

GRAPE: GRAPH REASONING WITH ANONYMOUS PATH ENCODERS

Anonymous authors

Paper under double-blind review

ABSTRACT

Since the introduction of retrieval-augmented generation (RAG), a standard component of large language model (LLM) reasoning pipelines has been the navigation of a knowledge base (KB) to generate answers grounded in retrieved sources. However, recent studies show that LLMs struggle with complex queries requiring deep reasoning and interdependent knowledge, often leading to hallucinations. While several methods have been proposed to mitigate this issue, most rely on multiple additional calls to the LLM to decompose and validate reasoning steps, thereby increasing inference cost and latency. In this paper, we introduce GRAPE (Graph Reasoning with Anonymous Path Encoders), a framework that leverages path encodings over uncertain nodes and relations in knowledge graphs (KGs) to heuristically guide KB navigation. Rather than depending on a fully LLM-native retrieval pipeline, GRAPE replaces repeated model calls with encoder-only models that act as a semantic fuzzy query-matching engine. Experiments across multiple multi-hop QA benchmarks show that GRAPE achieves up to 85% faster inference than LLM-based pipelines, while consistently matching or exceeding state-of-the-art accuracy. These results demonstrate that encoder-only hybrid reasoning pipelines provide a practical and scalable alternative to expensive LLM-native retrieval, combining efficiency, robustness, and strong generalization.

1 INTRODUCTION

Writing a question is easy for humans, but despite strong performance across many tasks, large language models (LLMs) still struggle to answer queries that demand complex reasoning or deep, long-tail knowledge Petroni et al. (2021); Talmor et al. (2019); Malek et al. (2025); Tan et al. (2025b). Modern systems address this by supplying LLMs with retrieval mechanisms that surface evidence at inference time, mitigating the limits of pretraining Wei et al. (2022); Yao et al. (2023a); Besta et al. (2024); Jeong et al. (2024). Recently, structural signals from *knowledge graphs* (KGs) have been integrated for knowledge-graph question answering (KGQA), where answers are grounded in graph topology Sun et al. (2023); Li et al. (2023b); Agrawal et al. (2024). Since the emergence of retrieval-augmented generation (RAG) Lewis et al. (2020), many pipelines let the LLM act as an “agent,” issuing pre-formatted prompts to query, filter, and reason over the KG Sun et al. (2023); Tan et al. (2025a); Luo et al. (2024); Walter & Bast (2025).

This design has two drawbacks. **(i) Computational/latency:** multi-hop reasoning requires traversing and exploring the graph; adding LLM latency to every step makes end-to-end inference impractical at scale Oche et al. (2025). **(ii) Cost:** LLMs are flexible but expensive; driving the entire retrieval loop with recurrent model calls quickly becomes prohibitive Chen et al. (2023).

To address these issues, we propose GRAPE—a graph reasoning system with an *anonymous paths encoder*. GRAPE outsources LLM usage from the exploration loop to encoders that perform *fuzzy* path matching between a query-derived anonymous KG pattern and the target KG, where nodes may be variables to be resolved. In doing so, we reformulate KG retrieval as encoder-guided graph matching, preserving reasoning structure while reducing LLM calls, latency, and cost, and enabling scalable KGQA.

2 RELATED WORK

In general, LLM-based question answering is highly susceptible to hallucination, where plausible but unfounded intermediate steps lead to errors in complex reasoning tasks Sadat et al. (2023); Huang et al. (2025); Jiang et al. (2024). Early efforts to mitigate this problem emerged with Chain-of-Thought (CoT) Wei et al. (2022) and its many variants—such as Auto-CoT Zhang et al. (2022b), Complex-CoT Fu et al. (2023), Zero-Shot CoT Kojima et al. (2022), Tree-of-Thought (ToT) Yao et al. (2023a), and Graph-of-Thought (GoT) Besta et al. (2024)—which introduce intermediate reasoning steps into the prompt to guide the model’s reasoning and reduce hallucination. While these approaches produce more grounded responses, their effectiveness on knowledge-intensive or multi-hop reasoning tasks remains constrained by the model’s internal training data, motivating the integration of external knowledge sources during reasoning Luo et al. (2024); Li et al. (2023b); Yao et al. (2023b).

One natural extension has been to use knowledge graphs (KGs) as an external interface that models can query during reasoning, thereby alleviating both the training-data limitation and hallucination Pan et al. (2024); Agrawal et al. (2024). Early knowledge-graph question answering (KGQA) approaches attempted to embed KG structure directly into transformer architectures at training and fine-tuning time Zhang et al. (2022a); Peters et al. (2019); Li et al. (2023b). However, this strategy sacrifices scalability, as the model becomes tightly coupled to the training-time KG, and reduces representational reasoning flexibility Wen et al. (2024); Hu et al. (2024). More recent KGQA systems instead introduce additional extraction and processing steps within the native LLM-based QA pipeline, which circumvents the scalability issue Ma et al. (2024); Jin et al. (2024).

Among this new family of models, the most relevant to our work are: *Think on Graph* (ToG) Sun et al. (2023), which performs beam search over KG paths to identify the most likely reasoning chains; *Think on Graph 2.0* (ToG-2) Ma et al. (2024), which extends ToG by tightly coupling KG traversal with entity-linked document retrieval, allowing structured and unstructured sources to reinforce each other in multi-hop reasoning; and *Paths on Graphs* (PoG) Tan et al. (2025a), which adds a pruning step using an off-the-shelf model such as SBERT Reimers & Gurevych (2019) to filter candidate paths after the expansion and search phases, thereby reducing the number of reasoning calls.

Although these approaches achieve significantly better performance on complex reasoning tasks than CoT-based methods, they incur substantial overhead in both cost and latency, as their retrieval pipelines require repeated LLM calls Chen et al. (2023); Oche et al. (2025); Wang et al. (2024); ?. The bottleneck typically lies in the graph exploration phase, where most methods rely on iterative expansion, pruning, and evaluation, invoking the LLM at every step of the planned reasoning process during the initial query resolution Su et al. (2025). Some recent work has explored dynamic graph pruning for KGQA—for example, KAPING Baek et al. (2023), which prunes at the triplet level (entity–relation–entity) using semantic similarity between questions and triplets; KG-GPT Kim et al. (2023), which decomposes the main question into sub-questions to retrieve relevant relations; and PipeNet Su et al. (2024), which prunes nodes based on their distance to query-matched sub-spans. While these methods improve efficiency, their pruning operates locally at the triplet level, often overlooking the global graph structure and sacrificing performance on long-range dependencies Tan et al. (2025a).

To the best of our knowledge, no existing KGQA method dynamically prunes graph exploration paths while considering the global graph structure and avoiding additional LLM calls. To this end, we introduce GRAPE, a simple framework based on path encoding with uncertain (anonymous) entities that reformulates KGQA as a fuzzy graph matching problem between the query graph and the KG.

3 METHODOLOGY

The GRAPE framework aims to reduce reliance on repeated external calls to the LLM within the retrieval pipeline by replacing LLM-based search with fuzzy path matching between the knowledge graph (KG) and an Anonymous Knowledge Graph (AKG) constructed from the query. Matching

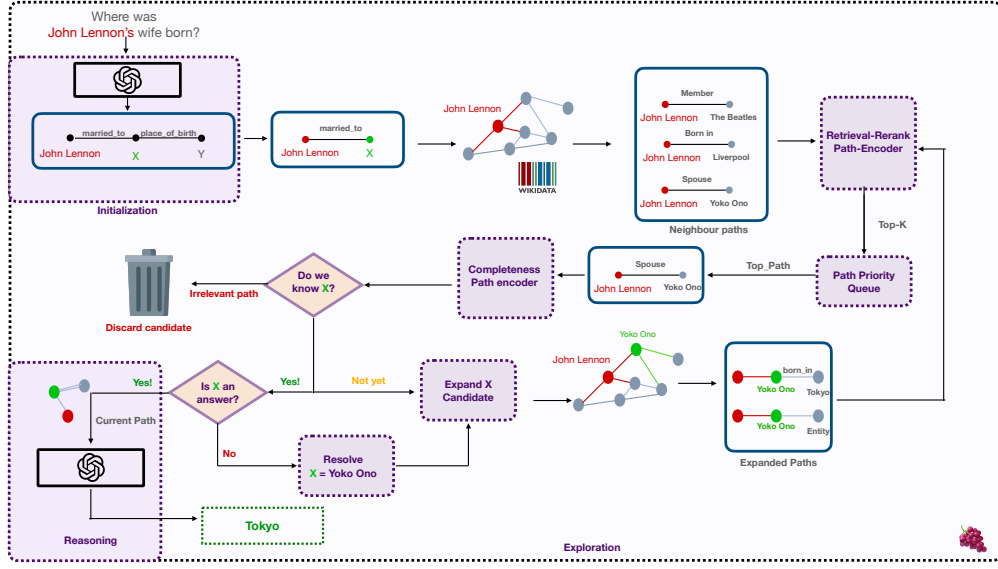


Figure 1: **GRAPE pipeline.** (1) *Initialization*: convert the query to an Anonymous Knowledge Graph (AKG) and link known entities. (2) *Exploration*: encoder-guided fuzzy path matching with relation clustering, a fixed-size beam, and completeness labels $\{S, C, D\}$ (Stop/Continue/Drop). (3) *Reasoning*: a single LLM call aggregates top candidates and their supporting KG paths into final answers.

proceeds by iteratively disambiguating the unknown nodes in the AKG, evaluating path completeness through a dedicated encoder, and pruning candidates via top-N semantic similarity. Once the anonymous entities have been resolved, a single final call to the LLM is used to generate the answer. In line with prior work, the framework is organized into three components: initialization, exploration, and reasoning.

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 defining the graph in the form (head, relation, tail). A path in $\mathcal{G}$ is defined as

$$P := [(e_0, r_0, e_1), (e_1, r_1, e_2), \dots, (e_{k-1}, r_k, e_k)] \in \mathcal{P}(\mathcal{G}),$$

where $\mathcal{P}(\mathcal{G})$ denotes the set of all valid paths in the graph.

Formally, a system for knowledge graph question answering is a mapping from a query in natural language to a subset of supported evidence (i.e., an answer set):

$$\begin{aligned} f : \mathcal{L} \times \mathcal{G} &\longrightarrow \mathcal{P}(\mathcal{E}) \\ (q, \mathcal{G}) &\longmapsto \mathcal{A}_q := \{e \in \mathcal{E} \mid \exists P = [(e_0, r_1, e_1), \dots, (e_{k-1}, r_k, e_k)] \in \mathcal{P}(\mathcal{G}), \\ &\quad P \models q \wedge e = e_k\}, \end{aligned}$$

where $\mathcal{L} := \langle \mathbb{V} \rangle$ is the language set generated from a vocabulary $\mathbb{V}$ and the relation $A \models B$ indicates that the structure A satisfies the conditions expressed by B .

Modern LLM-based systems often decompose the initial query q into a sequence of subqueries (a resolution plan) $(q_0, \dots, q_N)$. The resolution process then follows the recursion

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

where $f(\cdot, \cdot; H)$ denotes a call to the system with additional context H (in this case, the previous answer set), and the recursion terminates with $f_k = \mathcal{A}_q$.

3.2 INITIALIZATION

Using an LLM to extract a resolution plan is an easy and flexible way to determine an action plan from the initially given query. However, this makes a retrieval model entirely dependent on the LLM during the exploration process, as decomposing the query necessarily produces sub-queries that lose the global structure of the original intent. This, in turn, requires repeated calls to the LLM to maintain coherence, which increases inference cost and introduces additional opportunities for hallucination.

By contrast, our approach reformulates the query as an *Anonymous Knowledge Graph (AKG)*, preserving its overall structure while allowing unknown entities to remain as variables to be resolved (see Figure 1). For example, the query “*Director of a movie in which Tom Hanks acted*” is translated into the AKG

$$\text{Tom Hanks} \xrightarrow{\text{acted.in}} X_1 \xrightarrow{\text{directed.by}} X_2,$$

where X_1 denotes an anonymous variable corresponding to an unknown movie and X_2 an anonymous variable representing the director to be returned as the answer.

Formally, for a given query q we define its associated *Anonymous Knowledge Graph (AKG)* as

$$\hat{\mathcal{G}}_q = (E_q, R_q, T_q),$$

where

$$E_q = E \cup \hat{E}, \quad T_q = T \cup \hat{T}_1 \cup \hat{T}_2.$$

Here E denotes the set of entities explicitly mentioned in the query, $\hat{E}$ the set of anonymous variables introduced during query parsing, and R_q the set of relations extracted from q . The set T contains triplets composed only of known entities, $\hat{T}_1$ the triplets involving one anonymous variable, and $\hat{T}_2$ the triplets involving two anonymous variables.

Instead of LLM planned query decomposition, the resolution process can be expressed as

$$\hat{\mathcal{G}}_q^{(k)} = \begin{cases} \hat{\mathcal{G}}_q, & k = 0, \\ \text{resolve}(\hat{\mathcal{G}}_q^{(k-1)}, X_i \mapsto e^*), & k \geq 1, \end{cases}$$

$$e^* = \arg \max_{e \in \mathcal{E}} s(X_i \mid \hat{\mathcal{G}}_q^{(k-1)}, e, \mathcal{G}), \quad (1)$$

where at each step a single anonymous variable $X_i \in \hat{E}$ is substituted with a candidate entity $e^* \in \mathcal{E}$ from the knowledge graph $\mathcal{G}$. The scoring function $s(\cdot)$ evaluates the compatibility of e with the current state of the AKG $\hat{\mathcal{G}}_q^{(k-1)}$ and the evidence in $\mathcal{G}$. The operator $\text{resolve}(\hat{\mathcal{G}}_q^{(k-1)}, X_i \mapsto e^*)$ then updates the graph by replacing X_i with e^* in all affected triplets, yielding the new state $\hat{\mathcal{G}}_q^{(k)}$. The process continues until all anonymous variables are resolved, resulting in the fully instantiated query graph $\hat{\mathcal{G}}_q^{(*)}$.

In practice, we generate the initial AKG graph $\mathcal{G}_q$ by using a few shots example query to an LLM (in our case GPT-4o OpenAI et al. (2024)) that can be found in Section A.1. We also follow ToG Sun et al. (2023) technique to map all non-anonymous entities to node candidates in the KG.

3.3 EXPLORATION

Exploration is framed as a graph matching problem between the AKG $\hat{\mathcal{G}}_q$ and the KG $\mathcal{G}$. At each iteration, we attempt to resolve one anonymous variable $X \in \hat{E}$ by aligning the set of AKG triplets that involve it,

$$T_X := \{ (h, r, t) \in T^{(k)} \mid h = X \vee t = X \} \subseteq T^{(k)},$$

with candidate continuations drawn from $\mathcal{G}$. Unlike exact matching, the process is *fuzzy*: a relation $r \in R_q$ in the AKG does not need to correspond to an identical edge in $\mathcal{G}$, but can be approximated by semantically similar relations or by a short path.

Formally, let the query graph at iteration k be

$$\hat{\mathcal{G}}_q^{(k)} = (E \cup \hat{E}, R_q, T^{(k)}),$$

where E are bound entities, $\hat{E}$ are still-unresolved variables, and $T^{(k)}$ is the current set of triplets (some involving anonymous nodes). We maintain a beam $\mathcal{F}_k$ of partial paths with fixed beam size b and a maximum expansion depth $D_{\max}$ (in our experiments $b = 10$, $D_{\max} = 3$) as possible candidates to align T_X . The exploration process can then be decomposed into the following phases:

Candidate generation. Given the frontier entity e_m of a path $P \in \mathcal{F}_k$ in the beam, we enumerate its one-hop neighborhood $\mathcal{N}(e_m) = \{(e_m, r, v)\} \in \mathcal{T}$. Each neighbor induces a candidate continuation $P' = P \oplus (e_m, r, v)$. We will expand the relations for every path in the beam sequentially and according to its similarity score.

Scoring. Each candidate extension P' is scored with two components:

1. A *retrieve-rerank* score $s^{\text{rank}}(T_{X_k}, q, (e_m, r, v) \mid P)$, computed in two stages: first, a bi-encoder retriever (paraphrase-MiniLM-L3-v2 Reimers & Gurevych (2019)) evaluates the semantic similarity between the query and a plain-text representation of the path $P' = P \oplus (e_m, r, v)$; second, a cross-encoder reranker (fine-tuned from bge-reranker-v2-m3 Chen et al. (2024)) compares the local neighborhood of the anonymous entity X_k (i.e., its incident triplets T_{X_k}) with the candidate continuation P' and outputs a score in $[0, 1]$ depending on semantic similarity of both sets.
2. A *completeness* score $h_\theta(T_{X_k}, P') \in \{S, C, D\}$, predicted by a multiclass classifier fine-tuned from DeBERTa-v3-Large He et al. (2021), where S denotes *Stop* (path already satisfies T_{X_k}), C denotes *Continue* (expand further), and D denotes *Drop* (irrelevant). Only paths predicted as S or C are retained.

Therefore, the scoring function in eq. (1) can be instantiated as:

$$s(X \mid \hat{\mathcal{G}}_q^{(k)}, e_m, \mathcal{G}) = \mathbb{1}[h_\theta(T_{X_k}, P') = S] + \mathbb{1}[h_\theta(T_{X_k}, P') = C] s^{\text{rank}}(T_{X_k}, q, (e_m, r, v) \mid P)$$

Where $\mathbb{1}(\cdot)$ denotes the indicator function. A detailed explanation of the training of these models can be found in Section A.2.

Clustering and representatives. To control high fan-out, expanded neighbors $\mathcal{N}(e_m)$ are clustered by relation, $\mathcal{N}(e_m) = \bigcup_r \mathcal{C}_r$, with $\mathcal{C}_r = \{(e_m, r, v) : v \in \mathcal{N}(e_m)\}$. From each cluster, we select a representative

$$v_r^* = \arg \max_{(r, v) \in \mathcal{C}_r} s^{\text{rank}}(q, (e_m, r, v) \mid P),$$

forming the representative set $\mathcal{R}(e_m) = \{(e_m, r, v_r^*)\}_r$. Clusters are only decompressed when selected, which reduces redundant scoring.

Beam update and backtracking. During disambiguation of T_{X_k} , after generating and scoring new candidates, we keep the beam size fixed at b by retaining the top- b paths and discarding the rest. The next path to expand is the highest-scoring one in the beam. If further expansion of that path becomes unpromising (its score degrades), we continue with the next best path in the current beam, allowing the search to switch branches without additional machinery.

Termination. A path stops when $h_\theta(q, P) = S$ or $\text{depth}(P) = D_{\max}$. The surviving paths $\mathcal{F}_K$ define the candidate answer set by their terminal entities.

Method	WebQSP	Adv HotpotQA	QALD- 10-en	FEVER	CREAK	Zero- Shot RE	CWQ	Simple Questions	Web Questions	T-REx
ToG-2 (GPT-3.5)	81.1	42.9	54.1	63.1 [†]	93.5	91.0	—	—	—	—
ToG (GPT-4)	76.2	26.3	50.2	52.7	93.8	88.0	57.1	53.6	54.5	76.8
PoG-E (GPT-4)	95.4	—	—	—	—	—	78.5	81.2	82.0	—
PoG (GPT-4)	96.7	—	—	—	—	—	81.4	84.0	84.6	—
IO prompt (ChatGPT)	63.3	—	42.0	—	89.7	27.7	37.6	20.0	48.7	33.6
CoT (ChatGPT)	62.2	—	42.9	—	90.1	28.8	38.8	20.3	48.5	32.0
CoK (6-shot)	—	35.4	—	58.5	—	—	—	—	—	—
RoG	85.7	—	—	—	—	—	62.6	—	—	—
RRKG	91.5	—	—	—	—	—	68.7	—	—	—
GRAPE (ours)	94.74 ± 3.87	60.31 ± 1.39	65.67 ± 3.84	79.12 ± 4.04	92.77 ± 3.74	94.20 ± 2.79	80.15 ± 2.17	82.87 ± 2.07	88.15 ± 4.87	94.86 ± 3.57

Table 1: Best per column in **bold**. Metrics: Hits@1 for WebQSP, AdvHotpotQA, QALD-10-en, Zero-Shot RE, CWQ, SimpleQuestions, WebQuestions, T-REx; Accuracy for FEVER and CREAK. GRAPE results are mean $\pm$ std over 5 runs. [†] FEVER for ToG-2 reported with 3-shots.

3.4 REASONING

We perform a single aggregation step with an LLM GPT-4o; prompt in the Section A.1). The top candidate entities and their supporting KG paths are provided to the model together with the original query and the AKG. The LLM selects the best answer(s) when multiple candidates remain (i.e., no path has been labeled *S* by the completeness classifier).

Because we retain the KG relations used to reach each disambiguated entity (the logical path schema), we subsequently issue a SQL query over the KG store to retrieve any additional entities that instantiate the same path pattern, thereby covering multi-answer queries. The final answer set is the union of the LLM selection and the matched entities, deduplicated.

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 GRAPE on ten datasets spanning KGQA, open-domain QA, slot filling, and fact checking. For KGQA we use four multi-hop sets—CWQ Talmor & Berant (2018), WebQSP Yih et al. (2016), QALD-10 Perevalov et al. (2022), and AdvHotpotQA Pan et al. (2024)—and one single-hop set, SimpleQA Bordes et al. (2015). All KGQA experiments are executed over the Wikidata Vrandečić & Krötzsch (2014) knowledge graph. For open-domain QA we use WebQuestions Berant et al. (2013) for slot filling, T-REx El Sahar 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 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 over the full benchmark (over one month) and report mean $\pm$ standard deviation. Ablations examine (i) incremental addition of retrieval components starting from a vanilla LLM-only retriever, (ii) the effect of beam width and depth, and (iii) different LLM backbones and the completeness cross-encoder. Base configuration details are provided in the Section A.4.

4.2 MAIN RESULTS

4.3 COMPARISON WITH PREVIOUS WORK

In this experiment we compare Hits@1 for question-answer datasets and accuracy for claim-verification datasets. In Table 1 we can observe that GRAPE pushes performance on slot-filling benchmarks: on T-REx we obtain 94.86, clearly above all baselines in the table, and on Zero-Shot RE 94.20, slightly surpassing ToG-2 (91.0). Because relations in these datasets are generally shallow by construction, they rarely require deep reasoning; in practice, GRAPE’s completeness mechanism tends to stop early (often at one hop), keeping the reasoning burden on the LLM small (see Appendix). By contrast, AdvHotpotQA and QALD-10-en are more knowledge-

System	LLM calls (avg.)	Exec. time / query (avg., s)
ToG	16.3	69.3
ToG-2	5.4	27.3
PoG	7.8	34.6
CoK	11.0	30.1
GRAPE (ours)	2.4	10.4

Table 2: Average LLM calls and average execution time per query. Lower is better.

intensive and frequently need multi-hop evidence. Here, GRAPE significantly outperforms alternative approaches—60.31 vs 42.9 on AdvHotpotQA and 65.67 vs 54.1 on QALD-10-en (both vs ToG-2)—which we attribute to the combination of plan-aligned exploration, relation-wise clustering with deferred decompression, and a learned completeness judge that halts expansion once sufficient evidence is assembled. For fact verification, GRAPE is essentially on par with ToG on CREAK (92.77 vs 93.8) but substantially better on FEVER (79.12 vs 52.7 for ToG and 58.5 for CoK). We note that CREAK is a binary (true/false) setting that can often be resolved with shallow Wikidata evidence—both systems readily reach the first-level connections and, using a similar reasoning LLM, end up close. FEVER, in contrast, is a three-way decision where many ToG errors fall into the Not Enough Information class; GRAPE naturally mitigates this by passing multiple candidate paths to the final reasoning step when completeness has not yet marked a query as solved. On the remaining datasets GRAPE is broadly on par in accuracy; crucially, as shown next, it achieves this while considerably reducing execution time and the number of LLM calls.

4.4 INFERENCE COST

A key practical aspect is inference cost. By design, GRAPE keeps LLM usage small: in the worst case it makes $D_{\max} + 2$ calls—at most one per depth level for seed disambiguation, plus one for planning and one for final reasoning—independent of the beam width b . This is well below agent-style methods whose calls grow with both b and $D_{\max}$, e.g., $2bD_{\max} + D_{\max} + 1$ for ToG and approximately $3D_{\max}$ for ToG-2. Empirically (Table 2), GRAPE reduces average LLM calls by 85.3% versus ToG (2.4 vs. 16.3), 55.6% versus ToG-2 (2.4 vs. 5.4), 69.2% versus PoG (2.4 vs. 7.8), and 78.2% versus CoK (2.4 vs. 11.0). The same trend holds for latency: average execution time per query drops by 85.0% versus ToG (10.4s vs. 69.3s), 61.9% versus ToG-2 (10.4s vs. 27.3s), 69.9% versus PoG (10.4s vs. 34.6s), and 65.4% versus CoK (10.4s vs. 30.1s). These savings stem from the encoder-driven exploration and the completeness judge controlling overexpansion.

4.5 ABLATION STUDY ON EXPLORATION COMPONENTS

We analyze the exploration stack by toggling four modules while keeping the rest of the pipeline fixed (planner/final-answer LLM, Wikidata KG, $b=10$, $D_{\max}=3$). *Expansion* is always present and is not ablated.

Components.

1. *Bi-encoder retriever*. Shortlists candidate continuations $P' = P \oplus (e_m, r, v)$ by semantic similarity (query/path text) before fine scoring.
2. *Cross-encoder reranker*. Computes $s^{\text{rank}}(T_{X_k}, q, (e_m, r, v) \mid P)$ by comparing the incident triplets T_{X_k} to the candidate P' .
3. *Completeness classifier*. Predicts $h_{\theta}(T_{X_k}, P') \in \{\text{STOP}, \text{CONTINUE}, \text{DROP}\}$ to control halt/expand/drop decisions.
4. *Relation-wise clustering (deferred decompression)*. Groups $\mathcal{N}(e_m)$ by relation; selects a representative per relation and only decompresses a cluster if selected.

Ablation protocol (substitutions when a module is off). If the *bi-encoder* is off, we prefilter with BM25. If the *cross-encoder* is off, the LLM returns a scalar score for (T_{X_k}, P') used for ranking. If the *completeness classifier* is off, the LLM outputs one of $\{\text{STOP}, \text{CONTINUE}, \text{DROP}\}$. If *clustering*

Variant	Bi-enc. retr.	Cross-enc. rerank	h_θ	Rel. clustering	Hits@1/Acc (%)	LLM calls (avg)	Time / query (s)
Baseline (LLM rank + LLM comp.)	N	N	N	N	70.2	55.0	120.9
V1	Y	N	N	N	73.0	50.5	90.2
V2	Y	Y	N	N	78.5	14.8	38.6
V3	Y	Y	Y	N	80.6	3.2	16.5
V4 (full)	Y	Y	Y	Y	81.8	2.4	10.4

Table 3: Ablation over exploration components. Baseline: plain expansion with LLM-based ranking and completeness (no encoders, no clustering). V1: + bi-encoder shortlist. V2: + cross-encoder s^{rank} (LLM no longer ranks). V3: + h_θ (exploration decisions no longer call the LLM; only plan/answer/seed tie-breakers remain). V4: + relation-wise clustering (full system).

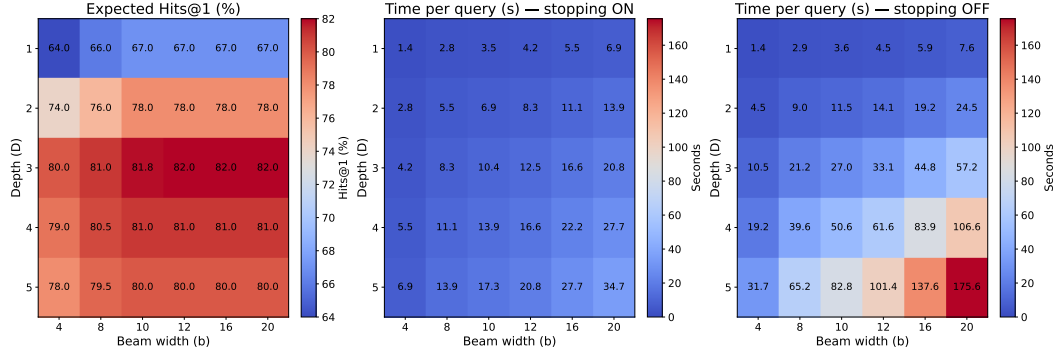


Figure 2: Beam-depth sweep. **Left:** Expected Hits@1 (%). **Middle:** Time/query (s) with h_θ ON. **Right:** Time/query (s) with h_θ OFF (shared color scale with the middle panel). Accuracy saturates around $b \approx 10$ and $D \approx 3$; without h_θ , runtime grows sharply with depth due to the absence of early stopping and pruning.

is off, we expand all neighbors (optionally asking the LLM to choose a small set of relation types first).

As we can appreciate in Table 3, the LLM-native baseline is costly because the LLM both ranks many neighbors and decides completeness at each hop. Adding the bi-encoder (V1) reduces fan-out and latency but accuracy remains limited because ranking is still LLM-based. The cross-encoder (V2) is the inflection point: ranking moves off the LLM, average calls drop sharply and accuracy rises as promising candidates are promoted. Introducing h_θ (V3) removes LLM calls from exploration decisions, cutting calls to roughly three per query and further reducing time. Finally, relation-wise clustering (V4) eliminates redundant scoring on high-fan-out relations, yielding the lowest latency (10.4 s) with 2.4 average LLM calls.

4.6 WIDTH AND DEPTH VARIATIONS IN BEAM SEARCH

We study how beam width and maximum depth affect performance and inference time during exploration. We sweep beam widths $b \in \{4, 8, 10, 12, 16, 20\}$ and depths $D \in \{1, 2, 3, 4, 5\}$, keeping the rest of the pipeline fixed (Wikidata KG; planner/final-answer LLM; encoder stack). For each (b, D) we report the average Hits@1 and the average execution time per query. We also report the time when the completeness module h_θ is deactivated (no early stopping / path dropping).

As shown in Figure 2, widening the beam helps up to $b \approx 10$ and then plateaus; increasing depth beyond $D \approx 3$ yields little gain and can slightly degrade accuracy (due to longer, noisier paths). With h_θ active, time grows roughly with $b \cdot D$ (encoder work), consistent with linear scaling in width and depth. When h_θ is disabled, time increases dramatically with depth (and mildly with width), reflecting the lack of early stopping and aggressive pruning. This highlights h_θ 's role in containing search and keeping latency low.

Backbone	Avg. Hits@1 (QA)	Avg. Acc. (Fact)	LLM calls (avg.)	Time/query (s)
GPT-4o (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 4: Backbone comparison with GRAPE ($b=10$, $D=3$). 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).

4.7 BACKBONE SENSITIVITY

Our main GRAPE results (Hits@1) use GPT-4o as the reasoning backbone. To assess backbone sensitivity, we swap only the planner/final-answer LLM for comparably capable models, while keeping the encoder stack (bi-/cross-encoder), h_θ , relation clustering, and Wikidata fixed ($b=10$, $D=3$). We report averages over QA tasks (Hits@1) and fact checking (Acc.), together with average LLM calls and time per query. Because exploration is encoder-driven, LLM calls remain essentially unchanged across backbones; latency varies modestly with each model’s generation speed.

GRAPE’s performance is largely robust to the backbone because retrieval, path scoring, and stopping are encoder-driven. GPT-4o yields the strongest QA average (82.6) and fact accuracy (85.9); smaller or instruction-tuned alternatives trail by 1–3 pp on QA and 1–3 pp on fact checking. Average LLM calls remain ≈ 2.4 across backbones, while latency shifts modestly with model speed (e.g., slightly faster with 4o-mini, slightly slower for larger open models). Overall, backbone choice nudges final precision, but GRAPE’s efficiency and most of its accuracy gains come from the exploration stack rather than the reasoning LLM.

5 LIMITATIONS AND FUTURE WORK

GRAPE is a KGQA-centric system: unlike hybrid RAG approaches (e.g., ToG-2), integrating non-graph evidence (free text, tables, web) is non-trivial and would require additional alignment modules. The system is also sensitive to entity linking errors and KG coverage/staleness, which can yield premature stops or spurious paths, and to distribution shift in the trained components (bi-/cross-encoders and h_θ), which may necessitate re-tuning. Finally, answer verbalization remains LLM-based and can introduce occasional aggregation or normalization mistakes. At present, the order of anonymous-entity disambiguation is seeded by the LLM’s initial plan; a promising direction is to combine this with graph-derived statistics (e.g., node uncertainty or expected information gain) to prioritize easier decisions and prune earlier, thereby reducing end-to-end latency.

6 CONCLUSION

In this paper we introduced GRAPE, a KGQA framework that shifts retrieval from LLM-directed search to encoder-guided fuzzy graph matching with explicit stopping. In evaluation, GRAPE reduces both LLM calls and end-to-end latency by 60–80%, improves accuracy on 6 of 10 datasets, and remains competitive on the remainder. Our key insight is that effective multi-hop reasoning over KGs requires LLMs for plan induction and answer synthesis—not for steering the search. Consequently, this encoder-driven design offers a practical path to faster, more scalable KGQA without sacrificing accuracy.

REFERENCES

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. Can knowledge graphs reduce hallucinations in LLMs? : A survey. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the As-*

- sociation for Computational Linguistics: *Human Language Technologies (Volume 1: Long Papers)*, pp. 3947–3960, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.219. URL <https://aclanthology.org/2024.naacl-long.219/>.
- Meta AI. The LLaMA 3 model card and technical report. <https://ai.meta.com/research/publications/the-llama-3-model-card-and-technical-report/>, 2024a. Accessed: 2025-06-23.
- Mistral AI. Mistral large. <https://mistral.ai/news/mistral-large/>, 2024b. Accessed: 2025-06-23.
- Anthropic. Introducing claude 3.5 sonnet. <https://www.anthropic.com/news/claude-3-5-sonnet>, 2024. Accessed: 2025-06-23.
- 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. Non-determinism of "deterministic" llm settings, 2025. URL <https://arxiv.org/abs/2408.04667>.
- Jinheon Baek, Alham Fikri Aji, and Amir Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In Estevam Hruschka, Tom Mitchell, Sajjadur Rahman, Dunja Mladenić, and Marko Grobelnik (eds.), *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*, pp. 70–98, Toronto, ON, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.matching-1.7. URL <https://aclanthology.org/2023.matching-1.7/>.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL <https://aclanthology.org/D13-1160/>.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. 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, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i16.29720. URL <https://doi.org/10.1609/aaai.v38i16.29720>.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks, 2015. URL <https://arxiv.org/abs/1506.02075>.
- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL <https://aclanthology.org/2024.findings-acl.137/>.
- Lingjiao Chen, Matei A. Zaharia, and James Y. Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *ArXiv*, abs/2305.05176, 2023. URL <https://api.semanticscholar.org/CorpusID:258564349>.
- Hady Elsahar, Pavlos Vougiouklis, Arslan Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. T-REx: A large scale alignment of natural language with knowledge base triples. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, H  l  ne

- Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL <https://aclanthology.org/L18-1544/>.
- Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning, 2023. URL <https://arxiv.org/abs/2210.00720>.
- Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *ArXiv*, abs/2111.09543, 2021. URL <https://api.semanticscholar.org/CorpusID:244346093>.
- Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. A survey of knowledge enhanced pre-trained language models. *IEEE Trans. on Knowl. and Data Eng.*, 36(4):1413–1430, April 2024. ISSN 1041-4347. doi: 10.1109/TKDE.2023.3310002. URL <https://doi.org/10.1109/TKDE.2023.3310002>.
- Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2), January 2025. ISSN 1046-8188. doi: 10.1145/3703155. URL <https://doi.org/10.1145/3703155>.
- Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.389. URL <https://aclanthology.org/2024.naacl-long.389/>.
- Che Jiang, Biqing Qi, Xiangyu Hong, Dayuan Fu, Yang Cheng, Fandong Meng, Mo Yu, Bowen Zhou, and Jie Zhou. On large language models’ hallucination with regard to known facts. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1041–1053, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.60. URL <https://aclanthology.org/2024.naacl-long.60/>.
- Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. Graph chain-of-thought: Augmenting large language models by reasoning on graphs, 2024. URL <https://arxiv.org/abs/2404.07103>.
- Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. KG-GPT: A general framework for reasoning on knowledge graphs using large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9410–9421, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.631. URL <https://aclanthology.org/2023.findings-emnlp.631/>.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 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, 2022. Curran Associates Inc. ISBN 9781713871088.
- 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. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

- Tianle Li, Xueguang Ma, Alex Zhuang, Yu Gu, Yu Su, and Wenhua Chen. Few-shot in-context learning on knowledge base question answering. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6966–6980, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.385. URL <https://aclanthology.org/2023.acl-long.385/>.
- Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq R. Joty, Soujanya Poria, and Lidong Bing. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. In *International Conference on Learning Representations*, 2023b. URL <https://api.semanticscholar.org/CorpusID:263610099>.
- Pasquale Lops, Antonio Silletti, Marco Polignano, Cataldo Musto, and Giovanni Semeraro. Reproducibility of llm-based recommender systems: the case study of p5 paradigm. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys ’24, pp. 116–125, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705052. doi: 10.1145/3640457.3688072. URL <https://doi.org/10.1145/3640457.3688072>.
- Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, 2024.
- Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *International Conference on Learning Representations*, 2024. URL <https://api.semanticscholar.org/CorpusID:273185432>.
- Alan Malek, Jiawei Ge, Nevena Lazic, Chi Jin, András György, and Csaba Szepesvári. Frontier llms still struggle with simple reasoning tasks, 2025. URL <https://arxiv.org/abs/2507.07313>.
- Agada Joseph Oche, Ademola Glory Folashade, Tirthankar Ghosal, and Arpan Biswas. A systematic review of key retrieval-augmented generation (rag) systems: Progress, gaps, and future directions, 2025. URL <https://arxiv.org/abs/2507.18910>.
- Yasumasa Onoe, Michael J. Q. Zhang, Eunsol Choi, and Greg Durrett. Creak: A dataset for commonsense reasoning over entity knowledge, 2021. URL <https://arxiv.org/abs/2109.01653>.
- OpenAI. Gpt-3.5 models. <https://platform.openai.com/docs/models/gpt-3-5>, 2023. 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, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel

- Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kopic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeesh Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL <https://arxiv.org/abs/2303.08774>.
- Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Trans. on Knowl. and Data Eng.*, 36(7):3580–3599, July 2024. ISSN 1041-4347. doi: 10.1109/TKDE.2024.3352100. URL <https://doi.org/10.1109/TKDE.2024.3352100>.
- Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 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)*, pp. 229–234, 2022. doi: 10.1109/ICSC52841.2022.00045.
- Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 43–54, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1005. URL <https://aclanthology.org/D19-1005/>.
- 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. KILT: a benchmark for knowledge intensive language tasks. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2523–2544, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.200. URL <https://aclanthology.org/2021.naacl-main.200/>.
- Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of*

- the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL <https://aclanthology.org/D19-1410/>.
- Mobashir Sadat, Zhengyu Zhou, Lukas Lange, Jun Araki, Arsalan Gundroo, Bingqing Wang, Rakesh Menon, Md Parvez, and Zhe Feng. DelucionQA: Detecting hallucinations in domain-specific question answering. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 822–835, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.59. URL <https://aclanthology.org/2023.findings-emnlp.59/>.
- Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Fast or better? balancing accuracy and cost in retrieval-augmented generation with flexible user control, 2025. URL <https://arxiv.org/abs/2502.12145>.
- Ying Su, Jipeng Zhang, Yangqiu Song, and Tong Zhang. PipeNet: Question answering with semantic pruning over knowledge graphs. In Danushka Bollegala and Vered Shwartz (eds.), *Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024)*, pp. 360–371, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.starsem-1.29. URL <https://aclanthology.org/2024.starsem-1.29/>.
- Jiashuo Sun, Chengjin Xu, Luminyuan Tang, Sai Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, 2023. URL <https://api.semanticscholar.org/CorpusID:263333907>.
- Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1059. URL <https://aclanthology.org/N18-1059/>.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Tamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL <https://aclanthology.org/N19-1421/>.
- Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. Paths-over-graph: Knowledge graph empowered large language model reasoning. In *Proceedings of the ACM on Web Conference 2025*, pp. 3505–3522, 2025a.
- Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. Hydrarag: Structured cross-source enhanced large language model reasoning, 2025b. URL <https://arxiv.org/abs/2505.17464>.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL <https://aclanthology.org/N18-1074/>.
- Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014. doi: 10.1145/2629489.
- Sebastian Walter and Hannah Bast. Grasp: Generic reasoning and sparql generation across knowledge graphs, 2025. URL <https://arxiv.org/abs/2507.08107>.

- Kai Wang, Yuwei Xu, Zhiyong Wu, and Siqiang Luo. LLM as prompter: Low-resource inductive reasoning on arbitrary knowledge graphs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3742–3759, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.224. URL <https://aclanthology.org/2024.findings-acl.224/>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 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, 2022. Curran Associates Inc. ISBN 9781713871088.
- Yilin Wen, Zifeng Wang, and Jimeng Sun. MindMap: Knowledge graph prompting sparks graph of thoughts in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10370–10388, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.558. URL <https://aclanthology.org/2024.acl-long.558/>.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023a. URL <https://arxiv.org/abs/2305.10601>.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023b. URL <https://arxiv.org/abs/2210.03629>.
- Wentao Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Annual Meeting of the Association for Computational Linguistics*, 2016. URL <https://api.semanticscholar.org/CorpusID:13905064>.
- Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. Poolingformer: Long document modeling with pooling attention, 2022a. URL <https://arxiv.org/abs/2105.04371>.
- Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models, 2022b. URL <https://arxiv.org/abs/2210.03493>.

A ADDITIONAL IMPLEMENTATION DETAILS

A.1 PROMPTS

Prompt for generating the anonymous graph:

```
We need to make triplets to make a query from a given text, where the
unknown entities or verbs to match are represented by an X_i.
The idea is to substitute an entity that is unknown by X_i, this entities
might be parsed later using a triplet extracted from a knowledge
graph.
Try to use relations as verbs and entities as nouns, although is not
mandatory if the relations are complex.
Take as anchor entity the one that is more restrictive, i.e. the one that
is more specific.
There should always be at least X_1 but there might be more than one
entity to match, so be as fine-grained creating entities as possible,
the more the better and careful with modifiers like dates or numbers
Note that we should consider every sentence as positive as our goal is
just to generate a query for checking information. Therefore ignore
negations and quantifiers like never, not, only, exclusivly, always
...
Here are a few examples:
```

```

810 - "John Wick is a movie starting the american actor Keenau Reeves and
811     directed by Stephen Spielberg" -> [(John Wick, starring, X_1), (X_1,
812     country_of_origin, United States), (John Wick, directed_by, X_2)].
813 - "Barack Obama is married with a lady who has a kid that has blue eyes"
814     -> [(Barack Obama, married, X1), (X1, has_children, X2), (X2,
815     has_eye_color, blue)].
816 - "Berlin is the capital city of Germany" -> [(Berlin, is_capital_of,
817     Germany)].
818 - "Paul McCartney has never been in a band" -> [(X_1, is, band), (X_1,
819     is_member, Paul McCartney)].
820 - "John Doe had dinner with Barack Obama" -> [(John Doe, X_i, Barack
821     Obama)].
822 - "Duke Leto Attreides is a character in the Dune poem" -> [(Duke Leto
823     Attreides, is_character_in, X_1), (X_1, is, poem)].
824 - "The movie Titanic was directed by a Canadian man." -> [(X_1,
825     director_of, Titanic), (X_1, is, man)].
826 - "The movie Titanic starred an actress." -> [(X_1, starring, Titanic), (
827     X_1, is, actress)].
828 - "Don McLean starred with a British actress and a German actor in a 2016
829     movie." -> [(Don McLean, starring, X_1), (X_1, release_date, 2016),
830     (X_2, starring, X1), (X_2, nationality, British), (X_2, profession,
831     actress), (X_3 starring, X_1), (X_3, profession, actor). (X_3,
832     nationality, German)].
833 - "The movie Breakfast at Tiffany's featured a song that was not composed
834     by Henry Mancini." -> [(Breakfast at Tiffany's, featured, X_1), (X_1
835     , is, song), (X_1, composed_by, X_2), (X_2, is_not, Henry Mancini)].
836 - "Which TV show Barack Obama appeared in?" -> [(Barack Obama,
837     appeared_in, X_1), (X_1, is, TV show)].
838 - "The actor who played Jack Ryan in the movie The Hunt for Red October
839     (1969) is from the same country as Barack Obama" -> [(X_1,
840     interpreted, Jack_Ryan), (Jack_Ryan, character_of,
841     The_Hunt_of_the_Red_October), (The_Hunt_Of_The_Red_October,
842     release_date, 1969), (X_1, nationality, X_2), (Barack_Obama,
843     nationality, X_2)].
844 - "The director of the movie Move (2010) is from the same country as the
845     director of the movie The Dark Knight" -> [(X_1, director_of, Move),
846     (Move, release_date, 2010), (X_1, nationality, X_2), (X_3,
847     director_of, The_Dark_Knight), (X_3, nationality, X_2)].
848 - "The film The Dark Knight was released in the same year as the film The
849     Hunt for Red October" -> [(The_Dark_Knight, release_date, X_1), (
850     The_Hunt_Of_The_Red_October, release_date, X_1)].
851 - "Who is the paternal grandfather of John Doe?" -> [(X_1, father_of,
852     John_Doe), (X2, father_of, X_1)]. (Note here the grandfather relation
853     can be splitted in two)
854 - "Who is the maternal grandfather of John Doe?" -> [(X_1, mother_of,
855     John_Doe), (X2, father_of, X_1)].
856 - "From which universities are the Canadian recipients of the Turing
857     Award that work in the field of Deep Learning?" -> [(X_1, nationality
858     , Canadian), (X_1, received, Turing_Award), (X_1, field_of_work,
859     Deep_Learning), (X_1, studied_at, X_2), (X_2, is, university)].
860
861 Parenthesis near known entities usually give some information about the
862 entity, like the year of a movie, profession of some person, a
863 clarification of what the entity is, or place of an event, like The
    Hunt for Red October (1969) -> [(The_Hunt_Of_The_Red_October,
    release_date, 1969)].
    Note that for example in the last ones even if we matched X_1 we would
    still need to check that X_1 is a TV_show.
    However, it's also extremely important not to create dummy entities like (
    X_1, name, Duke Leto Atreides) or (X_1, is, Duke Leto Atreides) as
    this is not useful for querying.
    Return a list (i.e [triplet, triplet, triplet ....]) of triplets
    representing the following text, keep it short and simple but be as

```

specific as possible in the entities, and *be careful not to drop information about the X_i entities*

Prompt for reasoning:

You are given:

1. A query expressed as triplets with anonymous entities (X_i).
2. The target entity to resolve (e.g., X_1).
3. A list of candidate answers for the target entity.
4. A set of logical paths extracted from a knowledge graph.

Your task:

- Identify which candidate answer best resolves the target entity.
- Replace the anonymous entities (X_i) in the query triplets with the resolved entities from the logical path.
- If no candidate matches, answer "None".

Example:

Target entity: X_1

Query triplets:

$[(X_1, \text{director_of}, \text{Titanic}), (X_1, \text{nationality}, \text{Canadian}), (\text{Titanic}, \text{release_date}, 1997)]$

Candidates:

- James Cameron
- Steven Spielberg
- Denis Villeneuve

Logical paths:

- James Cameron director_of Titanic James Cameron nationality Canadian Titanic release_date 1997
- Steven Spielberg director_of Titanic Steven Spielberg nationality American Titanic release_date 1997
- Denis Villeneuve director_of Arrival Denis Villeneuve nationality Canadian Arrival release_date 2016

Answer:

James Cameron

A.2 TRAINING THE ANONYMOUS-GRAPH ENCODER

Goal. We train an encoder to score the compatibility between an *anonymous query subgraph* and a *candidate KG path/subgraph*, so that fuzzy matches (synonyms, variable-length alignments, type-consistent substitutions) receive high scores while incomplete or corrupted matches receive low scores.

Supervision source. From the train/dev splits of all KGQA benchmarks, we convert each query q into its AKG $\hat{q} = (E_q, R_q, T_q)$ and identify the subset $T_X \subseteq T_q$ of triplets incident to a chosen anonymous variable X (the variable to resolve at this step). When gold supporting paths are available, we use them; otherwise, we mine minimal satisfying paths in the KG that instantiate X and make the query true.

Positive examples. Starting from (T_X, P^*) pairs (where P^* is a supporting path), we create *hard positives* via: (i) **synonymic rewrites** of relations/entities (from a curated synonym list) that preserve truth conditions; (ii) **neutral triplet insertion**—add edges touching nodes in T_X but not changing the answer set (e.g., types/attributes); (iii) **query refinements**—add logically implied constraints (e.g., *instance_of*(X , *Film*)). All such augmentations keep the label positive.

Algorithm 1 Anonymous-Graph Training Pair Generation

Require: Datasets $\mathcal{D}$, KG $\mathcal{G}$, synonym maps SynRel, SynEnt, topology patterns Topo, augmentation budget K

Ensure: Training set $\mathcal{S} = \{(T_X, P, y)\}$

```

1:  $\mathcal{S} \leftarrow \emptyset$ 
2: for all  $(q, \text{answers}) \in \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{dev}}$  do
3:    $\hat{\mathcal{G}}_q = (E_q, R_q, T_q) \leftarrow \text{AnonGraph}(q)$ 
4:   for all  $X \in \text{AnonymousVars}(\hat{\mathcal{G}}_q)$  do
5:      $T_X \leftarrow \{(h, r, t) \in T_q \mid h = X \vee t = X\}$  ▷ constraints incident to  $X$ 
6:      $\mathcal{P}^* \leftarrow \text{MineSupportingPaths}(T_X, \mathcal{G})$  ▷ use gold if available
7:     for all  $P^* \in \mathcal{P}^*$  do
8:        $\mathcal{S} \leftarrow \mathcal{S} \cup \{(T_X, P^*, 1)\}$  ▷ base positive
9:       for  $k = 1$  to  $K$  do ▷ augment positives
10:         $T_X^+ \leftarrow \text{SynonymicRewrite}(T_X, \text{SynRel}, \text{SynEnt})$ 
11:         $T_X^+ \leftarrow \text{InsertNeutralTriplets}(T_X^+, \mathcal{G})$ 
12:         $T_X^+ \leftarrow \text{AddImpliedConstraints}(T_X^+)$ 
13:         $P^+ \leftarrow \text{TopologyJitter}(P^*, \text{Topo})$  ▷ equivalent detours
14:         $\mathcal{S} \leftarrow \mathcal{S} \cup \{(T_X^+, P^+, 1)\}$ 
15:      end for
16:      for  $k = 1$  to  $K$  do ▷ generate hard negatives
17:         $T_X^-, P^- \leftarrow \text{CorruptOne}(T_X, P^*; \text{drop-edge} \vee \text{bad-syn} \vee \text{target-swap} \vee$ 
18:        topology-perturb)
19:        if  $\neg \text{Satisfies}(T_X^-, P^-)$  then
20:           $\mathcal{S} \leftarrow \mathcal{S} \cup \{(T_X^-, P^-, 0)\}$ 
21:        end if
22:      end for
23:    end for
24:  end for
25: return  $\mathcal{S}$ 

```

Negative examples. We generate *hard negatives* by minimally breaking support: (i) **edge dropout**—remove a key hop in P^* or a required constraint in T_X ; (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 perturbations**—re-route or re-order hops to produce misleading chains. These produce pairs $(T_X, \tilde{P})$ that *look* similar but do not satisfy T_X .

Order-invariance (sets, not sequences). To learn over *sets of triplets*, we randomly permute the triplets in T_X at sampling time and re-linearize them (and optionally shuffle non-critical hops in the candidate) so the encoder does not overfit to serialization order.

Input formatting and objective. We serialize (T_X, P) as two text fields (or graph tokens) and feed them to a cross-encoder to obtain a score $s(T_X, P) \in \mathbb{R}$. We train with binary labels $y \in \{0, 1\}$ using a logistic loss:

$$\mathcal{L} = -y \log \sigma(s(T_X, P)) - (1 - y) \log(1 - \sigma(s(T_X, P))),$$

or, optionally, a pairwise margin loss for ranking positives above negatives sampled for the same T_X .

Topology diversity. To avoid overfitting to chains, we synthesize examples spanning chains, forks, diamonds, and short detours (1–3 hops), ensuring the encoder sees multiple structures that can satisfy the same T_X .

Algorithm 2 Batch sampling with set invariance**Require:** Training set $\mathcal{S}$, permutations per item M

```

1: for each minibatch do
2:   Sample  $(T_X, P, y)$  items from  $\mathcal{S}$ 
3:   for all items in batch do
4:      $T_X^{(\pi)} \leftarrow \text{RandomPermutation}(T_X)$ 
5:      $P^{(\pi)} \leftarrow \text{OptionalShuffle}(P)$  ▷ shuffle non-critical hops
6:     Encode pair  $(T_X^{(\pi)}, P^{(\pi)})$  and update via logistic (or margin) loss
7:   end for
8: end for

```

A.3 HARDWARE SPECIFICATIONS

All experiments were performed in a machine with the technical capabilities reported in section A.3.

CPU	AMD Ryzen Threadripper 3975WX
RAM	256 GB
Cores	64
GPU	2x Nvidia A100 160GB

Table 5: Specifications of the machine in which the experiments were executed.

A.4 TRAINING HYPERPARAMETERS

The full table of hyperparameters used in the training of the path cross encoder systems and inference for LLMs can be found in table 6. Different options for the setting tried of the system appear between curly braces, while the selected ones appear in bold.

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	$\{\mathbf{Cosine Annealing}, \text{Linear}\}$
Weight Decay	0.01
Maximum Gradient Norm	$\mathbf{1}, 5, 10$
Loss Function	Cross-Entropy with logits
Max Tokens	512
D_{max} (maximum expansion depth)	3
b (beam size)	10
top-p	1
temperature	0.0
max_generated_tokens	2048

Table 6: Training hyperparameters for the proposed system. Between curly braces are all values tested during optimization, the one selected are marked in bold.

A.5 KG EXPLORATION PSEUDOCODE

Algorithm 3 Anonymous KG Exploration with Relation Clustering

Require: Query q ; initial $\hat{G}^0$; beam size $b=10$; depth limit $D_{\max}$

- 1: Initialize frontier $\mathcal{F}^0 \leftarrow \{\emptyset\}$, accepted set $\mathcal{A} \leftarrow \emptyset$, priority queue $Q \leftarrow \emptyset$
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: **for each** $P \in \mathcal{F}^k$ **do**
- 4: Select anonymous target $\hat{x} \in \hat{E}$ referenced by the next planned triplet $\tau^* = (\hat{x}, r^*, o^*)$
- 5: Let e be the current binding for $\hat{x}$ reached along P
- 6: Enumerate $\mathcal{N}(e) = \{(r, v)\}$ and cluster: $\mathcal{N}(e) = \bigsqcup_r \mathcal{C}_r$
- 7: **for each** cluster $\mathcal{C}_r$ **do**
- 8: $v_r^* \leftarrow \arg \max_{(r,v) \in \mathcal{C}_r} s_{\text{rank}}(q, (e, r, v) \mid P)$
- 9: **end for**
- 10: $\mathcal{R} \leftarrow \{(e, r, v_r^*)\}_r$
- 11: Candidate continuations $\mathcal{C} \leftarrow \{P \oplus (e, r, v_r^*) : (e, r, v_r^*) \in \mathcal{R}\}$
- 12: $\tilde{\mathcal{F}} \leftarrow \text{TOPB}(\mathcal{C}, b \text{ by } s_{\text{rank}})$
- 13: **for each** $P' \in \tilde{\mathcal{F}}$ **do**
- 14: $y \leftarrow h_\theta(q, P') \in \{\text{COMPLETE}, \text{EXPAND}, \text{IRRELEVANT}\}$
- 15: **if** $y = \text{IRRELEVANT}$ **then**
- 16: **continue** ▷ drop
- 17: **else if** $y = \text{COMPLETE}$ **or** $\text{depth}(P') = D_{\max}$ **then**
- 18: $\mathcal{A} \leftarrow \mathcal{A} \cup \{P'\}$ ▷ accept and stop expanding
- 19: **else**
- 20: Push P' into Q with key $u(P')$ ▷ mark for expansion
- 21: **end if**
- 22: **end for**
- 23: **end for**
- 24: $\mathcal{F}^{k+1} \leftarrow \emptyset$
- 25: **while** $|\mathcal{F}^{k+1}| < b$ **and** $Q \neq \emptyset$ **do**
- 26: $P^* \leftarrow \text{POPMAX}(Q)$
- 27: **if** P^* chose representative (e, r, v_r^*) for expansion **then**
- 28: *// cluster decompression on demand*
- 29: Insert $P^* \oplus (e, r, v)$ for top- k members $(r, v) \in \mathcal{C}_r$ by s_{rank}
- 30: $\mathcal{F}^{k+1} \leftarrow \mathcal{F}^{k+1} \cup \{P^*\}$ ▷ advance beam; enables backtracking if needed
- 31: **end while**
- 32: **if** $Q = \emptyset$ **then**
- 33: **break**
- 34: **end if**
- 35: **end for**
- 36: **return** $\mathcal{A}$
