

Span-Based Reasoning with Anonymous Knowledge Graphs

Daniel Guzman-Olivares

Bulil Technologies
Autonomous University of Madrid
daniel.guzmano@estudiante.uam.es

Lara Quijano-Sanchez

Autonomous University of Madrid
lara.quijano@uam.es

Federico Liberatore

Cardiff University
liberatoref@cardiff.ac.uk

Abstract

Modern KGQA pipelines address complex multi-hop reasoning by shifting retrieval from classical embedding-based methods to iterative LLM calls. Although effective, this incurs significant computational and economic overhead that limits their applicability in real-world settings. In this paper, we introduce GRASE (Graph Reasoning with Anonymous Span Encoders), a framework that replaces LLM-driven search with a novel encoder-based graph-span representations over uncertain nodes and relations in a knowledge graph. Across several multi-hop KGQA benchmark datasets, GRASE achieves up to 85% faster inference than LLM-native pipelines while matching or exceeding state-of-the-art performance, demonstrating that encoder-only hybrid reasoning is a robust, practical, and scalable alternative to fully LLM-driven retrieval.

1 Introduction

Despite strong performance across many tasks, large language models (LLMs) still struggle with queries requiring knowledge beyond their pretraining corpus. Modern systems address this using *Retrieval-Augmented Generation* (RAG) (Lewis et al., 2020), which equips LLMs with retrieval mechanisms that surface evidence at inference time (Wei et al., 2022; Yao et al., 2023a; Besta et al., 2024; Jeong et al., 2024). However, queries requiring complex reasoning or deep long-tail knowledge remain challenging (Petroni et al., 2021; Malek et al., 2025; Tan et al., 2025b).

Structural signals from *knowledge graphs* (KGs) have been integrated into knowledge-graph question answering (KGQA), where answers are grounded in graph topology within the RAG framework (Sun et al., 2023; Li et al., 2023b; Agrawal

et al., 2024). Early embedding-based methods (Sen et al., 2021; Saxena et al., 2020) were efficient but struggled with multi-hop queries, while GNN-based approaches (Jiang et al., 2023; Mavromatis and Karypis, 2022) improved reasoning but are expensive at realistic KG scales.

Recent KGQA systems instead perform iterative reasoning using the LLM as an agent to guide graph exploration (Sun et al., 2023; Tan et al., 2025a; Luo et al., 2024; Walter and Bast, 2025). While effective, repeated LLM calls at each traversal step make inference impractical at scale and quickly prohibitive in cost (Oche et al., 2025; Chen et al., 2023).

To address these issues, we propose *GRASE*—a graph reasoning framework based on an *anonymous spans encoder*. GRASE removes LLMs from the exploration loop by replacing them with encoders that perform *fuzzy* span matching between a query-derived anonymous graph pattern and candidate subgraphs in the KG, reducing LLM calls, latency, and cost without requiring prior knowledge of all possible relations.

We evaluate GRASE on multiple public QA benchmarks and show that it matches or outperforms state-of-the-art KGQA models while drastically reducing execution time, with ablation studies quantifying the contribution of each component.

2 Related Work

LLM-based question answering is highly susceptible to hallucination in complex reasoning tasks (Sadat et al., 2023; Huang et al., 2025; Jiang et al., 2024). Chain-of-Thought prompting (Wei et al., 2022) and its variants (Zhang et al., 2022b; Fu et al., 2023; Kojima et al., 2022; Yao et al., 2023a; Besta et al., 2024) reduce hallucination but remain

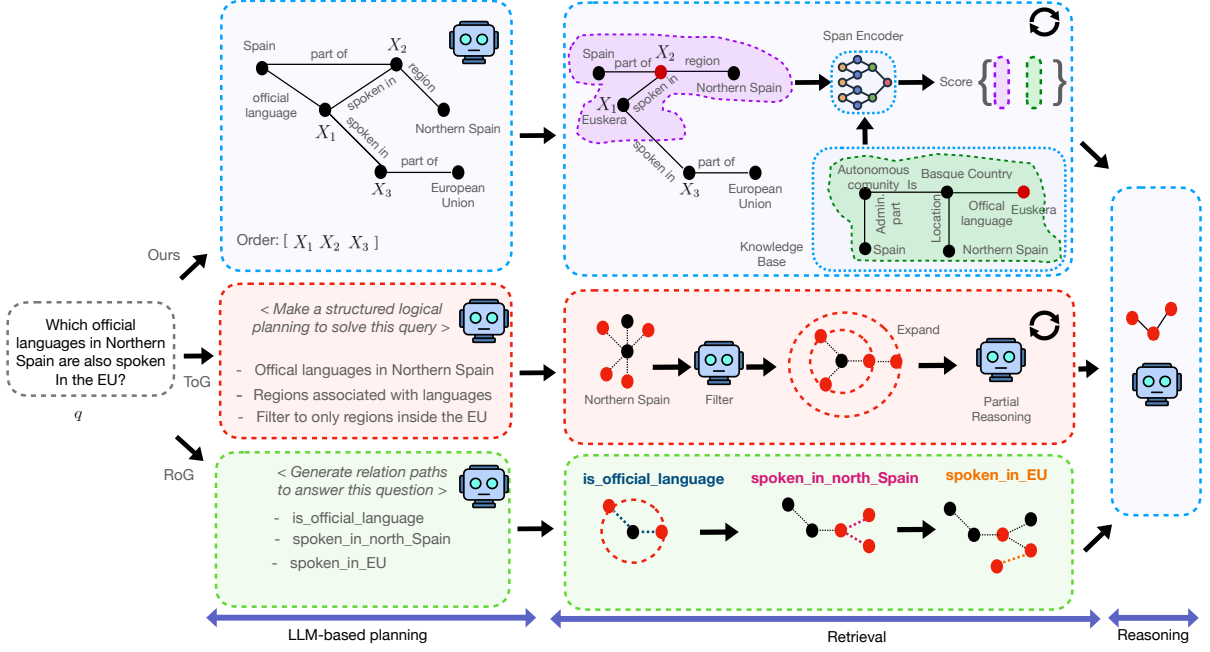


Figure 1: Comparison between GRASE (top), Think-on-Graphs (middle), and Reasoning-on-Graphs (bottom). ToG delegates reasoning entirely to the LLM; RoG maps LLM-planned paths one-to-one onto KG relations. GRASE instead constructs an AKG and explores the KG via scored spans, grounding reasoning in the graph while the LLM provides structured guidance only at planning and answer selection.

constrained by training data, motivating external knowledge integration (Luo et al., 2024; Li et al., 2023b; Yao et al., 2023b).

Knowledge graphs provide a structured external interface that alleviates data limitations and hallucination (Pan et al., 2024; Agrawal et al., 2024). Early KG-based RAG relied on embedding-based retrieval (Sen et al., 2021; Saxena et al., 2020; Shi et al., 2021) but failed on multi-hop reasoning. Methods encoding KG structure into transformers (Zhang et al., 2022a; Peters et al., 2019; Li et al., 2023b) reduce adaptability by coupling to the training-time KG (Wen et al., 2024; Hu et al., 2024). GNN-based methods (Jiang et al., 2023; He et al., 2021a) achieve competitive performance while more recent systems integrate KG retrieval into the LLM pipeline for scalability (Ma et al., 2024; Jin et al., 2024).

Modern KGQA systems follow a planning–retrieval–reasoning pipeline (see Figure 2). LLM-guided methods such as ToG (Sun et al., 2023), ToG-2 (Ma et al., 2024), PoG (Tan et al., 2025a), and EPERM (Long et al., 2025) improve on complex tasks but incur high cost from repeated LLM calls (Chen et al., 2023; Su et al., 2025). Efficiency-oriented variants prune the graph dynamically (Baek et al., 2023; Kim et al., 2023; Su et al.,

2024) but overlook global structure, degrading performance on long-range dependencies (Tan et al., 2025a). GNN-RAG methods (Mavromatis and Karypis, 2024; He et al., 2024) integrate graph neural networks into retrieval but remain expensive at scale.

Closest to our work, RoG (Luo et al., 2024) estimates path relevance via constrained beam search but assumes prior knowledge of all relations and depends on exact LLM-planned path matches. Sub-GraphRAG (Li et al., 2025) retrieves top- K locally scored triples leaving chain composition to the LLM, making performance sensitive to K and increasing hallucination risk. PathMind (Liu et al., 2025) follows a retrieve–prioritize–reason paradigm, scoring paths with a semantic-aware priority function and aligning the LLM with path-wise preferences. While PathMind reduces LLM calls compared to agentic methods, its reasoning component requires task-specific fine-tuning, which may limit generalization to unseen relations or tasks beyond training corpus; path scoring also remains independent of global query structure. In contrast, GRASE decouples retrieval entirely from the LLM via a frozen encoder, preserves global query structure through the AKG, and generalizes to unseen relations without fine-tuning.

To the best of our knowledge, within LLM-augmented KGQA pipelines, GRASE is the first to replace iterative LLM-based graph scoring with a frozen subgraph span encoder operating over anonymous query patterns, preserving global query structure without repeated LLM calls during exploration.

3 Methodology

GRASE replaces the LLM-guided graph retrieval loop with fuzzy span matching between the KG and a query-derived Anonymous Knowledge Graph (AKG), iteratively disambiguating anonymous variables via a frozen span encoder until a single final LLM call generates the answer. The framework follows the standard planning–exploration–reasoning pipeline.

3.1 Formal Setting of KGQA

Let $\mathcal{G} := (\mathcal{E}, \mathcal{R}, \mathcal{T})$ denote a knowledge graph, where $\mathcal{E}$ is the set of entities (nodes), $\mathcal{R}$ the set of relations (edges), and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ the set of triplets (e, r, e') with $e, e' \in \mathcal{E}$ and $r \in \mathcal{R}$. A (directed) span in $\mathcal{G}$ is a subgraph defined by the triplets sequence:

$$P := \{(e_i, r_i, e'_i)\}_i \in \mathcal{P}(\mathcal{G}),$$

where $\mathcal{P}(\mathcal{G})$ denotes the set of all valid spans (i.e. connected subgraphs).

Formally, a KGQA system maps a natural-language query q and a graph $\mathcal{G}$ to an answer subgraph $\mathcal{A}_q \in \mathcal{P}(\mathcal{G})$, i.e. the minimal span satisfying q :

$$\begin{aligned} f : \mathcal{L} \times \mathcal{G} &\longrightarrow \mathcal{P}(\mathcal{G}) \\ (q, \mathcal{G}) &\longmapsto \mathcal{A}_q \models q, \end{aligned}$$

where $\mathcal{L}$ is a language and $\mathcal{A}_q \models q$ denotes that the answer subgraph logically satisfies the query.

Modern LLM-based systems frequently decompose q into a sequence of sub-queries $(q_0, \dots, q_N)$, solving each step conditioned on prior results:

$$f_k = \begin{cases} f(q_0, \mathcal{G}; \emptyset) = \mathcal{A}_0, \\ f(q_k, \mathcal{G}; f_{k-1}) = \mathcal{A}_k, & k \geq 1, \end{cases}$$

where f_{k-1} is the context accumulated across calls, and $f_k = \mathcal{A}_q$ at termination.

3.2 Planning

Using an LLM to derive a resolution plan is simple and flexible. However, decomposing the query fragments the original structure and requires repeated

context updates across calls to maintain coherence throughout exploration. This increases inference cost and introduces additional opportunities for hallucination.

Instead, we reformulate the query as an *Anonymous Knowledge Graph (AKG)*, which preserves global structure while allowing unknown entities and relations to remain as variables to be resolved during exploration (see Figure 2). For example, the query “*Director of a movie in which Tom Hanks acted*” is mapped to

$$\text{Tom Hanks} \xrightarrow{\text{acted_in}} X_1 \xrightarrow{\text{directed_by}} X_2,$$

where X_1 denotes the unknown movie and X_2 the desired answer.

For a given query q , the AKG is defined as $\hat{\mathcal{G}}_q = (E_q, R_q, T_q)$, with

$$E_q := E \cup \hat{E}, \quad R_q := R \cup \hat{R}, \quad T_q := T \cup \hat{T}_1 \cup \hat{T}_2,$$

where $E \subseteq \mathcal{E}$ are the explicit entities and $\hat{E}$ the anonymous entity variables; $R \subseteq \mathcal{R}$ are the known relations and $\hat{R}$ the anonymous relation variables (i.e. relations whose type is unknown and must be resolved); and $\hat{T}_1, \hat{T}_2$ collect the triplets containing one or two anonymous variables, respectively.

We write $\hat{\mathcal{G}}_k$ for the AKG at iteration k , dropping the q subscript since the query is fixed throughout. Starting from the initial AKG $\hat{\mathcal{G}}_0$, we iteratively substitute each anonymous variable at iteration k , $X_k \in \hat{E} \cup \hat{R}$, with its highest-scoring candidate:

$$\hat{\mathcal{G}}_k = \begin{cases} \hat{\mathcal{G}}_0, & k = 0, \\ \text{resolve}(\hat{\mathcal{G}}_{k-1}, X_k \mapsto x^*), & k \geq 1, \end{cases}$$

$$x^* = \arg \max_{x \in \mathcal{E} \cup \mathcal{R}} \text{score}(X_k, x \mid \hat{\mathcal{G}}_{k-1}, \mathcal{G}) \quad (1)$$

until all anonymous variables are instantiated. Here, $\text{score}(\cdot)$ measures the affinity between the current AKG state $\hat{\mathcal{G}}_{k-1}$ and a candidate substitution $x \in \mathcal{E} \cup \mathcal{R}$ for X_k , grounded as a matching span in $\mathcal{G}$. The initial AKG $\hat{\mathcal{G}}_0$ is obtained via few-shot prompting of GPT-4 (Section A.1), following ToG (Sun et al., 2023) for entity linking. A per-dataset evaluation of AKG correctness is reported in Table 8.

3.3 Exploration

Exploration is framed as a *span matching* problem between the AKG at iteration k , $\hat{\mathcal{G}}_k$, and the knowledge graph $\mathcal{G}$. At each iteration, we resolve one

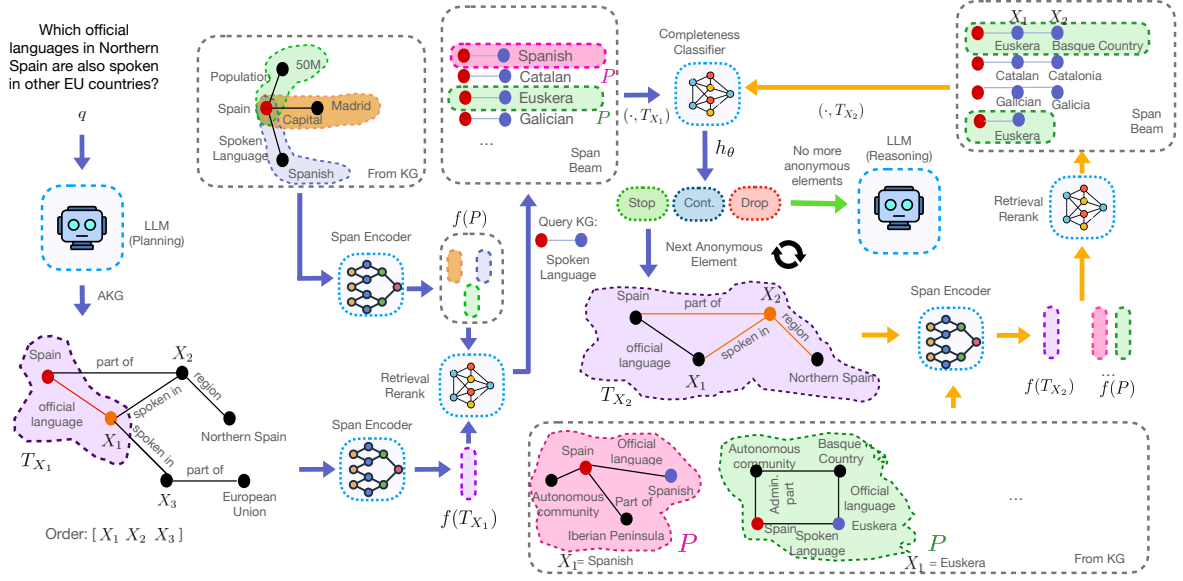


Figure 2: **GRASE pipeline.** A query is converted into an Anonymous Knowledge Graph (AKG) with variables X_1, X_2, X_3 . For each X_k , the frozen span encoder produces embeddings $f(T_{X_k})$ and $f(P)$ driving a retrieve-rerank pipeline over candidate spans P . The completeness classifier h_θ decides whether to expand (CONTINUE), accept (STOP), or discard (DROP) each span. Once all variables are resolved, the LLM selects the final answer.

anonymous variable $X_k \in \hat{E} \cup \hat{R}$ by aligning the set of AKG triplets that involve it:

$$T_{X_k} := \{(e, r, e') \in \hat{T}_1 : X_k \in \{e, e'\}\}$$

Alignment is *fuzzy*: a relation in the AKG may match a semantically or structurally similar span in $\mathcal{G}$, rather than requiring exact symbolic equality. To enable this, we introduce a span encoder and a retrieve-rerank pipeline, with early-stopping detection via a completeness classifier described below.

Span encoder. We represent each candidate span as a structured token sequence using five special tokens that mark span, anonymous, entity, triplet, and relation boundaries. Each triplet (e, r, e') is linearized with these tokens, and the embedding of the global span token, denoted $f(\cdot)$, serves as the span-level representation produced by the encoder. The span encoder is a DeBERTa-v3-Large (He et al., 2021b) model fine-tuned with a triplet loss:

$$\mathcal{L} = \max(d(f(T_X), f(P^+)) - d(f(T_X), f(P^-)) + \alpha, 0), \quad (2)$$

where P^+ and P^- are positive and hard-negative spans, d is cosine distance, and $\alpha=0.15$ is the

margin. Once trained, the span encoder is frozen and shared across all downstream components. Training data consists of query-answer pairs from SimpleQA (Wei et al., 2024), Natural Questions (Kwiatkowski et al., 2019), and WikiHop (Welbl et al., 2018), converted into graph form via GPT-4 and augmented with relation synonyms, alternative reasoning paths, and hard negatives at varying reasoning depths. Entities and relations appearing in any evaluation split are excluded from training to prevent leakage; full details are in Appendix A.2.

Retrieve-rerank. Because exhaustive search over $\mathcal{G}$ is intractable, we score candidate span extensions with a two-stage pipeline. Given the current partial span $P \in \mathcal{F}_k$ (a search beam), we enumerate all one-hop extensions:

$$P' = P \oplus (e, r, x), \quad (e, r, x) \in \mathcal{T},$$

where e is a known entity in P and $\oplus$ denotes appending a triplet to the span. Let $\text{tok}(\cdot)$ denote the linearized token sequence of a span produced by the encoder’s tokenizer. The pipeline then scores each P' as follows: (i) a **bi-encoder retriever** computes the cosine similarity between the frozen span embeddings $f(T_{X_k})$ and $f(P')$ to efficiently short-list the most promising extensions; (ii) a **cross-**

encoder reranker jointly encodes the concatenation of the token sequences of T_{X_k} and P' through a dedicated encoder via full self-attention, where $z = \text{CrossEnc}(\text{tok}(T_{X_k}) [\text{SEP}] \text{tok}(P'))$ denotes the [CLS] embedding. The compatibility score is:

$$\sigma(T_{X_k}, P' | P) = \text{head}(z \parallel f(T_{X_k}) \parallel f(P')), \quad (3)$$

where $\parallel$ denotes concatenation, $\sigma \in [0, 1]$, and $\text{head}(\cdot)$ are two fully-connected layers with sigmoid output.

Completeness classifier. A three-class classifier $h_\theta(T_{X_k}, P')$ maps each candidate P' to $\{\text{STOP}, \text{CONTINUE}, \text{DROP}\}$ —determining whether P' satisfies the structural and semantic constraints of T_{X_k} , partially matches them and requires further expansion, or is incompatible with them entirely. It shares the same architecture as the reranker: the [CLS] embedding z from the cross-encoder is concatenated with $f(T_{X_k})$ and $f(P')$, and passed through a lightweight classification head to produce the final decision. Together with σ , this defines the scoring function from Eq. 1:

$$\begin{aligned} \text{score}(X_k, x | \hat{\mathcal{G}}_{k-1}, \mathcal{G}) \\ = \mathbf{1}[h_\theta(T_{X_k}, P') = \text{STOP}] \\ + \sigma(T_{X_k}, P' | P) \cdot \mathbf{1}[h_\theta(T_{X_k}, P') = \text{CONT.}], \end{aligned} \quad (4)$$

where spans with $h_\theta(T_{X_k}, P') = \text{DROP}$ are discarded. The completeness classifier h_θ is trained on a dataset directly derived from the one used for the span encoder, with labels derived from the same augmentation and corruption procedures; full details are provided in Appendix A.3.

Beam search. We maintain a beam $\mathcal{F}_k$ of partial subgraph spans with beam size $b=10$ and maximum depth $D_{\max}=3$. At each step, we take the top- b spans from $\mathcal{F}_k$, generate all one-hop extensions P' for each, and score them via Eq. 4. The beam is then updated by merging the current spans with the newly scored extensions and retaining the top- b by score. The top span in the updated beam is then passed to the completeness classifier: if $h_\theta(T_{X_k}, P') = \text{STOP}$, X_k is resolved and expansion halts; if $h_\theta(T_{X_k}, P') = \text{CONTINUE}$, the span is expanded in the next step; if $h_\theta(T_{X_k}, P') = \text{DROP}$, it is discarded and the next-best span in the beam is promoted. This process repeats until X_k is resolved or $\text{depth}(P) = D_{\max}$ for all spans in the beam.

3.4 Reasoning

Once exploration terminates, we perform a single aggregation step with GPT-4 (prompt in Section A.1). The top candidate entities, their supporting KG paths, the original query, and the fully disambiguated AKG are provided as context. The LLM receives the spans labelled STOP by the completeness classifier as candidates; if none are available, the top CONTINUE spans are provided instead.

To support multi-answer queries, the resolved AKG defines a traversal pattern executed against $\mathcal{G}$; the final answer set is the deduplicated union of LLM-selected and pattern-matched entities.

4 Experiments

4.1 Experimental Design

Following prior work (Sun et al., 2023; Tan et al., 2025a; Ma et al., 2024; Li et al., 2023b), we evaluate GRASE on ten datasets spanning KGQA, open-domain QA, slot filling, and fact checking. For KGQA we use four multi-hop sets—CWQ (Talmon and Berant, 2018), WebQSP (Yih et al., 2016), QALD-10 (Perevalov et al., 2022), and AdvHotpotQA (Ye and Durrett, 2022)—and one single-hop set, Simple Questions (Bordes et al., 2015), all executed over Wikidata (Vrandečić and Krötzsch, 2014) and Freebase (Bollacker et al., 2008). For open-domain QA we use WebQuestions (Berant et al., 2013); for slot filling, T-REx (Elsahar et al., 2018) and ZeroShotRE (Petroni et al., 2021); and for fact checking, FEVER (Thorne et al., 2018) and CREAK (Onoe et al., 2021). Our primary metric is Hits@1 (Li et al., 2023a; Baek et al., 2023; Sun et al., 2023); we also report F1, average LLM calls, and wall-clock time per query. Because LLMs can be non-deterministic even at temperature 0.0 (Lops et al., 2024; Atil et al., 2025), we run each method five times and report mean $\pm$ std. Ablations examine (i) incremental addition of retrieval components, (ii) beam width and depth, and (iii) LLM backbone sensitivity. Base configuration details are in Section A.5.

4.2 Main Results

KGQA benchmarks. Table 1 compares GRASE against classical KGQA systems and recent LLM-based pipelines on WebQSP and CWQ. On WebQSP, GRASE achieves 89.1 Hits@1 and 73.5 F1, competitive with the strongest prior system PathMind (89.5 / 72.8) while outperforming all GNN-

Model	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
GraftNet	66.7	62.4	36.8	32.7
NSM	68.7	62.8	47.6	42.4
UniKGQA	72.2	72.2	51.2	49.1
ReaRev	76.4	70.9	52.9	47.8
G-Retriever	70.1	–	–	–
Qwen2-7B	49.6±1.6	34.1±1.3	24.4±1.4	20.7±1.5
Llama3-8B	54.1±1.3	33.5±1.6	19.3±1.2	21.5±1.5
GPT-4o	60.1±1.1	42.4±1.4	42.1±1.7	37.0±1.2
TOG	81.5±0.6	69.8±0.7	67.2±1.3	60.9±0.9
ROG	84.6±0.8	69.7±0.7	61.4±1.2	55.1±1.1
LightProof [†]	82.7	–	58.1	–
GNN-RAG	85.4±0.5	67.9±0.8	66.1±0.9	58.0±0.7
SubGraphRAG [†]	86.6	70.6	47.2	57.0
EPERM [†]	88.8	72.4	66.2	58.9
PathMind [†]	89.5	<u>72.8</u>	70.7	<u>61.4</u>
POG	86.4±0.6	70.6±0.4	<u>70.9±1.1</u>	60.2±0.8
GRASE	<u>89.1±0.3</u>	73.5±0.7	73.9±1.1	61.8±0.4

Table 1: Results on WebQSP and CWQ. Best in bold, second-best underlined. [†] indicates results taken from prior work (no public code; SubGraphRAG not reproducible). LLM-based methods report mean $\pm$ std over 5 runs; this does not apply to [†]. GPT-4 is used when available for fair comparison.

based methods. On CWQ, which requires deeper multi-hop reasoning, GRASE attains 73.9 Hits@1 and 61.8 F1, surpassing all prior systems including PathMind (70.7 / 61.4) and POG (72.0). The consistent F1 gains suggest that GRASE retrieves higher-quality supporting spans rather than relying on isolated match signals. Figure 3 further breaks down CWQ performance by reasoning depth, and Figures 4 and 5 analyze AKG robustness and encoder generalization respectively.

Broader generalization. Table 2 reports results across the remaining eight datasets. On slot-filling tasks (T-REx, Zero-Shot RE), GRASE leads with 94.86 and 94.20 Hits@1, with the completeness classifier halting early on single-hop queries. On multi-hop benchmarks (AdvHotpotQA, QALD-10-en), GRASE outperforms ToG-2 by 17.4 and 11.6 points. Notably, PathMind—the strongest KGQA baseline—has not reported results beyond WebQSP and CWQ, and does not generalize across this broader evaluation. GRASE is competitive on CREAK (92.77 vs. 93.8 for ToG) and substantially stronger on FEVER (79.12 vs. 52.7).

4.3 Inference Cost

By design, GRASE makes at most $D_{\max}+2$ LLM calls per query—one per depth level for seed disambiguation, one for planning, and one for final reasoning—independent of beam width b . This compares favourably to $2bD_{\max}+D_{\max}+1$ for ToG and $\approx 3D_{\max}$ for ToG-2. As shown in Table 3, GRASE reduces average LLM calls by 85.3% versus ToG (2.4 vs. 16.3) and latency by 85.0% (10.4s vs. 69.3s), with proportional gains over all other baselines. These savings stem from encoder-driven exploration replacing per-hop LLM scoring, and the completeness classifier controlling overexpansion.

4.4 Ablation Study

We analyze the exploration stack by toggling three modules while keeping the planner, final-answer LLM, Wikidata KG, $b=10$, and $D_{\max}=3$ fixed. When a module is disabled, it is replaced by its LLM equivalent: BM25 replaces the bi-encoder, and the LLM provides scalar scores or Stop/Continue/Drop decisions in place of the cross-encoder and completeness classifier respectively.

As shown in Table 4, the LLM-native baseline is costly since the LLM both ranks neighbors and decides completeness at each hop (55.0 calls, 120.9s). Adding the bi-encoder (V1) reduces fan-out and latency with modest accuracy gain. The cross-encoder (V2) is the inflection point: ranking moves off the LLM, calls drop sharply (14.8), and accuracy rises as better candidates are promoted. Adding the completeness classifier (V3) removes LLM calls from exploration entirely, reducing to ≈ 2.4 calls and 10.4s per query.

4.5 Width, Depth, and Backbone Sensitivity

Widening the beam helps up to $b \approx 10$ and then plateaus; increasing depth beyond $D \approx 3$ yields diminishing returns and can degrade accuracy due to noisier paths. With the completeness classifier active, time scales roughly as $b \cdot D$ (encoder work); without it, time grows dramatically with depth, confirming the classifier’s role in containing search (see Figure 8 in Appendix B).

Swapping only the planner/final-answer LLM while keeping the encoder stack fixed shows that GRASE is largely backbone-agnostic: GPT-4 gives the best QA average (82.6) and fact accuracy (85.9), with smaller models lagging by 1–3 pp (see Table 12 in Appendix B for full results). Average

Method	Adv HotpotQA	QALD- 10-en	FEVER	CREAK	Zero- Shot RE	Simple Questions	Web Questions	T-REx
ToG-2 (GPT-3.5)	42.9	54.1	63.1 [†]	93.5	91.0	—	—	—
ToG (GPT-4)	26.3	50.2	52.7	93.8	88.0	53.6	54.5	76.8
PoG-E (GPT-4)	—	—	—	—	—	81.2	82.0	—
PoG (GPT-4)	—	—	—	—	—	84.0	84.6	—
IO prompt (ChatGPT)	—	42.0	—	89.7	27.7	20.0	48.7	33.6
CoT (ChatGPT)	—	42.9	—	90.1	28.8	20.3	48.5	32.0
CoK (6-shot)	35.4	—	58.5	—	—	—	—	—
GRASE (ours)	60.31±1.39	65.67±3.84	79.12±4.04	92.77±3.74	94.20±2.79	82.87±2.07	88.15±4.87	94.86±3.57

Table 2: Broader generalization results. Hits@1 for AdvHotpotQA, QALD-10-en, Zero-Shot RE, SimpleQuestions, WebQuestions, and T-REx; Accuracy for FEVER and CREAK. For fact-checking tasks, the reasoning prompt is extended to include a truth assessment step over the retrieved evidence. GRASE results are mean $\pm$ std over five runs; baseline results are taken as published. [†]ToG-2 FEVER uses 3-shot prompting.

System	LLM calls (avg.)	Time / query (s)
ToG	16.3	69.3
ToG-2	5.4	27.3
PoG	7.8	34.6
CoK	11.0	30.1
RoG	4.7	44.8
GRASE	2.4	10.4

Table 3: Average LLM calls and execution time per query. Lower is better.

LLM calls remain at ≈ 2.4 regardless of backbone, confirming that efficiency and most accuracy gains stem from the exploration stack rather than the reasoning LLM. Encoder-capacity scaling and controls for the AKG representation and span-based retriever are reported in Tables 9 to 11.

4.6 Multi-hop Reasoning Depth

Figure 3 reports CWQ F1 broken down by the number of reasoning hops required. GRASE consistently outperforms RoG and PathMind at every depth level. For single-hop questions the margin is small (0.780 vs. 0.775 for PathMind), as all methods handle direct lookups well. The gap widens at two hops (0.648 vs. 0.605) and is most pronounced at three or more hops (0.422 vs. 0.406), where preserving global graph structure via the AKG becomes critical. The graceful degradation with depth—as opposed to the sharper drop observed in RoG (0.629 $\rightarrow$ 0.378)—confirms that anonymous-graph matching maintains reasoning coherence across longer chains.

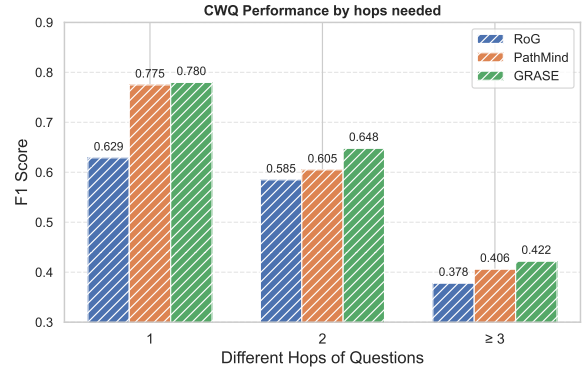


Figure 3: CWQ F1 by reasoning depth. GRASE degrades more gracefully than RoG and PathMind as hop count increases.

4.7 AKG Robustness Analysis

To assess sensitivity to planning errors, we induce controlled perturbations in the AKG generated for CWQ queries requiring two or more hops, varying the fraction of affected elements from 10% to 90%. We consider four error types: *removed edges* (edges deleted from the AKG), *uninformative substitution* (an entity or relation replaced with an uninformative placeholder), *synonym substitution* (replaced with a semantically equivalent term), and *removed entities* (an entity and all its incident edges removed).

Results in Figure 4 reveal distinct robustness profiles. Synonym substitution is the most benign: F1 remains stable across all perturbation levels (0.61 $\rightarrow$ 0.57), confirming that fuzzy span matching handles semantic variation well. Removed edges and uninformative substitutions cause moderate degradation (0.56 $\rightarrow$ 0.20 and 0.58 $\rightarrow$ 0.28 at 90% perturbation), as the AKG loses structural constraints

Variant	Bi-enc. retr.	Cross-enc. rerank	Compl. class.	Hits@1/Acc (%)	LLM calls (avg)	Time / query (s)
Baseline	N	N	N	70.2	55.0	120.9
V1	Y	N	N	73.0	50.5	90.2
V2	Y	Y	N	78.5	14.8	38.6
V3 (full)	Y	Y	Y	81.8	2.4	10.4

Table 4: Ablation over exploration components. Baseline: plain expansion with LLM-based ranking and completeness. V1: +bi-encoder shortlist. V2: +cross-encoder σ (LLM no longer ranks). V3: +completeness classifier (exploration decisions no longer call the LLM).

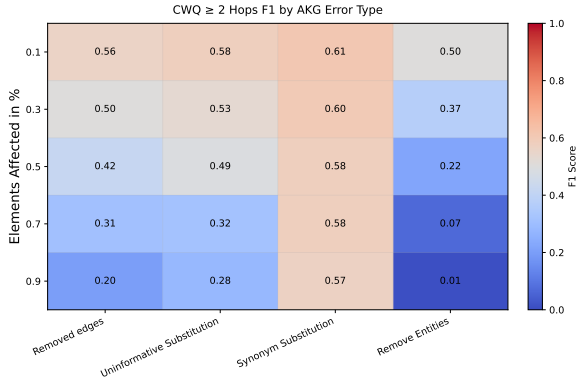


Figure 4: CWQ F1 under controlled AKG perturbations (≥ 2 hops). Synonym substitution has minimal impact; entity removal is the most damaging error type.

that guide exploration. Entity removal is the most damaging: F1 collapses to near zero at high rates ($0.50 \rightarrow 0.01$), since missing anchor entities prevent exploration from initialising correctly. These results show that GRASE is robust to lexical variation but relies on structural completeness of the AKG, motivating careful entity linking at planning time.

4.8 Encoder Generalization to Unseen Relations

A practical concern for deployment is whether the span encoder generalizes to relations unseen during training. Figure 5 reports CWQ F1 as an increasing fraction of relations is withheld from training (0% to 70%).

F1 remains stable up to 50% held-out relations ($0.638 \rightarrow 0.619$), with a modest drop only at 60–70% ($0.599, 0.571$). The narrow standard deviation band confirms consistent behavior across runs. This robustness stems from the fuzzy matching objective: the encoder learns to match spans by semantic content and structural role rather than memorizing specific relation surface forms, allowing generalization via synonymy and structural analogy. Performance degrades only when the majority

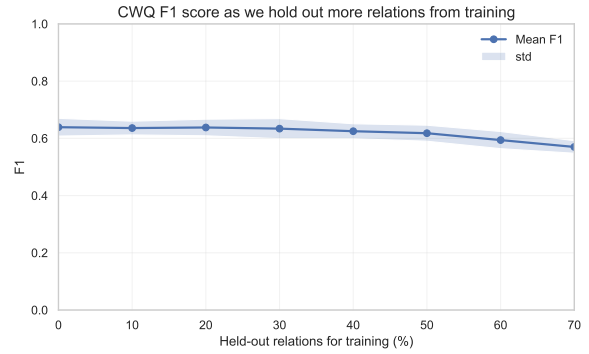


Figure 5: CWQ F1 as an increasing fraction of relations is held out from encoder training. The held-out fraction is defined relative to the total set of relations in the training data. Performance remains stable up to 50% held-out relations, confirming generalization via semantic matching rather than relation memorization.

of relations are withheld, suggesting that a moderately diverse training set is sufficient for strong generalization.

5 Limitations and Future Work

GRASE is designed for structured KG retrieval and does not currently incorporate unstructured evidence sources such as free text or tables; extending to hybrid settings would require additional alignment modules, which we leave to future work. Like all retrieval systems grounded in external KGs, performance is bounded by KG coverage and the quality of entity linking at planning time—a limitation shared by all methods in our comparison. The trained encoder components may require re-tuning when applied to KGs with substantially different relation distributions, though our held-out relation experiments (Figure 5) suggest the encoder generalizes well up to 50% unseen relations. Finally, the single LLM call at reasoning time introduces a small residual dependence on LLM reliability for answer verbalization and aggregation.

6 Conclusion

We introduced GRASE, a KGQA framework that replaces LLM-directed graph exploration with encoder-guided fuzzy span matching over anonymous knowledge graphs. By confining LLM calls to planning and final answer synthesis, GRASE reduces latency and cost by 60–80% while matching or exceeding state-of-the-art accuracy across ten diverse benchmarks spanning KGQA, slot filling, open-domain QA, and fact checking. Our broad evaluation suggests that encoder-driven retrieval with global query structure is a robust and practical alternative to fully LLM-guided pipelines across task types and reasoning depths.

Acknowledgments

The research of Guzman-Olivares was conducted with the support of Bulil Technologies S.L., which provided the hardware for the development of this project, and with financial support from the Spanish Ministry of Science and Innovation, grants PID2022-139131NB-I00 and PID2025-167968NB-I00. The research of Quijano-Sanchez was conducted with financial support from the Spanish Ministry of Science and Innovation, grants PID2022-139131NB-I00, PR47/24 PID2024-157561NB-I00, and PID2025-167968NB-I00. The research of Liberatore was conducted with financial support from the Spanish Ministry of Science and Innovation, grant PID2025-167968NB-I00.

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. Association for Computational Linguistics.
- Meta AI. 2024a. The LLaMA 3 model card and technical report. <https://ai.meta.com/research/publications/the-llama-3-model-card-and-technical-report/>. Accessed: 2025-06-23.
- Mistral AI. 2024b. Mistral large. <https://mistral.ai/news/mistral-large/>. Accessed: 2025-06-23.
- Anthropic. 2024. Introducing claude 3.5 sonnet. <https://www.anthropic.com/news/claude-3-5-sonnet>. Accessed: 2025-06-23.
- Tu Ao, Yanhua Yu, Yuling Wang, Yang Deng, Zirui Guo, Liang Pang, Pinghui Wang, Tat-Seng Chua, Xiao Zhang, and Zhen Cai. 2025. [Lightprof: a lightweight reasoning framework for large language model on knowledge graph](#). In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’25/IAAI’25/EAAI’25. AAAI Press.
- Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. 2025. [Non-determinism of "deterministic" llm settings](#). *Preprint*, arXiv:2408.04667.
- Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. [Knowledge-augmented language model prompting for zero-shot knowledge graph question answering](#). In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCH-ING 2023)*, pages 70–98, Toronto, ON, Canada. Association for Computational Linguistics.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. [Qwen technical report](#). *Preprint*, arXiv:2309.16609.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. [Semantic parsing on Freebase from question-answer pairs](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. 2024. [Graph of thoughts: solving elaborate problems with large language models](#). In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’24/IAAI’24/EAAI’24. AAAI Press.
- Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. [Freebase: a collaboratively created graph database for structuring human knowledge](#). In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. [Large-scale simple question answering with memory networks](#). *Preprint*, arXiv:1506.02075.

- Lingjiao Chen, Matei A. Zaharia, and James Y. Zou. 2023. [Frugalgpt: How to use large language models while reducing cost and improving performance](#). *ArXiv*, abs/2305.05176.
- Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. [T-REx: A large scale alignment of natural language with knowledge base triples](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. [Complexity-based prompting for multi-step reasoning](#). *Preprint*, arXiv:2210.00720.
- Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021a. [Improving multi-hop knowledge base question answering by learning intermediate supervision signals](#). In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining, WSDM '21*, page 553–561, New York, NY, USA. Association for Computing Machinery.
- Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021b. [Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing](#). *ArXiv*, abs/2111.09543.
- Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. [G-retriever: retrieval-augmented generation for textual graph understanding and question answering](#). In *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24*, Red Hook, NY, USA. Curran Associates Inc.
- Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2024. [A survey of knowledge enhanced pre-trained language models](#). *IEEE Trans. on Knowl. and Data Eng.*, 36(4):1413–1430.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. [A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions](#). *ACM Trans. Inf. Syst.*, 43(2).
- Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. [Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity](#). 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 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.
- Che Jiang, Biqing Qi, Xiangyu Hong, Dayuan Fu, Yang Cheng, Fandong Meng, Mo Yu, Bowen Zhou, and Jie Zhou. 2024. [On large language models' hallucination with regard to known facts](#). 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 1041–1053, Mexico City, Mexico. Association for Computational Linguistics.
- Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. [Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph](#). *Preprint*, arXiv:2212.00959.
- Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhan Wang, Yu Meng, and Jiawei Han. 2024. [Graph chain-of-thought: Augmenting large language models by reasoning on graphs](#). *Preprint*, arXiv:2404.07103.
- Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. [KG-GPT: A general framework for reasoning on knowledge graphs using large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9410–9421, Singapore. Association for Computational Linguistics.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. [Large language models are zero-shot reasoners](#). In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA. Curran Associates Inc.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- Mufei Li, Siqi Miao, and Pan Li. 2025. [Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation](#). In *The Thirteenth International Conference on Learning Representations*.
- Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhui Chen. 2023a. [Few-shot in-context learning on knowledge base question answering](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6966–6980, Toronto, Canada. Association for Computational Linguistics.

- Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq R. Joty, Soujanya Poria, and Lidong Bing. 2023b. [Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources](#). In *International Conference on Learning Representations*.
- Yu Liu, Xixun Lin, Yanmin Shang, Yangxi Li, Shi Wang, and Yanan Cao. 2025. [Pathmind: A retrieve-prioritize-reason framework for knowledge graph reasoning with large language models](#). *Preprint*, arXiv:2511.14256.
- Xiao Long, Liansheng Zhuang, Aodi Li, Minghong Yao, and Shafei Wang. 2025. [Eperm: an evidence path enhanced reasoning model for knowledge graph question and answering](#). In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press.
- Pasquale Lops, Antonio Silletti, Marco Polignano, Cataldo Musto, and Giovanni Semeraro. 2024. [Reproducibility of llm-based recommender systems: the case study of p5 paradigm](#). In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, page 116–125, New York, NY, USA. Association for Computing Machinery.
- 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*.
- Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2024. [Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation](#). In *International Conference on Learning Representations*.
- Alan Malek, Jiawei Ge, Nevena Lazic, Chi Jin, András György, and Csaba Szepesvári. 2025. [Frontier llms still struggle with simple reasoning tasks](#). *Preprint*, arXiv:2507.07313.
- Costas Mavromatis and George Karypis. 2022. [ReaRev: Adaptive reasoning for question answering over knowledge graphs](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Costas Mavromatis and George Karypis. 2024. [Gnn-rag: Graph neural retrieval for large language model reasoning](#). *Preprint*, arXiv:2405.20139.
- Agada Joseph Oche, Ademola Glory Folashade, Tirthankar Ghosal, and Arpan Biswas. 2025. [A systematic review of key retrieval-augmented generation \(rag\) systems: Progress, gaps, and future directions](#). *Preprint*, arXiv:2507.18910.
- Yasumasa Onoe, Michael J. Q. Zhang, Eunsol Choi, and Greg Durrett. 2021. [Creak: A dataset for commonsense reasoning over entity knowledge](#). *Preprint*, arXiv:2109.01653.
- OpenAI. 2023. Gpt-3.5 models. <https://platform.openai.com/docs/models/gpt-3-5>. Accessed: 2025-06-23.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, 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.
- Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. [Unifying large language models and knowledge graphs: A roadmap](#). *IEEE Trans. on Knowl. and Data Eng.*, 36(7):3580–3599.
- 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.
- Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. [Knowledge enhanced contextual word representations](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.
- Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. [KILT: a benchmark for knowledge intensive language tasks](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.
- Mobashir Sadat, Zhengyu Zhou, Lukas Lange, Jun Araki, Arsalan Gundroo, Bingqing Wang, Rakesh Menon, Md Parvez, and Zhe Feng. 2023. [DelusionQA: Detecting hallucinations in domain-specific question answering](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 822–835, Singapore. Association for Computational Linguistics.
- Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. [Improving multi-hop question answering over knowledge graphs using knowledge base embeddings](#).

- In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4498–4507, Online. Association for Computational Linguistics.
- Priyanka Sen, Armin Oliya, and Amir Saffari. 2021. [Expanding end-to-end question answering on differentiable knowledge graphs with intersection](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8805–8812, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jiaxin Shi, Shulin Cao, Lei Hou, Juanzi Li, and Hanwang Zhang. 2021. [TransferNet: An effective and transparent framework for multi-hop question answering over relation graph](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4149–4158, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. 2025. [Fast or better? balancing accuracy and cost in retrieval-augmented generation with flexible user control](#). *Preprint*, arXiv:2502.12145.
- Ying Su, Jipeng Zhang, Yangqiu Song, and Tong Zhang. 2024. [PipeNet: Question answering with semantic pruning over knowledge graphs](#). In *Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024)*, pages 360–371, Mexico City, Mexico. Association for Computational Linguistics.
- Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. [Open domain question answering using early fusion of knowledge bases and text](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.
- Jiashuo Sun, Chengjin Xu, Luminyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung yeung Shum, and Jian Guo. 2023. [Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph](#). In *International Conference on Learning Representations*.
- Alon Talmor and Jonathan Berant. 2018. [The web as a knowledge-base for answering complex questions](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.
- Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025a. [Paths-over-graph: Knowledge graph empowered large language model reasoning](#). In *Proceedings of the ACM on Web Conference 2025*, pages 3505–3522.
- Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. 2025b. [Hydrarag: Structured cross-source enhanced large language model reasoning](#). *Preprint*, arXiv:2505.17464.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. [FEVER: a large-scale dataset for fact extraction and VERification](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.
- Denny Vrandečić and Markus Krötzsch. 2014. [Wiki-data: A free collaborative knowledgebase](#). *Communications of the ACM*, 57(10):78–85.
- Sebastian Walter and Hannah Bast. 2025. [Grasp: Generic reasoning and sparql generation across knowledge graphs](#). *Preprint*, arXiv:2507.08107.
- Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. [Measuring short-form factuality in large language models](#). *ArXiv*, abs/2411.04368.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS ’22*, Red Hook, NY, USA. Curran Associates Inc.
- Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. [Constructing datasets for multi-hop reading comprehension across documents](#). *Transactions of the Association for Computational Linguistics*, 6:287–302.
- Yilin Wen, Zifeng Wang, and Jimeng Sun. 2024. [MindMap: Knowledge graph prompting sparks graph of thoughts in large language models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10370–10388, Bangkok, Thailand. Association for Computational Linguistics.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. [Tree of thoughts: Deliberate problem solving with large language models](#). *Preprint*, arXiv:2305.10601.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. [React: Synergizing reasoning and acting in language models](#). *Preprint*, arXiv:2210.03629.
- Xi Ye and Greg Durrett. 2022. [The unreliability of explanations in few-shot prompting for textual reasoning](#). In *Proceedings of the 36th International Conference on Neural Information Processing Systems*,

NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Wentau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. [The value of semantic parse labeling for knowledge base question answering](#). In *Annual Meeting of the Association for Computational Linguistics*.

Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2022a. [Poolingformer: Long document modeling with pooling attention](#). *Preprint*, arXiv:2105.04371.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. [Automatic chain of thought prompting in large language models](#). *Preprint*, arXiv:2210.03493.

A Additional Implementation Details

A.1 Prompts

Anonymous Graph Construction Prompt. In our framework, large language models are used in two well-defined roles during inference. First, we employ a *graph-construction prompt* (see Figure 6) to convert a natural-language question into an *anonymous knowledge graph* (AKG): a set of logical triplets in which unknown entities or relations are replaced by anonymous variables X_i . This abstraction allows the model to express the structural constraints implied by the question while postponing the resolution of concrete entities to later stages. Importantly, anonymisation prevents the model from relying on memorised world knowledge and ensures that downstream reasoning is grounded in the KG rather than in the language model itself.

Reasoning Prompt. Second, we use a *reasoning prompt* (Figure 7) to select the final answer once candidate entities and supporting KG paths have been retrieved by exploration. The model receives the completed or best available spans and selects the candidate that best satisfies the logical constraints encoded in the AKG, acting as a consistency checker over structured evidence rather than as an external knowledge source.

A.2 Training the Span Encoder

Goal. We train a single span encoder to score compatibility between an *anonymous query subgraph* T_{X_k} and a *candidate KG span* P' , so that fuzzy matches (synonyms, variable-length alignments, type-consistent substitutions) receive high scores while incomplete or corrupted matches receive low scores. Concretely, the encoder is trained to map compatible query-span pairs to nearby embeddings and incompatible pairs to distant embeddings under the triplet-loss objective. Once trained, the encoder is frozen and its span embeddings $f(\cdot)$ are shared across the bi-encoder retriever, the cross-encoder reranker head, and the completeness classifier head (see Section 3.3).

Base model. The encoder is DeBERTa-v3-Large (He et al., 2021b), fine-tuned with the triplet loss defined in Eq. 2. The reranker and completeness classifier are lightweight linear heads trained on top of the frozen encoder.

Supervision source. We construct a training set

$$\mathcal{S}_{\text{enc}} = \{(T_{X_k}, P, y)\},$$

where T_{X_k} is the anonymous query subgraph incident to variable X_k , P is a candidate KG span, and $y \in \{0, 1\}$ is a binary compatibility label indicating whether P satisfies the constraints in T_{X_k} . This supervised pool is used to sample training triplets (T_{X_k}, P^+, P^-) for the triplet-loss objective, where P^+ is a compatible span and P^- is an incompatible span associated with the same T_{X_k} . We build $\mathcal{S}_{\text{enc}}$ from the train splits of SimpleQA (Wei et al., 2024), Natural Questions (Kwiatkowski et al., 2019), and WikiHop (Welbl et al., 2018). For each query q we convert it into its AKG $\hat{\mathcal{G}}_q = (E_q, R_q, T_q)$ via GPT-4 and identify the subset $T_{X_k} \subseteq T_q$ of triplets incident to each anonymous variable X_k . Where gold supporting paths are available (KILT (Petroni et al., 2021) format), we use them directly; otherwise, we mine minimal satisfying paths in the KG. The full construction procedure is given in Algorithm 1.

Positive examples. Starting from (T_{X_k}, P^+) pairs, we create hard positives via: (i) **synonym rewrites** of relations and entities (from a curated list of 20 alternatives per relation type) that preserve truth conditions; (ii) **neutral triplet insertion**—edges touching nodes in T_{X_k} that do not change the answer set; (iii) **node-order permutation**—random shuffling of triplet order in T_{X_k} to enforce order invariance. All positive augmentations are verified to preserve satisfaction before being added to $\mathcal{S}_{\text{enc}}$.

Negative examples. Hard negatives are generated by minimally breaking support: (i) **edge dropout**—remove a key hop from P^+ or a required constraint from T_{X_k} ; (ii) **relation/entity corruption**—replace a relation or entity with a near-synonym that changes truth conditions; (iii) **target corruption**—swap the answer with a type-consistent non-answer; (iv) **topology perturbation**—re-route or re-order hops to produce misleading chains. Negatives are verified to not satisfy T_{X_k} before inclusion.

Order invariance. To learn over sets of triplets rather than sequences, we randomly permute the triplets in T_{X_k} at sampling time and re-linearize them, so the encoder does not overfit to serialization order. The batch-sampling strategy is summarized in Algorithm 2.

Topology diversity. To avoid overfitting to chain-structured paths, we synthesize examples spanning chains, forks, diamonds, and short detours (1–3 hops), ensuring the encoder encounters multiple structural patterns satisfying the same T_{X_k} .

Data deduplication and leakage prevention. To prevent contamination between encoder training and downstream evaluation, we perform explicit entity and relation deduplication across all datasets. Concretely, we extract the full set of entities and relations appearing in the dev and test splits of all ten evaluation benchmarks and exclude any training pair in $\mathcal{S}_{\text{enc}}$ whose T_{X_k} or P^+ contains a matching entity or relation. This ensures that the span encoder never trains on structural patterns grounded in evaluation entities or relations. The held-out relation experiment in Figure 5 provides empirical confirmation that the encoder generalizes via semantic span matching rather than entity or relation memorization.

Binary label space. At the span-encoder stage, the label space is binary ($y \in \{0, 1\}$), since the goal is to construct compatible/incompatible query–span pairs from which triplets (T_{X_k}, P^+, P^-) can be sampled for retrieval and reranking.

A.3 Training the Completeness Classifier

The completeness classifier h_θ is trained on a derived dataset

$$\mathcal{S}_{\text{cls}} = \{(T_{X_k}, P, y)\},$$

constructed from $\mathcal{S}_{\text{enc}}$ by extending the binary compatibility labels to the three-way decision space $y \in \{\text{STOP}, \text{CONTINUE}, \text{DROP}\}$. In particular, STOP and DROP examples are inherited from the positive and negative constructions used for the span encoder, while CONTINUE examples are created by truncating gold supporting paths at intermediate nodes to produce partially correct but incomplete spans. Thus, the classifier is trained on the same underlying supervision source as the span encoder, but with an expanded label space tailored to the exploration decision.

Label construction. STOP labels are assigned to gold supporting paths P^+ and their hard-positive augmentations—synonym rewrites, neutral-triplet insertions, and node-order permutations—that fully satisfy T_{X_k} and contain the correct answer entity. CONTINUE labels are assigned to partially constructed paths generated by truncating P^+ at

intermediate nodes, where at least one required hop is missing but all existing hops are correct. DROP labels are assigned to hard negatives produced by the corruption procedures—edge dropout, relation/entity corruption, target corruption, and topology perturbation—subgraphs that are structurally plausible but do not satisfy T_{X_k} .

Training objective. The classifier is trained with standard cross-entropy loss on top of the frozen span encoder, sharing the same input representation $f(T_{X_k}) \parallel f(P')$ as the reranker but with a three-class output head. This construction ensures that the three classes are precisely defined in terms of graph-level satisfaction of the anonymous query constraints, and that the classifier is trained on the same distribution of structural patterns as the span encoder, making the two components fully consistent by design.

Class balance. To avoid the classifier collapsing to the majority class, we sample equal numbers of STOP, CONTINUE, and DROP examples per T_{X_k} during training, using the same permutation batching strategy described in Algorithm 2 to enforce order invariance.

Classifier evaluation. To verify that the completeness classifier makes meaningful decisions independently of the encoder, we evaluate h_θ in isolation on a held-out portion of $\mathcal{S}_{\text{cls}}$ not used during training. Table 5 reports precision, recall, and F1 per class.

Class	Precision	Recall	F1
STOP	0.91	0.87	0.89
CONTINUE	0.83	0.85	0.84
DROP	0.88	0.93	0.90
Macro avg.	0.87	0.88	0.88

Table 5: Completeness classifier performance on a held-out evaluation split.

The classifier achieves strong performance across all three classes, confirming that the accuracy gains observed in the ablation study (Table 4) reflect genuine improvements in exploration decisions rather than a confounded signal from the encoder alone.

A.4 Hardware Specifications

Our experiments were run on a workstation with an AMD Ryzen Threadripper 3975WX processor, 256 GB system memory, and two NVIDIA A100

Algorithm 1 Construction of training pairs for the span encoder

Require: KGQA datasets $\mathcal{D}$, KG $\mathcal{G}$, synonym maps, topology patterns, augmentation budget K

Ensure: Training set $\mathcal{S}_{\text{enc}} = \{(T_{X_k}, P, y)\}$

```

1:  $\mathcal{S}_{\text{enc}} \leftarrow \emptyset$ 
2: for all  $(q, \text{answers}) \in \mathcal{D}_{\text{train}}$  do
3:    $\hat{\mathcal{G}}_q \leftarrow \text{ANONGRAPH}(q)$ 
4:   for all anonymous variables  $X_k$  in  $\hat{\mathcal{G}}_q$  do
5:      $T_{X_k} \leftarrow \text{triplets in } T_q \text{ incident to } X_k$ 
6:      $\mathcal{P}^+ \leftarrow \text{MINE SUPPORTING PATHS}(T_{X_k}, \mathcal{G})$ 
7:     for all  $P^+ \in \mathcal{P}^+$  do
8:       Add  $(T_{X_k}, P^+, 1)$  to  $\mathcal{S}_{\text{enc}}$ 
9:     for  $j = 1$  to  $K$  do
10:       $(T_{X_k}^+, P_{\text{aug}}^+)$   $\leftarrow$  AUGMENTPOSITIVE( $T_{X_k}, P^+$ )
11:      if SATISFIES( $T_{X_k}^+, P_{\text{aug}}^+$ ) then
12:        Add  $(T_{X_k}^+, P_{\text{aug}}^+, 1)$  to  $\mathcal{S}_{\text{enc}}$ 
13:      end if
14:    end for
15:    for  $j = 1$  to  $K$  do
16:       $(T_{X_k}^-, P^-)$   $\leftarrow$  CORRUPTPAIR( $T_{X_k}, P^+$ )
17:      if NOTSATISFIES( $T_{X_k}^-, P^-$ ) then
18:        Add  $(T_{X_k}^-, P^-, 0)$  to  $\mathcal{S}_{\text{enc}}$ 
19:      end if
20:    end for
21:  end for
22: end for
23: end for
24: return  $\mathcal{S}_{\text{enc}}$ 

```

CPU	AMD Ryzen Threadripper 3975WX
RAM	256 GB
Cores	64
GPU	2 $\times$ NVIDIA A100 160GB

Table 6: Hardware used for all experiments.

GPUs with 160 GB memory each. All models were trained using mixed-precision computation; a single GPU was used per run unless otherwise noted.

A.5 Training Hyperparameters

We tune model and optimization hyperparameters on the development split of each dataset and select the final configuration based on validation performance averaged across runs. Search ranges and selected values are reported in Table 7. Unless otherwise stated, we use AdamW with cosine annealing, gradient clipping, and mixed-precision training. Beam search and expansion-depth parameters are fixed across all experiments.

A.6 Baseline Systems

We compare against the following systems. **Classical KGQA:** GraftNet (Sun et al., 2018), NSM

Algorithm 2 Batch sampling with random triplet permutation to enforce order invariance

Require: Training set $\mathcal{S}$, permutations per item M

```

1: for each minibatch do
2:   Sample a base set of items  $(T_{X_k}, P, y)$  from  $\mathcal{S}$ 
3:   for all sampled items  $(T_{X_k}, P, y)$  do
4:     for  $m = 1$  to  $M$  do
5:        $T_{X_k}^{(\pi_m)} \leftarrow \text{RANDOMPERMUTATION}(T_{X_k})$ 
6:        $P^{(\pi_m)} \leftarrow \text{OPTIONALSHUFFLE}(P)$ 
7:       Encode  $(T_{X_k}^{(\pi_m)}, P^{(\pi_m)})$ 
8:     end for
9:   end for
10: end for

```

(He et al., 2021a), UniKGQA (Jiang et al., 2023), ReaRev (Mavromatis and Karypis, 2022), and G-Retriever (He et al., 2024). **LLM-only:** Qwen2-7B (Bai et al., 2023), Llama3-8B (AI, 2024a), and GPT-4o (OpenAI et al., 2024). **LLM-guided retrieval:** ToG (Sun et al., 2023), RoG (Luo et al., 2024), LightProof (Ao et al., 2025), GNN-RAG (Mavromatis and Karypis, 2024), SubGraphRAG (Li et al., 2025), EPERM (Long et al., 2025), PathMind (Liu et al., 2025), and PoG (Tan et al., 2025a). **Broader evaluation baselines:** ToG-2 (Ma et al., 2024), CoK (Li et al., 2023b), IO prompt and CoT with ChatGPT (OpenAI et al., 2024).

A.7 KG Exploration Pseudocode

To identify candidate reasoning spans in the KG that satisfy the anonymous constraints encoded in the AKG, we perform guided beam search. At each step, the algorithm expands the current partial span by considering one-hop extensions from the currently resolved entity. The bi-encoder retriever shortlists promising extensions by cosine similarity between span embeddings; the cross-encoder reranker scores each extension via $\sigma(T_{X_k}, P' | P)$; and the completeness classifier determines whether the span satisfies the query (STOP), should be expanded (CONTINUE), or is irrelevant (DROP). Completed or depth-limited spans are stored as candidates; the top- b partial spans are retained for further expansion. The full procedure is given in Algorithm 3. The search is initialized from an anchor entity identified from the question and AKG, and proceeds by expanding partial spans while resolving anonymous variables according to a fixed exploration schedule.

B Additional Experiment Results

We report supplementary analyses to complement the main evaluation. Figure 8 studies the effect of

Parameter	Value
Optimizer	AdamW
Learning Rate	$\{10^{-7}, 10^{-6}, \mathbf{10^{-5}}, 10^{-4}\}$
Gradient Accumulation Steps	$\{1, \mathbf{5}, 10\}$
Maximum Gradient Norm	$\{\mathbf{1}, 5, 10, 50, 100\}$
Batch Size	$\{4, 16, 32, \mathbf{64}, 128\}$
Epochs	$\{1, 5, \mathbf{10}, 15, 20\}$
Evaluation Steps	1000
Scheduler	$\{\mathbf{Cosine Annealing}, \text{Linear}\}$
Span Encoder Triplet Loss α	$\{0.05, 0.1, \mathbf{0.15}, 0.2, 0.25\}$
Weight Decay	0.01
Max Tokens	512
$D_{\max}$	3
Beam size b	10
top_p	1
temperature	0.0
max_generated_tokens	2048

Table 7: Training hyperparameters. Bold marks the selected configuration.

Dataset	AKG correct (%)
WebQSP	91.0
CWQ	84.5
QALD-10	87.8
AdvHotpotQA	78.4
SimpleQuestions	93.2
WebQuestions	90.6
T-REx	96.5
ZeroShotRE	95.1
FEVER	85.5
CREAK	94.3

Table 8: AKG correctness by dataset. We sample 200 queries per dataset and use an LLM judge because gold AKG annotations are unavailable. An AKG is correct when it contains a valid entity–relation structure for answering the query and permits reconstruction of the original query.

Encoder	Params (M)	Hits@1	Time (s)
DeBERTa-v3-Large	304	81.8	10.4
DeBERTa-v3-Base	86	79.6	8.1
DeBERTa-v3-Small	44	76.9	6.2
DeBERTa-v3-XSmall	22	73.8	5.1

Table 9: Encoder scaling within the same GRASE framework. Smaller DeBERTa-v3 backbones reduce latency with moderate Hits@1 degradation.

beam size and maximum expansion depth on accuracy and runtime. Table 12 examines backbone sensitivity. Table 8 evaluates the quality of generated AKGs, while Tables 9 to 11 report encoder-capacity and representation/retriever controls.

Variant	Hits@1	LLM calls	Time (s)
GRASE (AKG)	81.8	2.4	10.4
With sub-queries	79.2	4.6	22.1

Table 10: AKG versus sub-query decomposition using the same DeBERTa-v3-Large encoder.

Variant	Retriever	Hits@1	Time (s)
GRASE	Span-based	81.8	10.4
Lightweight GNN	2-layer GAT	78.2	18.8

Table 11: Span encoder versus a lightweight GNN retriever trained on the same data.

Backbone	Avg. Hits@1 (QA)	Avg. Acc. (Fact)	LLM calls (avg.)	Time/query (s)
GPT-4 (baseline)	82.6	85.9	2.4	10.4
GPT-3.5	79.5	83.0	2.4	9.0
Llama-3 70B Instruct	81.0	85.0	2.4	11.6
Mistral Large Instruct	80.0	84.0	2.4	11.0
Claude 3.5 Sonnet	81.8	85.5	2.4	11.2

Table 12: Backbone comparison with GRASE ($b=10$, $D=3$). QA average aggregates WebQSP, AdvHotpotQA, QALD-10-en, Zero-Shot RE, CWQ, SimpleQuestions, WebQuestions, and T-REx; Fact average aggregates FEVER and CREAK. Comparison with GPT-3.5 (OpenAI, 2023), Llama-3 70B Instruct (AI, 2024a), Mistral Large Instruct (AI, 2024b), and Claude 3.5 Sonnet (Anthropic, 2024).

Algorithm 3 Anonymous KG Exploration with Beam Search

Require: Query q , AKG $\hat{\mathcal{G}}_q$, anchor entity e_{anchor} , beam size b , depth limit D_{max}
Ensure: Set of accepted spans $\mathcal{A}$

- 1: Initialize frontier $\mathcal{F}^0 \leftarrow \{(e_{\text{anchor}}, \emptyset)\}$
- 2: $\mathcal{A} \leftarrow \emptyset$, priority queue $Q \leftarrow \emptyset$
- 3: **for** $t = 0, 1, 2, \dots, D_{\text{max}} - 1$ **do**
- 4: **for all** $(e_m, P) \in \mathcal{F}^t$ **do**
- 5: Select next unresolved anonymous variable X_k according to the exploration schedule
- 6: Enumerate neighbors $\mathcal{N}(e_m) = \{(e_m, r, v) \in \mathcal{T}\}$
- 7: **for all** $(e_m, r, v) \in \mathcal{N}(e_m)$ **do**
- 8: Form candidate span $P' = P \oplus (e_m, r, v)$
- 9: Compute reranker score $\sigma(T_{X_k}, P' | P)$
- 10: $c \leftarrow \text{CLASSIFY}(T_{X_k}, P') \in \{\text{STOP}, \text{CONTINUE}, \text{DROP}\}$
- 11: **if** $c = \text{DROP}$ **then**
- 12: **continue**
- 13: **end if**
- 14: **if** $c = \text{STOP}$ **or** $\text{depth}(P') = D_{\text{max}}$ **then**
- 15: $\mathcal{A} \leftarrow \mathcal{A} \cup \{P'\}$
- 16: **else**
- 17: Push (v, P') into Q with priority $\sigma(T_{X_k}, P' | P)$
- 18: **end if**
- 19: **end for**
- 20: **end for**
- 21: $\mathcal{F}^{t+1} \leftarrow$ top- b items popped from Q
- 22: **if** $Q = \emptyset$ **then**
- 23: **break**
- 24: **end if**
- 25: **end for**
- 26: **return** $\mathcal{A}$

Task. Convert the input sentence into a set of logical triplets. Any unknown entity or relation should be represented as X_i , which will later be resolved against a knowledge graph. Also determine a resolution order over the anonymous variables.

Guidelines.

- Use relations as verbs wherever possible, and entities as nouns.
- Treat the most specific entity as the “anchor”.
- Always include at least one anonymous variable.
- Prefer finer-grained variables rather than merging distinct roles.
- Ignore negation and quantifiers (e.g., “never”, “only”, “always”), since the goal is evidence retrieval.

Do not create dummy triplets such as:

(X_1 , name, Duke Leto Atreides).

Output format. Return:

- A Python-style list of triplets
- A resolution order over variables (e.g., [X_1 , X_2 , X_3])

Examples

- “Berlin is the capital of Germany” → triplets=[(Berlin, capital_of, Germany)], order=[]
- “Barack Obama is married to a woman who has a child” → triplets=[(Barack Obama, married_to, X_1), (X_1 , parent_of, X_2)], order=[X_1 , X_2]
- “Tom Hanks acted in a movie directed by a Canadian” → triplets=[(Tom Hanks, acted_in, X_1), (X_1 , directed_by, X_2), (X_2 , nationality, Canadian)], order=[X_1 , X_2]

Now extract triplets and the resolution order for the following text. Keep the structure compact but do **not** drop constraints on anonymous variables.

Figure 6: Prompt template for constructing an anonymous knowledge graph (AKG) and resolution order from a natural-language question.

Inputs

- A query expressed as logical triplets containing anonymous variables X_i .
- A target anonymous variable to resolve (e.g., X_1).
- A set of candidate entities.
- A set of logical paths extracted from the knowledge graph, labelled as STOP (fully satisfying) or CONTINUE (partial, provided as fallback).

Task

- Select the candidate entity that best satisfies the logical constraints imposed by the query, prioritising STOP-labelled spans.
- Replace all occurrences of X_i in the triplets with the resolved entity.
- If no candidate satisfies the constraints, return “None”.

Example

- **Target:** X_1

- **Query:**

`[(X_1, director_of, Titanic),
(X_1, nationality, Canadian)]`

- **Candidates:** James Cameron; Steven Spielberg; Denis Villeneuve

- **Supporting KG paths:**

James Cameron $\rightarrow$ director_of $\rightarrow$ Titanic
James Cameron $\rightarrow$ nationality $\rightarrow$ Canadian

- **Answer:** James Cameron

Figure 7: Prompt template used to resolve an anonymous variable by selecting the candidate entity that best satisfies the AKG constraints, based on supporting KG paths returned by exploration.

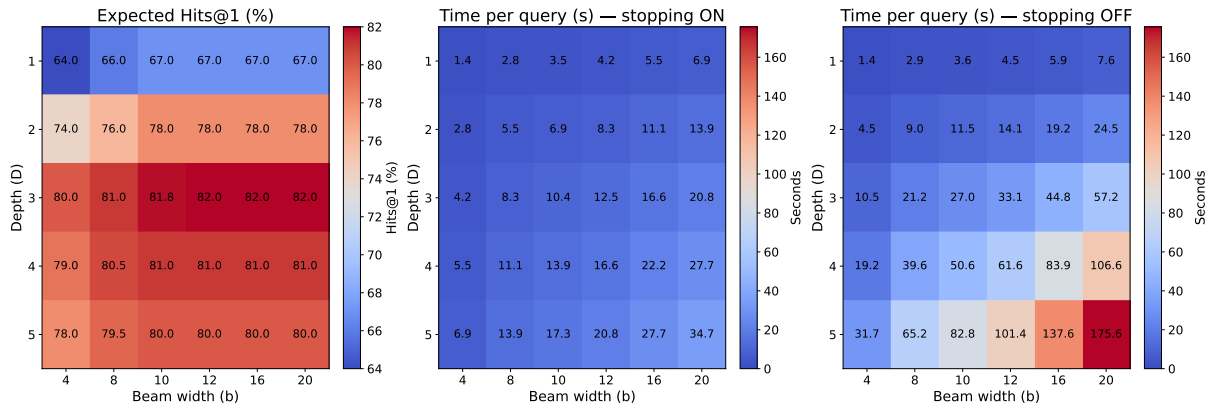


Figure 8: Beam-depth sweep. **Left:** Hits@1 (%). **Middle:** Time/query (s) with the completeness classifier active. **Right:** Time/query (s) with the completeness classifier deactivated (shared color scale with middle panel). Accuracy saturates around $b \approx 10$ and $D \approx 3$; without the classifier, runtime grows sharply with depth due to the absence of early stopping and pruning.