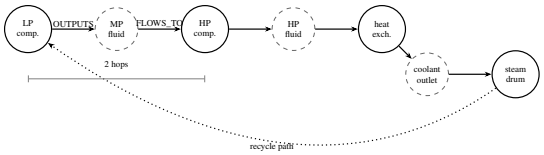


Figure 1: The topology backbone, shown schematically with generic equipment names. LP, MP and HP denote the low-, medium- and high-pressure stages.

3 Methodology

3.1 Knowledge Graph

The graph has a *topology backbone* and *knowledge layers*. In the backbone, derived from piping and instrumentation diagrams, process flow diagrams and process manuals, equipment items and process streams are both nodes, linked as `Equipment` $\xrightarrow{\text{OUTPUTS}}$ `ProcessStream` $\xrightarrow{\text{FLOWS_TO}}$ `Equipment`. Modelling streams as nodes gives an exchanger distinct outlets that lead to different parts of the plant, and because every link is directed, recycle and heat-integration loops are explicit rather than ambiguous (Figure 1). A route between two units is an alternating path of equipment and streams, so a failure's reach is measured in hops. Knowledge layers derived from over 30 engineering documents attach failure modes and causes, protective interlocks, instrumentation, maintenance tasks and past investigations to the backbone (Table 1). The graph holds 1,219 nodes and 1,962 relationships in Neo4j. The graph and documents are proprietary and cannot be released.

3.2 Agent

The agent is a tool-calling LLM (DeepSeek V4 Flash) (Yao et al., 2023) that reaches the graph only through read-only tools served over the Model Context Protocol (MCP): resolution and failure-chain tools, a Cypher tool, and two neighbourhood tools for propagation. `explore` returns everything within k hops of one node (k=2 by default) and `propagation_rca` (root cause analysis) does the same around a target (k=3) with summaries of its failure data and protective wiring. Both follow every relationship in either direction, up to k=8 and 150 returned nodes, so a neighbourhood spans the knowledge layers and grows far faster than the topology (Table 2). When a neighbourhood exceeds the cap, the prompt tells the agent to retry with a smaller k rather than reason from a partial result. Raising the cap is not a real op-

Layer	Node types (count)	Key relationships
Topology backbone	Equipment (158), ProcessStream (169), Stage (16)	OUTPUTS, FLOWS_TO, HAS_COMPONENT
Failure	FailMode (76), Cause (148), Event (3), Concept (13)	HAS_FAILMODE, HAS_CAUSE, CAN_LEAD_TO
Protection	Trigger (30), Interlock (8), Action (18)	INDICATES, TRIGGERS, ACTIVATES
Instrumentation	MeasurementPoint (141), Tag (188)	MONITORS, INSTRUMENTED_AS
Maintenance & procedures	MaintenanceTask (23), MonitoringTask (14), ProcedureStep (87)	MITIGATED_BY, VERIFIES, NEXT
Investigation history	RCADocument (5), AuditReport (1), Recommendation (57)	IDENTIFIES, RECOMMENDS, ADDRESSES

Table 1: Knowledge-graph layers and node counts. Names are as stored in the graph; the evaluation relies on HAS_FAILMODE (a unit's recorded failure modes), CAN_LEAD_TO (one failure mode causing another) and the protection chain, in which a Trigger indicates a failure mode and trips an Interlock that activates an Action. RCA: root cause analysis.

k	All relationships	Process flow only
2	27 / 71	4 / 5
3	118 / 346	7 / 9
4	290 / 685	9 / 12
6	750 / 1,088	15 / 23
8	1,095 / 1,156	25 / 40

Table 2: Nodes reached within k hops of the ten eight-hop source units (mean / max), when the walk follows every relationship in either direction or only downstream process flow (OUTPUTS, FLOWS_TO).

tion in deployment, since an unscoped eight-hop neighbourhood holds most of the graph, far more than can be passed to the model in one call. We therefore evaluate two configurations of the agent: *deployed*, with the tools as described, and *scoped*, which adds one option to both tools that restricts the walk to process flow, downstream or upstream along OUTPUTS/FLOWS_TO, described only in the tool descriptions; model, prompt, graph and defaults are unchanged. The scoped configuration is analysed as an ablation (§4.3).

For "if a failure occurs at A, what happens to B?", the prompt has the agent walk the topology from A to B and end every factual sentence with a typed citation: *graph* for a node it relies on, or *inference* for its own reasoning, also listed in an unverified field. Where the graph has no data or no route, it must say so rather than fill the gap.

3.3 Evaluation

Because the topology is explicit, routes, hop distances, direction and target failure data are all queryable, so answers can be checked against the graph without field labels. We generated 46 probes and ran each three times, each in a fresh conversation: 40 propagation probes, ten at each of two, four, six and eight hops, mixed on whether the target has failure data; and six direction probes asked in reverse, three with no route that way and three on recycle loops, each also attaching a failure mode to a unit that does not carry it. Answers are scored on the metrics in Table 3 in Cypher, with no LLM judge; direction probes are adjudicated by hand. For P3, we use 15 separate protection probes, run once, that name a failure mode and its unit, with no target, and ask which trip, interlock or action would stop it: nine on failure modes wired to an interlock node, and six on unprotected failure modes.

Baselines. We compare the agent with two retrieval-augmented generation (RAG) baselines that answer in a single call without tools, using the same model and a prompt that asks for the same output format. *Document RAG* uses no graph at all: 13 of the source documents the graph was built from are cut into 250-word chunks with a 50-word overlap, and the ten chunks that best match the question under BM25 form the context, about 2,500 words. The remaining sources are mostly diagrams, such as fault trees and logic diagrams, whose content is not ideal for text retrieval. The second RAG (*Node RAG*) keeps the graph's content but not its structure: every node, across all layers, is indexed as a short text document (its name, type and relationships), and the 20 best BM25 matches form the context. Each baseline runs three times on all propagation probes.

4 Results and Discussion

4.1 Comparison with RAG Baselines

Figure 2 compares path recall across the agent and both baselines. Process streams are named only in the graph, so a system answering from documents cannot name them; this comparison therefore scores equipment only, counting an item as named when the answer gives its tag or name, and leaves out two-hop routes, which have no intermediate equipment. These values are therefore not comparable with the full P1 in §4.2.

Up to six hops, document RAG matches the de-

	Metric	Gold
P1	**Path fidelity**: recall of intermediate nodes, for the shortest path and for the best-matching valid route of at most eight hops; precision: the share of equipment and streams an answer names, other than the endpoints and their immediate neighbours, that lie on a valid route; and their F1	`shortestPath` and full route enumeration over `OUTPUTS/FLOWS_TO`
P2	**Direction integrity** on reverse-pair probes	One-way vs. recycle pairs; human-adjudicated
P3	**Intervention recall**: the trip, interlock or action that would interrupt propagation; on unprotected controls, no safeguard may be claimed	`Trigger`→`Interlock`→`Action` wiring; controls human-adjudicated
P4	**Inference boundary**, two-sided. *Recall*: if the target has no failure data, consequences there must be declared as inference. *Precision*: if it has failure data, at least one recorded failure mode must be asserted as graph fact	`HAS_FAILMODE` at target
P5	**Entity grounding**: citation keys declared graph-backed that resolve to no node; and failure-mode attributions ("equipment E has failure mode F") that the graph does not record	Node resolution; `HAS_FAILMODE`
P6	**Termination**: cited equipment and streams stay within the source–target span	Path membership
—	**False absence claim**: the answer states that no route exists when the graph records one. Not automatic; counted by reading answers	`shortestPath` over `OUTPUTS/FLOWS_TO`

Table 3: Evaluation metrics. P1, P4 and P5 are scored automatically in Cypher; P2 and the P3 controls are human-adjudicated; P6 raises flags for review.

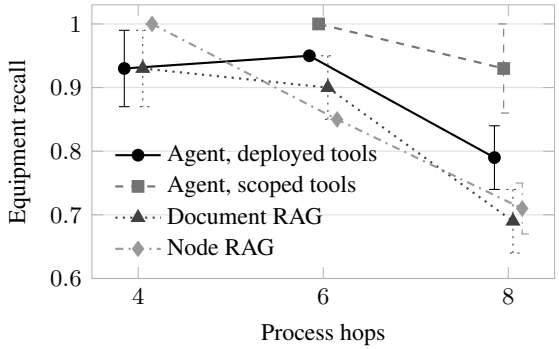


Figure 2: Path recall on equipment only, by hop distance, mean ± s.d. over three runs (n=10 probes per band).

ployed agent (0.93 and 0.90, against 0.93 and 0.95). At eight hops it falls to 0.69 against 0.79, while the scoped agent (§3.2) reaches 0.93. Despite the high recall, document RAG denies the route in 6, 12 and 16 of 30 answers at four, six and eight hops, against 6, 7 and 12 for the deployed agent (Table 5). Node RAG traces the route in all two-hop answers and 27 of 30 four-hop answers but in none of the 60 six- and eight-hop answers, so its recall there (0.85 and 0.71) comes entirely from units named while denying the route. The graph's advantage is thus clearest at depth with scoped traversal.

4.2 Agent Results

These results describe the deployed agent with the tools of §3.2. Figure 3 shows path recall, precision and F1 by hop distance as the mean ± standard deviation of the three per-run band means.

Check	Run 1	Run 2	Run 3
No failure data at target → inference	16/17	17/17	17/17
Failure data at target → stated as fact	23/23	23/23	21/23
Boundary correct (both)	**39/40**	**40/40**	**38/40**
Answer hedges something	39/40	39/40	40/40

Table 4: Inference boundary (P4) on the 40 propagation probes.

Path fidelity (P1). Recall against any valid route stays within 0.03 of recall against the shortest path in every band. Precision stays between 0.75 and 0.88 in every band, so the loss at depth is missed route, not padding. It is lowest at two hops, where the route has a single intermediate stream, so every extra unit an answer mentions as context counts against it. Band means are stable across runs (s.d. ≤ 0.08), but in terms of stability, a probe's recall is identical in all three runs for only 21 of the 40 propagation probes.

Inference boundary (P4). The boundary is placed correctly in nearly every answer (Table 4). Almost every answer marks some claim as inference, so passing the first row alone would prove little. The second row rules this out: when the graph records the target's failure modes, the agent states them as fact.

Entity grounding (P5). Each direction probe attaches a failure mode to a unit that does not carry it, and all 18 answers (six probes, three runs) re-

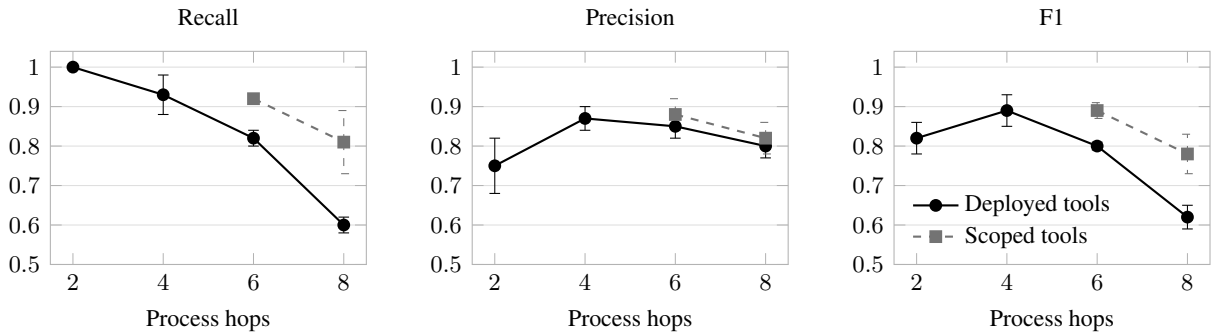


Figure 3: Path fidelity (P1) by hop distance, mean ± s.d. over three runs per condition (n=10 probes per band).

Hops	Run 1	Run 2	Run 3	Scoped (3 runs)
2	0/10	0/10	0/10	—
4	2/10	2/10	2/10	—
6	3/10	3/10	1/10	3/30
8	5/10	4/10	3/10	6/30
All	**10/40**	**9/40**	**6/40**	**9/60**

Table 5: Answers that deny a route the graph records. Runs 1–3 use the deployed tools; the scoped column pools three runs of the ablation (§4.3).

ject this false premise and name the correct owner. Every sentence tying a failure mode to a unit was matched against `HAS_FAILMODE`, and all 565 such attributions(across 198 answers: three deployed runs of all 46 probes and three scoped runs of the 20 six- and eight-hop probes) are correct. All 561 graph-backed citation keys also resolve to a node in the graph

False absence claims. We read every propagation answer and counted those that deny a process route the graph records; saying only that no *failure* link (`CAN_LEAD_TO`) exists is true and not counted. Denials grow with distance (Table 5) and recur on the same probes across runs. Most denials fall on routes that pass through a single node joining separate parts of a unit, such as a furnace's process and convection sections. Document RAG (§4.1) denies the same routes, so some denials may reflect how the graph models these units rather than an agent error.

Direction (P2), intervention (P3) and termination (P6). All 18 direction answers (six probes, three runs) are correct: the agent denies a route on the three one-way pairs and traces the loop on the three recycle pairs in every run. On the protection probes, the agent named the wired interlock, its trigger and its actions in all nine protected cases, and for the six unprotected failure modes stated that no protection is recorded rather than offering a nearby one. One control answer links the failure to protection on a related failure mode without marking the link as inference. P6 flags one eight-hop probe in the first run for citing a node outside the source–target span.

4.3 Why Long Routes Are Lost

Grounding holds, paths do not. From two to eight hops, path recall falls by 40 points while failure-mode attributions stay correct. The agent does not invent equipment as it loses the route; it makes false claims *about* the graph, such as denying a route that exists.

The layers bury the route. Because the tools follow every relationship, a neighbourhood floods into the knowledge layers (Table 2), and for eight of the ten eight-hop pairs even the smallest one that holds the whole route exceeds the 150-node cap. The agent is therefore told when a result is partial, but the prescribed recovery, a smaller k, moves a distant target further out of reach.

Ablation. With the scoped configuration (§3.2), over three reruns of the six- and eight-hop probes (Figure 3), eight-hop recall rose from 0.60 to 0.81 ± 0.08, with every scoped run (0.73–0.89) above every deployed run (0.57–0.61), and six-hop recall from 0.82 to 0.92. The gap between six and eight hops halves, from 0.22 to 0.11. Precision holds (0.80 to 0.82 at eight hops), so eight-hop F1 rises from 0.62 to 0.78 without answers padding the route. False absence claims at eight hops fall from 12 of 30 answers to 6 of 30 (Table 5).

Recycle loops. In six of the ten eight-hop pairs, a heat-integration or recycle loop also links the target back to the source, and these pairs lose the most recall (Table 6). On one-way pairs alone, recall

Hops	Pair connected	Deployed	Scoped
8	both ways (n=6)	0.51	0.75
8	one way (n=4)	0.74	0.90
6	both ways (n=5)	0.80	0.85
6	one way (n=5)	0.84	0.99
4	both ways (n=4)	0.86	—
4	one way (n=6)	0.98	—
2	both ways (n=5)	1.00	—
2	one way (n=5)	1.00	—

Table 6: Path recall for pairs that a recycle loop also connects in reverse versus one-way pairs, pooled over three runs per condition. The scoped ablation covers six and eight hops only.

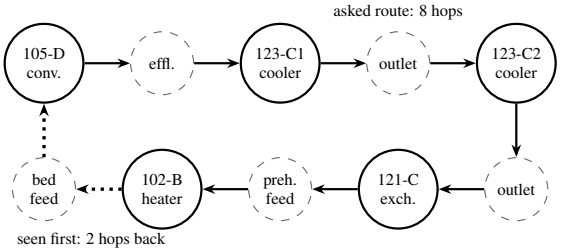


Figure 4: P-047, where a failure at converter 105-D is asked to reach start-up heater 102-B. The route runs eight hops through the effluent coolers and feed/effluent exchanger (solid), but 102-B also feeds 105-D directly (dotted). The agent found the short reverse edge, concluded that 102-B lies upstream, and reported no route.

drops from 1.00 at two hops to 0.74 at eight. Loops make it worse, pulling loop pairs down to 0.51. A typical failing answer finds the short link from target back to source first, decides that the target is upstream, and stops searching (Figure 4). Scoping improves both groups but does not close the gap.

5 Conclusion

We presented an LLM agent for an operating ammonia plant whose domain memory is a knowledge graph built around process topology, with a benchmark that checks propagation reasoning against the graph itself, without field labels. The agent's claims about entities hold: every failure-mode attribution is correct, and it rejects questions that place a failure mode on the wrong unit. Its claims about routes do not: recall falls from 1.00 at two hops to 0.60 at eight, and 40% of eight-hop answers deny a route the graph records. Because grounding and path fidelity are scored separately, this failure is visible even though no entity is invented. Restricting the same tools to the topology backbone recovers about half of the lost recall and halves the false denials, which points to the tools' traversal rather than the model; recycle loops remain the main source of error. Retrieval over the source documents matches the agent up to six hops, so the graph's advantage lies in long routes with scoped traversal and in answers that can be checked against it. Two practical lessons follow: model process topology explicitly so that propagation reasoning can be checked against the graph, and let traversal tools stay on that backbone instead of walking every layer at once.

Limitations

We measure consistency with the graph, not diagnostic correctness: a graph-consistent answer can still be wrong where the graph is. The baseline comparison scores equipment only, and grounding (P5) and the inference boundary (P4) apply to graph-backed answers alone. Results cover one plant, one graph and one model with limited probes and no confidence intervals. Probes are generated from the graph rather than collected from users. For a failure mode the graph does not record, the agent falls back on the model's general knowledge and marks it as inference, and such answers carry no graph guarantee.

Declaration on Generative AI

The system evaluated in this paper is itself an LLM agent (§3.2). Generative AI assistants were also used in building the knowledge graph, where they helped the author compare the content already ingested into Neo4j with the parts of the process topology not yet added. They were further used in preparing the paper: drafting and editing text, checking bibliographic details, writing the probe generation and scoring scripts, and reviewing agent responses. All numbers, adjudications and claims were checked by the authors, who take full responsibility for the graph and the paper.

References

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Can knowledge graphs reduce hallucinations in LLMs? : A survey. 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 3947–3960, Mexico City, Mexico.

Z. Ji, N. Lee, R. M. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. 2023. Survey of

hallucination in natural language generation. *ACM Computing Surveys*, 55(12).

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations (ICLR)*.

Y. Ma, S. Zheng, Z. Yang, H. Pan, and J. Hong. 2025. A knowledge-graph enhanced large language model-based fault diagnostic reasoning and maintenance decision support pipeline towards industry 5.0. *International Journal of Production Research*.

Claire Palmer and Paul W. H. Chung. 2000. Creating signed directed graph models for process plants. *Industrial & Engineering Chemistry Research*, 39(7):2548–2558.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Wan Barmawi Muhammad Razaq, Hao Chen, Marcos R. Machado, and João Luiz Rebelo Moreira. 2026. How can the integration of AI large language models and knowledge graph enhance fault diagnosis? A systematic literature review. *Applied Soft Computing*, 194:114908.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations (ICLR)*.

V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri. 2003. A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. *Computers & Chemical Engineering*, 27(3):313–326.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Author Index

Anish, Preethu Rose, 53
Avanesyan, Alina, 98

Biswas, Momojit, 53

Das, Bidit, 39
Diao, Yishi, 1

Goyal, Dev D., 109

Hwang, Tao, 1

Imannulhakim, Muhamad, 131

Khodorchenko, Maria, 98
Kundu, Boris, 26

Li, Baichuan, 15
Litvintsev, Alexander, 98

Maria, Sanjana Aktar, 84
Mehta, Pooja, 26
Mondal, Md Nazrul Islam, 63

Nimi, Sumaiya Tabassum, 84

Reza, Khan Raiyan Ibne, 84
Roshchupkin, Bogdan, 98

Shuvo, Md. Aukerul Moin, 63

Yang, Heyoung, 77
Yao, Junyi, 15
Yoo, Sung Tae, 77

Zheng, Zihao, 15
Zhukov, Andrei, 98