SPARK: UNIFYING STRUCTURE-AWARE PATH-PREFIX-CONDITIONED TREE SEARCH AND GRAPH-GROUNDED GENERATION FOR KNOWLEDGE GRAPH QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge Graph Question Answering requires retrieving compact and coherent multi-hop evidence from large-scale knowledge graphs and reasoning over it to produce accurate answers. Existing methods either invoke a large language model (LLM) repeatedly during graph exploration or use lightweight retrievers whose local decisions need not remain consistent with the full reasoning prefix. Furthermore, serializing graph evidence into verbose natural language weakens structural information and increases context length. We propose **SPARK**, which couples prefix-conditioned evidence retrieval with a graph-grounded generator. Its retriever factorizes each expansion into relation and entity decisions, scores each action jointly with the question and complete prefix, and labels an action as positive only when the extended trajectory remains a prefix of an answer path. For generation, we operationalize Knowledge Graph Language (KGL) as a question-specific interface that compactly encodes retrieved paths using a typed symbol dictionary and ordered symbol sequences; projected entity and relation representations replace the corresponding KGL-symbol embeddings in the LLM input. The two-stage collaborative training strategy further coordinates retrieval and generation through a shared graph encoder. Finally, extensive experiments on WebQSP and CWQ confirm that SPARK achieves significant performance gains compared to the state-of-the-art (SOTA) baselines.

1 INTRODUCTION

Knowledge Graph Question Answering (KGQA) aims to answer natural-language questions by reasoning over structured facts in knowledge graphs (KGs) Pan et al. (2024). Despite extensive research, a fundamental challenge remains Sun et al. (2018); Mavromatis & Karypis (2022): given a large-scale KG containing vast amounts of irrelevant information, how can a system identify complete, compact, and coherent multi-hop evidence under a limited search budget? Recent advances in large language models (LLMs) have opened new avenues for KGQA Zhu et al. (2024), owing to their strong capabilities in language understanding and complex reasoning Achiam et al. (2023); Team (2025). Yet, the parametric knowledge of LLMs is hard to update promptly Li et al. (2025b), and LLMs may produce factually unsupported answers in knowledge-intensive tasks Huang et al. (2025).

Inspired by Retrieval-Augmented Generation (RAG) Fan et al. (2024), a growing body of methods has embraced a "*retrieve-then-reason*" paradigm, which first retrieves question-relevant triples, paths, or subgraphs from the KG and then feeds them into an LLM for answer inference. Depending on whether the LLM serves as the primary controller of the retrieval stage, existing efforts broadly fall into two categories: 1) Treating the LLM as a retrieval agent that progressively performs relation and entity selection, evidence-sufficiency assessment, and reasoning-path planning using its parametric knowledge Liang & Gu (2025); Li et al. (2025a); Ma et al. (2025a); Polonuer et al. (2026); Xu et al. (2026a); Shan et al. (2026). Such methods exhibit strong semantic understanding and flexible exploration capabilities, yet they typically require multiple retrieval rounds and frequent LLM calls, incurring substantial inference latency and computational overhead. 2) Adopting lightweight models Li et al. (2025c); Jiang et al. (2025); Yu et al. (2026); Park et al. (2026); Liu et al. (2026)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

or specialized tree search algorithms Shen et al. (2025); Wang et al. (2025); Gao et al. (2025b) for candidate selection, path scoring, and branch pruning. These methods enable efficient retrieval, but many score nodes, triples, or completed paths without explicitly conditioning every expansion action on both the question and the complete path prefix. Consequently, an edge that is locally plausible can be selected even when it is inconsistent with the trajectory that reached the current state.

Beyond evidence retrieval, effectively conveying graph structure to the LLM is equally critical to answer quality. The two categories of methods discussed above commonly serialize triples or reasoning paths into plain text and concatenate them into the LLM's input context; however, this process tends to weaken the graph connectivity. As the volume of evidence grows, the verbose textual descriptions further increase context length and obscure salient information. To alleviate this, some studies harness graph encoders or knowledge adapters that project structural representations into the LLM's embedding space He et al. (2024); Ao et al. (2025); Gao et al. (2025a); Fu et al. (2026). While validating the efficacy of continuous graph representations, they predominantly focus on representations at the level of entire subgraphs, paths, or triples, and still fall short of simultaneously preserving symbolic semantics, path ordering, and fine-grained entity-relation structures within a compact input representation. See Appendix A for more related work.

To this end, we propose **SPARK**, a **S**tructure-aware **P**ath-Prefix-conditioned tree se**A**rch and g**R**aph-grounded generation framework for **K**GQA. Concretely, we introduce a structure-aware path encoder that jointly models the question semantics, the complete path prefix, and the current candidate action, enabling the retriever to perform context-consistent expansions conditioned on the existing reasoning trajectory. Relation selection first determines the next reasoning direction, followed by entity selection that pinpoints the specific graph branch, factorizing the vast triple-level action space into two semantically well-defined hierarchical decision stages. Furthermore, drawing inspiration from Knowledge Graph Language (KGL) Guo et al. (2024); Luo et al. (2025a), we instantiate KGL for KGQA as a typed, question-specific symbol dictionary and an ordered serialization of retrieved paths. Each entity or relation description appears once in the dictionary, while its symbol can be reused along multiple paths; at the same symbol position, a projected graph representation replaces the ordinary token embedding. This interface retains explicit path order and fine-grained entity–relation identity while avoiding repeated surface descriptions by construction. SPARK then uses a two-stage strategy: it first learns the search policy and subsequently optimizes retrieval and generation losses together through a shared graph encoder, while enabling representation-level coordination between them through a shared graph encoder.

To sum up, we highlight our contributions as follows: *First*, we propose a structure-aware, path-prefix-conditioned tree search, which decomposes multi-hop evidence retrieval into hierarchical relation-entity expansion and enables coherent and effective graph exploration via prefix-consistent supervision. *Second*, we develop a graph-aware KGL interface tailored to KGQA. It injects entity and relation representations at the corresponding symbol positions, allowing the LLM to jointly exploit symbolic semantics, reasoning paths, and fine-grained graph structures within a compact input. *Third*, we devise a two-stage training strategy that promotes representation-level coordination between retrieval and generation by a shared graph encoder while stating the optimization boundary explicitly. *At last*, experiments on WebQSP and CWQ show that SPARK's performance is highly competitive with the SOTA baselines.

2 PRELIMINARIES

Task Definition. A KG is represented as $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where each triple $(e_h, r, e_t) \in \mathcal{G}$ denotes a fact, with $e_h, e_t \in \mathcal{E}$ being the head and tail entities, respectively, and $r \in \mathcal{R}$ denoting the relationship between them. Here, $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively. Given a natural-language question q, KGQA systems commonly follow the "*retrieve-then-reason*" paradigm. A retriever f_θ extracts a question-relevant subgraph $\mathcal{G}_{\text{sub}} \subseteq \mathcal{G}$, while an answer generator f_ϕ predicts the answer entity set $\mathcal{A}_q \subseteq \mathcal{E}$ conditioned on q and $\mathcal{G}_{\text{sub}}$.

Path-Conditioned Search Space. We assume that entity linking has identified a set of seed entities $\mathcal{S}_q \subseteq \mathcal{E}$ for the question q Yih et al. (2015). Let e_v denote the current entity, and let the path prefix from a seed entity $e_0 \in \mathcal{S}_q$ to e_v be $\chi_v = (e_0, r_1, e_1, \ldots, r_{L_v}, e_v)$, where L_v is the number of hops in χ_v. Given a search state $o = (e_v, \chi_v)$, let $\mathcal{R}_o$ denote the set of admissible relations for expansion.

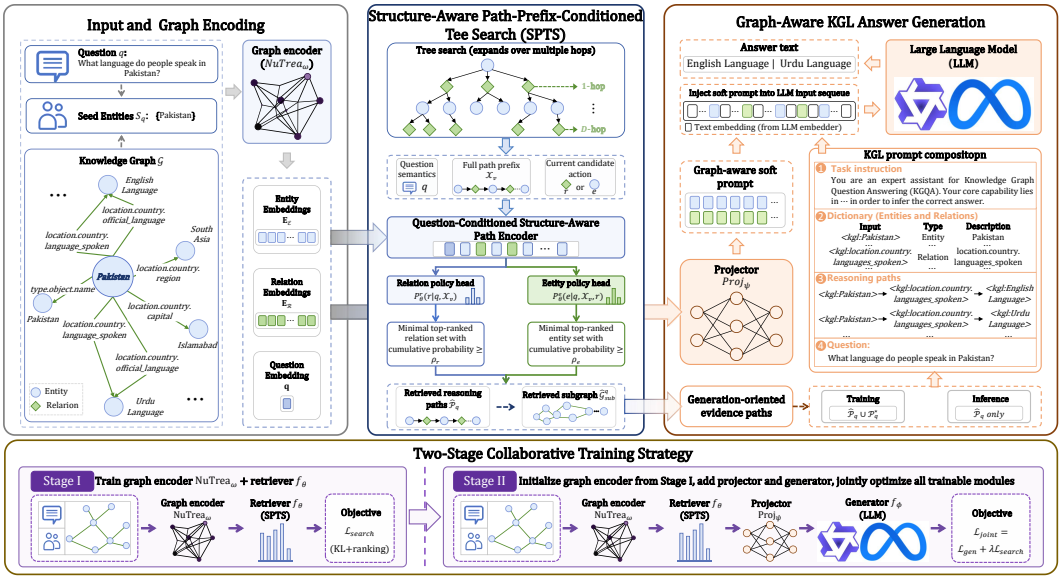

Figure 1: The overall process of the proposed SPARK method, including *Input and Graph Encoding*, *Structure-Aware Path-Prefix-Conditioned Tree Search (SPTS)*, *Graph-Aware KGL Answer Generation*, and *Two-Stage Collaborative Training strategy*.

For each relation $r \in \mathcal{R}_o$, let $\mathcal{N}_o^r$ be the corresponding set of candidate neighboring entities. Thus, the candidate triples at state o are $\mathcal{T}(e_v, \chi_v) = \{(e_v, r, e) | r \in \mathcal{R}_o,\ e \in \mathcal{N}_o^r\}$.

Question-Conditioned Graph Representations. In KGs, the number of entity nodes is typically much larger than that of relation types, and entities often appear as proper nouns with low information density; they may even be anonymized or encrypted for privacy protection Bollacker et al. (2008); Ning et al. (2026). To this end, we adopt NuTrea Choi et al. (2023) as the graph encoder, which explicitly incorporates the question semantics into the learned graph representation. Given q, $\mathcal{G}$, and $\mathcal{S}_q$, the encoding process is formulated as:

$$\mathbf{q}, \mathbf{E}_\mathcal{E}, \mathbf{E}_\mathcal{R} = \mathrm{NuTrea}_\omega(q, \mathcal{G}; \mathcal{S}_q), \tag{1}$$

where ω denotes the trainable parameters, $\mathbf{q} \in \mathbb{R}^{d_n}$ is the question representation, and $\mathbf{E}_\mathcal{E} \in \mathbb{R}^{|\mathcal{E}| \times d_n}$ and $\mathbf{E}_\mathcal{R} \in \mathbb{R}^{|\mathcal{R}| \times d_n}$ are the entity and relation representation matrices, respectively; d_n is the output dimension. Detailed computational procedures are provided in the Appendix B.

Knowledge Graph Language. Knowledge Graph Language (KGL), introduced in MKGL Guo et al. (2024), represents entities and relations as atomic symbols and expresses each KG fact as an entity–relation–entity sequence. While MKGL focuses on knowledge graph completion, SPARK extends KGL to KGQA by organizing retrieved multi-hop evidence into compact, ordered symbol sequences that preserve path connectivity and support graph-grounded answer generation (see Appendix C).

3 METHOD

This section details SPARK that comprises *structure-aware path-prefix-conditioned tree search*, *graph-aware KGL answer generation*, and *two-stage collaborative training strategy*. The overall framework of SPARK is illustrated in Fig. 1.

3.1 STRUCTURE-AWARE PATH-PREFIX-CONDITIONED TREE SEARCH (SPTS)

SPTS frames multi-hop evidence retrieval as a layer-wise tree search rooted at the seed entities in $\mathcal{S}_q$.

Hierarchical Candidate Action Representation. For a search state o, each one-step expansion is factorized into two consecutive decisions: selecting a relation $r \in \mathcal{R}_o$ and then selecting a next-hop entity $e \in \mathcal{N}_o^r$. Correspondingly, the relation and entity candidate sequences are respectively defined as: $\chi_v^r = \chi_v \oplus (r)$, $\chi_v^{r,e} = \chi_v \oplus (r, e)$, where $\oplus$ denotes sequence concatenation. These two sequences

correspond to a relation-selection action and an entity-selection action, respectively. Both retain the complete path prefix, allowing each candidate action to be evaluated in the context of the preceding reasoning trajectory.

Question-Conditioned Structure-Aware Path Encoding. To model the above candidate actions, we introduce a lightweight Transformer-based path encoder f_{enc}. Given a candidate sequence $\chi \in \{\chi_v^r, \chi_v^{r,e}\}$, we first map its entities and relations into a sequence of graph representations $[\mathbf{x}_1, \ldots, \mathbf{x}_{|\chi|}] \in \mathbb{R}^{|\chi| \times d_n}$ according to Eq. (1). Since the path sequence contains both sequential order and entity-relation type distinctions, we augment each position i with a learnable positional embedding $\mathbf{p}_i \in \mathbb{R}^{d_n}$ and a type embedding $\mathbf{u}_{\tau_i} \in \mathbb{R}^{d_n}$, where $\tau_i \in \{\text{ent}, \text{rel}\}$. We then prepend $\mathbf{q}$ as a conditioning token and feed the resulting sequence into f_{enc}:

$$\mathbf{H}_\chi = f_{\text{enc}}\big([\mathbf{q} + \mathbf{p}_0, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_i, \ldots, \hat{\mathbf{x}}_{|\chi|}]\big), \tag{2}$$

where $\hat{\mathbf{x}}_i = \mathbf{x}_i + \mathbf{p}_i + \mathbf{u}_{\tau_i}$. Through the self-attention mechanism, f_{enc} establishes global interactions among the question semantics, the preceding search trajectory, and the current candidate action, yielding a question-conditioned path context representation. In this way, we use the hidden state of the final action token as the candidate representation, denoted by $\mathbf{h}_r$ for χ_v^r and $\mathbf{h}_e$ for $\chi_v^{r,e}$.

Hierarchical Relation-Entity Search Policy. At each search step, the relation selection determines the semantic direction of subsequent reasoning, whereas the entity selection specifies the concrete graph branch. We therefore invoke two task-specific policy heads on top of the shared path encoder. The scores and probabilities for a candidate relation r and a candidate entity e are defined as:

$$s_o(r) = \text{MLP}_r(\mathbf{h}_r),\ p_\theta^r(r|q,o) = \text{SM}_{\mathcal{R}_o}(s_o(r)), \tag{3}$$

$$s_o^r(e) = \text{MLP}_e(\mathbf{h}_e),\ p_\theta^e(e|q,o,r) = \text{SM}_{\mathcal{N}_o^r}(s_o^r(e)), \tag{4}$$

where $\text{SM}_{\mathcal{C}}(s(k)) = \frac{\exp(s(k))}{\sum_{k' \in \mathcal{C}} \exp(s(k'))}$. Thus, the joint probability of selecting the expansion triple (e_v, r, e) is factorized as $p_\theta(r, e|q, o) = p_\theta^r(r|q,o) p_\theta^e(e|q,o,r)$. This factorization avoids directly modeling a large, flat triple-level action space and follows the natural KG expansion process of first selecting a relation and then a neighboring entity.

Path-Prefix Supervision and Policy Optimization. During training, we construct search supervision using answer entities. For each $a \in \mathcal{A}_q$, we search for shortest paths starting from the seed set $\mathcal{S}_q$ with a hop length not exceeding the maximum depth D, and collect all such paths to form the set $\mathcal{P}_q^\star$. We let $\mathcal{O}_q^\star = \big\{(e_v, \chi_v) \;\big|\; \exists \chi^\star \in \mathcal{P}_q^\star,\ \chi_v \prec \chi^\star\big\}$ be the set of non-terminal prefix states induced by $\mathcal{P}_q^\star$, where $\prec$ denotes "is a proper prefix of". The induced supervision subgraph $\mathcal{G}_q^\star$ is defined as the union of all triples appearing in these paths. Simply labeling an edge as positive whenever it appears in $\mathcal{G}_q^\star$ can introduce erroneous supervision across different answer paths or search branches; instead, we require that the expanded sequence remains a prefix of at least one supervision path. Specifically, for the state o, the label of a candidate r is $y_o(r) = \mathbb{I}\big[\exists \chi^\star \in \mathcal{P}_q^\star,\ \chi_v \oplus (r) \preceq \chi^\star\big]$, where $\preceq$ denotes "is a prefix of". Given a selected relation r, the label of a candidate entity e is $y_o^r(e) = \mathbb{I}\big[\exists \chi^\star \in \mathcal{P}_q^\star,\ \chi_v \oplus (r, e) \preceq \chi^\star\big]$. Because a state may admit multiple valid relation or entity expansions, we normalize the binary labels to obtain target distributions, i.e., $\widetilde{y}_o(r) = y_o(r) / \sum_{r' \in \mathcal{R}_o} y_o(r')$ and $\widetilde{y}_o^r(e) = y_o^r(e) / \sum_{e' \in \mathcal{N}_o^r} y_o^r(e')$. We then encourage the predicted policies to match these target distributions using Kullback–Leibler (KL) divergence:

$$\mathcal{L}_{\text{KL}}^o = \text{KL}\left(\widetilde{\mathbf{y}}_o \parallel \mathbf{p}_o\right),\ \mathcal{L}_{\text{KL}}^{o,r} = \text{KL}\left(\widetilde{\mathbf{y}}_o^r \parallel \mathbf{p}_o^r\right), \tag{5}$$

where $\widetilde{\mathbf{y}}_o = [\widetilde{y}_o(r)]_{r \in \mathcal{R}_o}$, $\widetilde{\mathbf{y}}_o^r = [\widetilde{y}_o^r(e)]_{e \in \mathcal{N}_o^r}$, $\mathbf{p}_o = [p_\theta^r(r|q,o)]_{r \in \mathcal{R}_o}$ and $\mathbf{p}_o^r = [p_\theta^e(e|q,o,r)]_{e \in \mathcal{N}_o^r}$.

Although KL divergence aligns the predicted and supervised probability distributions, it may not enforce a sufficiently large score margin between positive and negative actions, particularly when the candidate set is large or highly imbalanced. We therefore introduce an auxiliary pairwise ranking objective. Let $\mathcal{C}_o^\pm$ denote the positive/negative relation candidates and $\mathcal{C}_o^{r,\pm}$ the corresponding entity candidates conditioned on r. We define the pairwise ranking loss as

$$\mathcal{L}_{\text{rank}}^o = -\frac{1}{|\mathcal{C}_o^+||\mathcal{C}_o^-|} \sum_{i \in \mathcal{C}_o^+} \sum_{j \in \mathcal{C}_o^-} \log \sigma(\delta_r^{ij}), \mathcal{L}_{\text{rank}}^{o,r} = -\frac{1}{|\mathcal{C}_o^{r,+}||\mathcal{C}_o^{r,-}|} \sum_{i \in \mathcal{C}_o^{r,+}} \sum_{j \in \mathcal{C}_o^{r,-}} \log \sigma(\delta_e^{ij}), \tag{6}$$

where $\delta_r^{ij} = s_o(i) - s_o(j)$ and $\delta_e^{ij} = s_o^r(i) - s_o^r(j)$. We omit a ranking term if either candidate set is empty. Finally, the overall retrieval objective for q is

$$\mathcal{L}_{\text{search}}(q) = \frac{1}{|\mathcal{O}_q^\star|} \sum_{o \in \mathcal{O}_q^\star} \left(\mathcal{L}_{\text{KL}}^o + \beta_r \mathcal{L}_{\text{rank}}^o + \sum_{r \in \mathcal{R}_o^+} \widetilde{y}_o(r) \left(\mathcal{L}_{\text{KL}}^{o,r} + \beta_e \mathcal{L}_{\text{rank}}^{o,r} \right) \right), \tag{7}$$

where $\mathcal{R}_o^+ = \{r \in \mathcal{R}_o \mid y_o(r) = 1\}$, and β_r, β_e control the weights of the ranking constraints.

Tree Search during Inference. During inference, SPTS performs tree search using only q, $\mathcal{S}_q$, and $\mathcal{G}$. The search starts from each seed entity and expands up to D layers. In a state o, we first sort candidate relations in descending order according to p_θ^r, and retain the smallest top-ranked relation set whose cumulative probability reaches the threshold ρ_r. For each retained relation, we analogously select entities using p_θ^e and threshold ρ_e. Note that $\rho_r, \rho_e \in (0, 1]$. The selected relation-entity pairs instantiate child nodes for subsequent expansion. The final retrieved path set and the subgraph it induces are represented by $\widehat{\mathcal{P}}_q$ and $\widehat{\mathcal{G}}_{\text{sub}}^q$, respectively. Note that D controls the search depth, while ρ_r and ρ_e control the search width at the relation and entity levels; larger threshold values generally lead to more search branches.

3.2 Graph-Aware KGL Answer Generation

The paths retrieved by SPTS contain candidate answer entities while retaining the ordered reasoning trajectories that connect them to the seed entities. To fully exploit this information, we represent the retrieved paths in KGL and inject graph representations of entities and relations into the LLM via a soft prompting mechanism.

Generation-Oriented Evidence Path Construction. During training, the retriever may not yet fully retrieve the answer evidence, causing the LLM to learn answer mappings from incomplete reasoning paths. To stabilize generator training, we augment the retrieved paths with the supervision paths during training, while using only retrieved paths at inference time, that is, $\mathcal{P}_{\text{gen}}^q = \widehat{\mathcal{P}}_q \cup \mathcal{P}_q^\star$ for training phase; otherwise $\mathcal{P}_{\text{gen}}^q = \widehat{\mathcal{P}}_q$. Note that this augmentation creates a deliberate train–test evidence mismatch. We therefore treat its effect as an empirical question and compare it with retrieved-path-only training in Section 4.3.

Graph-Aware KGL Prompt Construction. Let $\mathcal{Z}_q$ be the set of entities and relations appearing in $\mathcal{P}_{\text{gen}}^q$. For any $z \in \mathcal{Z}_q$, we denote its KGL symbol by $\kappa(z)$, e.g., the entity *Pakistan* corresponds to `<kgl:Pakistan>`, and the relation *location.country.official_language* corresponds to `<kgl:location.country.official_language>`. As shown in Fig. 1, the generation prompt consists of a task instruction, a dictionary of KGL symbols, reasoning paths, and the question. The symbol dictionary is defined as $\mathcal{D}_q = \big\{(\kappa(z), \text{type}(z), \text{desc}(z)) | z \in \mathcal{Z}_q\big\}$, where $\text{type}(z) \in \{\text{entity}, \text{relation}\}$, and $\text{desc}(z)$ is the canonical textual description of the entity or relation. Meanwhile, each path $\chi = (e_0, r_1, e_1, \ldots, r_{L_\chi}, e_{L_\chi}) \in \mathcal{P}_{\text{gen}}^q$ with $L_\chi \leq D$ is linearized as $\kappa(e_0) \to \kappa(r_1) \to \kappa(e_1) \to \ldots \to \kappa(r_{L_\chi}) \to \kappa(e_{L_\chi})$. See Appendix C for a concrete prompt example.

To further inject graph structural information, each KGL symbol is represented by a single placeholder token position, establishing a one-to-one correspondence between the symbol and its graph-aware embedding. For $\kappa(z)$, its corresponding graph representation is $\mathbf{g}_z = \mathbf{E}_{\mathcal{E}}[z]$ if $z \in \mathcal{E}$; otherwise $\mathbf{g}_z = \mathbf{E}_{\mathcal{R}}[z]$. Since the output dimension d_n of NuTrea generally differs from the LLM embedding dimension d_{llm}, we introduce a lightweight projector Proj_ψ to map the graph representation into the LLM embedding space: $\mathbf{m}_z = \text{Proj}_\psi(\mathbf{g}_z) \in \mathbb{R}^{d_{\text{llm}}}$. Furthermore, let τ^q denote the hybrid sequence formed by concatenating the task instruction, dictionary, reasoning paths, and question. Ordinary text tokens are encoded using the original LLM token-embedding layer, whereas each KGL symbol position is replaced with its corresponding projected graph embedding:

$$\mathbf{x}_i^{\text{llm}} = \begin{cases} \text{Emb}_{\text{llm}}(\tau_i^q), & \tau_i^q \text{ is a text token}, \\ \mathbf{m}_z, & \tau_i^q = \kappa(z). \end{cases} \tag{8}$$

Thus, the resulting input embedding sequence for the LLM is $\mathbf{X}_{\text{llm}}^q = [\mathbf{x}_1^{\text{llm}}, \ldots, \mathbf{x}_{|\tau^q|}^{\text{llm}}]$. This design combines dictionary-grounded symbols with continuous graph representations. Because a description is written once in the dictionary and its symbol is reused across paths, KGL removes repeated surface descriptions by construction.

Autoregressive Training Objective. Given $\mathbf{X}_{\text{llm}}^q$, the generator f_ϕ produces the answer text autoregressively. Since the answer $\mathcal{A}_q$ to question q is inherently an unordered set of entities, different permutations may yield semantically equivalent but textually distinct target sequences. Therefore, we first sort the answer entities according to a predefined deterministic rule, and then join them with

a fixed separator to form a canonical answer text. For example, the answer in Fig. 1 is represented as "English Language|Urdu Language". The resulting canonical answer string is tokenized into the target token sequence $a_q = (a_{q,1}, \ldots, a_{q,T_q})$, where T_q is the length of a_q. Under teacher forcing, the generator predicts the t-th target token conditioned on the ground-truth history tokens $a_{q,<t}$. The generation loss is

$$\mathcal{L}_{\text{gen}}(q) = -\sum_{t=1}^{T_q} \log p_\phi(a_{q,t}|a_{q,<t}, \mathbf{X}^q_{\text{llm}}) \,. \tag{9}$$

By minimizing this objective, the generator learns to jointly leverage the natural-language question, the KGL-encoded path evidence, and the associated graph-aware soft prompts to generate a canonical answer text consistent with the target answer-entity set.

3.3 Two-Stage Collaborative Training Strategy

To improve training stability, SPARK adopts a two-stage collaborative training strategy. In the first stage, the graph encoder NuTrea_ω and the retriever f_θ are jointly optimized via $\mathcal{L}_{\text{search}}(q)$ to learn a stable tree search policy. In the second stage, the graph encoder and the retriever are initialized with the parameters obtained from the first stage, after which the generator f_ϕ and the KGL projector Proj_ψ are introduced. The complete model is then jointly optimized using the following objective:

$$\mathcal{L}_{\text{joint}}(q) = \mathcal{L}_{\text{gen}}\big(q; \omega, \psi, \phi\big) + \lambda \mathcal{L}_{\text{search}}(q; \omega, \theta), \tag{10}$$

where λ controls the relative weight of the retrieval objective. Accordingly, θ is optimized by the retrieval objective, ϕ and ψ by the generation objective, and the shared graph encoder parameters ω by both objectives. This promotes representation-level coordination between retrieval and generation while preserving their respective optimization objectives.

4 Experiments

4.1 Experimental Settings

Datasets and Baselines. We conduct experiments on the widely used KGQA datasets, WebQSP Yih et al. (2016) and CWQ Talmor & Berant (2018). For evaluation, we compare our retrieval method SPTS against existing baselines including G-Retriever He et al. (2024), GNN-RAG Mavromatis & Karypis (2025), SubgraphRAG Li et al. (2025c), and QSRAG Xu et al. (2026b), and compare our KGQA method SPARK against strong baselines, which we group into four categories (*GNN Reasoning*, *LLM Reasoning*, *KG+LLM Reasoning*, and *GNN/Lightweight Retriever+LLM Reasoning*) detailed in Table 2. For SPARK, we employ multiple open-source LLMs of varying scales as generators, specifically Llama3.1-8B-Instruct Grattafiori et al. (2024), Llama3.2-1B/3B-Instruct [1], and Qwen3-1.7B/4B/8B Team (2025).

Metrics and Configurations. Following prior work Li et al. (2025c); Xu et al. (2026b), we evaluate performance in terms of retrieval and reasoning. For retrieval, we adopt Triple Recall (TR) and Answer Recall (AR) as evaluation metrics. The former measures the proportion of triples in the retrieval results that lie on the shortest paths from seed entities to answer entities; the latter measures the proportion of answer entities covered by the subgraph induced by the retrieved triples. For reasoning (KGQA), we use Hits@1 and F1 as evaluation metrics. Hits@1 denotes the proportion of questions for which the correct answer is ranked first in the output; F1 combines precision and recall, measuring both accuracy and coverage. Throughout all experiments, unless otherwise specified, the hyperparameters are set as follows: for WebQSP and CWQ, we set $\beta = \beta_r = \beta_e = 0.3$ and 0.5, respectively, with $\rho = \rho_r = \rho_e = 0.95$ and $\lambda = 0.1$ for both datasets. The output dimension of f_{enc} is $d_n = 256$. Due to space limitations, the complete experimental settings are provided in Appendix D.

4.2 Comparison Analysis

[1] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct, https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Table 2: KGQA results comparison of different methods on WebQSP and CWQ. Best results are in **bold**. Second best results are indicated with underlines.

Methods	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
GNN Reasoning				
ReaRev Mavromatis & Karypis (2022)	76.40	70.90	52.90	47.80
Nutrea Choi et al. (2023)	77.40	72.70	53.60	49.50
LLM Reasoning				
GPT4o OpenAI (2024)	61.80	43.60	38.20	32.90
GPT4o+CoT Wei et al. (2022)	74.12	64.25	59.36	48.24
KG+LLM Reasoning				
SymAgent$_{qwen2\text{-}7b}$ Liu et al. (2025)	78.54	57.05	58.86	48.30
DoG$_{llama3.1\text{-}8b}$ Li et al. (2025a)	91.38	–	76.16	–
FRAG$_{llama3\text{-}70b}$ Gao et al. (2025b)	88.60	–	69.40	–
Hydrarag$_{gpt3.5\text{-}turbo}$ Tan et al. (2025)	**96.10**		81.20	
REKG-MCTS$_{gpt4}$ Song et al. (2025)	86.00	–	76.30	–
Debate on Graph$_{gpt4}$ Ma et al. (2025a)	91.00	–	56.00	–
FastToG$_{gpt4o\text{-}mini}$ Liang & Gu (2025)	65.80	–	45.00	–
AnchorRAG$_{gpt4o\text{-}mini}$ Xu et al. (2026a)	74.10	–	44.80	–
SKRAG$_{deepseek\text{-}v3}$ Xu et al. (2025)	90.10	75.30	67.20	54.60
EPERM$_{llama2\text{-}7b}$ Long et al. (2025)	88.80	72.40	66.20	58.90
OntGQA$_{llama2\text{-}7b}$ Shan et al. (2026)	91.50	77.10	74.60	56.80
GNN/Lightweight Retriever+LLM Reasoning				
G-Retriever$_{llama2\text{-}7b}$ He et al. (2024)	73.46	53.42	–	–
S-Path-RAG$_{llama2\text{-}7b}$ Fu et al. (2026)	88.90	78.20	77.90	75.20
RAPL$_{gpt4o}$ Yao et al. (2025)	93.30	80.70	69.00	58.80
SubgraphRAG$_{gpt4o}$ Li et al. (2025c)	90.91	78.24	67.49	59.42
STEM$_{gpt4o}$ Yu et al. (2026)	90.94	76.18	74.09	65.33
ParallaxRAG$_{gpt4o}$ Liu et al. (2026)	93.53	78.80	70.74	62.31
QSRAG$_{gpt4o\text{-}mini}$ Xu et al. (2026b)	84.77	55.50	54.04	49.77
DAMR$_{gpt4.1}$ Wang et al. (2025)	<u>94.00</u>	81.70	78.00	75.10
GNN-RAG$_{llama2\text{-}7b}$ Mavromatis & Karypis (2025)	85.70	71.30	66.80	59.40
RPO-RAG$_{llama2\text{-}7b}$ Um et al. (2026)	88.30	77.80	68.10	59.20
RwT$_{llama3\text{-}8b}$ Shen et al. (2025)	87.00	79.70	72.40	66.70
GRIL$_{llama3\text{-}8b}$ Chen et al. (2025)	86.80	73.00	68.30	60.50
LightPROF$_{llama3\text{-}8b}$ Ao et al. (2025)	83.80	–	59.30	–
D-RAG$_{llama3\text{-}8b}$ Gao et al. (2025a)	89.10	80.50	70.30	63.80
RPO-RAG$_{llama3.1\text{-}8b}$ Um et al. (2026)	89.90	81.30	72.30	64.50
RPO-RAG$_{llama3.2\text{-}1b}$ Um et al. (2026)	82.30	69.80	60.30	50.40
RPO-RAG$_{llama3.2\text{-}3b}$ Um et al. (2026)	87.30	76.40	66.00	57.30
SPARK$_{llama3.1\text{-}8b}$ (**Ours**)	91.99	**83.56**	<u>82.84</u>	**77.18**
SPARK$_{llama3.2\text{-}1b}$ (**Ours**)	89.35	76.90	75.18	69.66
SPARK$_{llama3.2\text{-}3b}$ (**Ours**)	91.22	<u>81.87</u>	76.50	71.01
SPARK$_{qwen3\text{-}1.7b}$ (**Ours**)	85.62	75.62	72.87	66.81
SPARK$_{qwen3\text{-}4b}$ (**Ours**)	88.02	77.13	75.05	68.53
SPARK$_{qwen3\text{-}8b}$ (**Ours**)	89.68	81.16	**83.43**	<u>76.53</u>

Table 1: Retrieval performance and average wall-clock time on WebQSP and CWQ.

Methods	WebQSP			CWQ		
	TR(↑)	AR(↑)	Avg. Time (s)(↓)	TR(↑)	AR(↑)	Avg. Time (s)(↓)
G-Retriever	0.294	0.545	672	0.183	0.375	1530
GNN-RAG	0.522	0.818	68	0.500	0.841	160
SubgraphRAG	0.883	0.944	6	0.811	0.914	12
QSRAG	0.905	0.942	–	0.903	0.955	–
SPTS(**Ours**)	**0.912**	**0.954**	**5**	**0.926**	**0.972**	**9**

Retrieval Results. Table 1 reports the retrieval performance and average wall-clock time on WebQSP and CWQ. SPTS consistently achieves the best results across both datasets and metrics. Notably, on the more challenging multi-hop dataset CWQ, SPTS attains a TR of 0.926 and an AR of 0.972, outperforming the strongest baseline QSRAG by 2.3 and 1.7 percentage points, respectively. On WebQSP, SPTS also achieves the highest TR (0.912) and AR (0.954). Among methods with reported runtimes, SPTS records the lowest average wall-clock time. Compared with SubgraphRAG, SPTS reduces inference time by 16.7% and 25.0% on the two datasets, while achieving higher TR and AR. These consistent gains demonstrate the benefit of conditioning each expansion on the complete path prefix, which promotes globally coherent search decisions and provides reliable evidence for downstream answer generation.

KGQA Results. Table 2 presents the KGQA results. SPARK achieves new SOTA F1 scores on both datasets. On WebQSP, SPARK$_{llama3.1\text{-}8b}$ attains an F1 of 83.56%, outperforming all baselines including those built on much larger LLMs (e.g., DAMR$_{gpt4.1}$ at 81.70%). On CWQ, SPARK$_{qwen3\text{-}8b}$ reaches the highest Hits@1 of 83.43% and SPARK$_{llama3.1\text{-}8b}$ delivers the best F1 of 77.18%, surpassing DAMR by 5.43 and 2.08 percentage points, respectively. These results suggest that the combination of coherent evidence retrieval and graph-aware KGL prompting enables the generators to produce more complete answer sets. Crucially, SPARK decouples efficient, prefix-consistent retrieval from LLM reasoning, avoiding repeated LLM-controlled retrieval calls required by iterative agent-based methods. SPARK also retains strong performance across generator scales: even SPARK$_{llama3.2\text{-}1b}$ achieves 89.35% Hits@1 and 76.90% F1 on WebQSP, outperforming the same-scale RPO-RAG variant and several systems based on larger generators. On CWQ, DAMR obtains

78.00% Hits@1 and 75.10% F1, whereas SPARK$_{llama3.1\text{-}8b}$ improves these scores to 82.84% and 77.18%, respectively.

4.3 Ablation Study

We carefully assess the efficacy and necessity of SPARK's core components and key hyperparameters on WebQSP and CWQ. Unless otherwise specified, all end-to-end studies, including *KGL prompting*, *collaborative training*, and *the effects of ρ and λ*, use SPARK$_{Llama3.2\text{-}3b}$. Retrieval-specific analyzes of *the core designs of SPTS* and *the ranking coefficient β* are conducted using SPTS.

Impact of Graph-Aware KGL Prompting. We evaluate graph-aware KGL prompting (gKGL) as a unified generator interface against Plain-text, which serializes the same retrieved paths using entity and relation surface forms. As shown in Table 3, gKGL improves Hits@1 and F1 from 86.37 and 76.13 to 91.22 and 81.87 on WebQSP, yielding gains of 4.85 and 5.74 percentage points, respectively. On CWQ, it increases Hits@1 by 4.09 points (72.41→76.50) and F1 by 2.69 points (68.32→71.01). gKGL also reduces the average input length by 23.5% (401.48→306.99 tokens), the P95 input length by 22.4% (812.51→630.46 tokens), and the average generation time by 38.5% (13→8 seconds). These results demonstrate that gKGL represents retrieved graph evidence more compactly while simultaneously improving answer accuracy and generation efficiency.

Table 3: Impact of graph-aware KGL prompting. P95 tokens denotes the 95th percentile (i.e., $\rho = 0.95$) of input-token counts.

	WebQSP		CWQ		Efficiency		
	Hits@1(↑)	F1(↑)	Hits@1(↑)	F1(↑)	Avg.tokens(↓)	P95 tokens(↓)	Avg.Times(s)(↓)
Plain-text	86.37	76.13	72.41	68.32	401.48	812.51	13
gKGL	**91.22**	**81.87**	**76.50**	**71.01**	**306.99**	**630.46**	**8**

Necessity of Collaborative Training and Impact of Evidence Augmentation. We consider four variants: No-RP removes Stage 1 retrieval-policy pretraining; Frozen-SE freezes the shared graph encoder during Stage 2; Unshared-GE assigns independent graph encoders for retrieval and generation; and No-GPA excludes gold-path augmentation from generator training. Table 4 shows that full SPARK achieves the best performance across all metrics. Both No-RP and Frozen-SE cause substantial degradation: on CWQ, Hits@1 falls from 76.50 to 67.28 and 69.31, respectively, while WebQSP F1 decreases from 81.87 to 77.62 and 75.24. These losses establish the importance of retrieval-policy pretraining and continued adaptation of the shared encoder. Unshared-GE incurs smaller but consistent declines, reducing F1 to 78.98 on WebQSP and 68.23 on CWQ, thereby supporting representation sharing between retrieval and generation. The most pronounced deterioration occurs with No-GPA: F1 drops to 62.37 and 45.85, while AR declines to 0.765 and 0.614 on WebQSP and CWQ, respectively. Gold-path augmentation therefore provides crucial supervision for both evidence learning and answer generation.

Table 4: Effects of the collaborative training strategy and gold-path augmentation.

	WebQSP				CWQ			
	TR(↑)	AR(↑)	Hits@1(↑)	F1(↑)	TR(↑)	AR(↑)	Hits@1(↑)	F1(↑)
Full SPARK	**0.912**	**0.954**	**91.22**	**81.87**	**0.926**	**0.972**	**76.50**	**71.01**
w/o No-RP	0.859	0.876	87.54	77.62	0.829	0.861	67.28	65.83
w/o Frozen-SE	0.842	0.879	86.33	75.24	0.854	0.886	69.31	66.14
w/o Unshared-GE	0.891	0.932	89.41	78.98	0.882	0.931	71.21	68.23
w/o No-GPA	0.681	0.765	73.06	62.37	0.509	0.614	57.28	45.85

Effect of different ρ. Fig. 2 examines the cumulative probability threshold ρ, which controls search width. Performance is maximized at $\rho = 0.95$, where the triple ratios decrease to 0.620 on WebQSP and 0.697 on CWQ, while both Hits@1 and F1 exceed those obtained under full search ($\rho = 1.0$). Here, the triple ratio is the number of triples retrieved at a given ρ divided by the number retrieved when $\rho = 1.0$. Retaining every prefix-consistent path therefore introduces redundant or distracting evidence that impedes LLM reasoning. Conversely, thresholds below 0.95 prune too aggressively and begin to discard answer-relevant triples, leading to performance degradation. These findings show that prefix-conditioned scoring concentrates high-quality evidence within a compact set of top-ranked branches, yielding an effective precision-completeness trade-off.

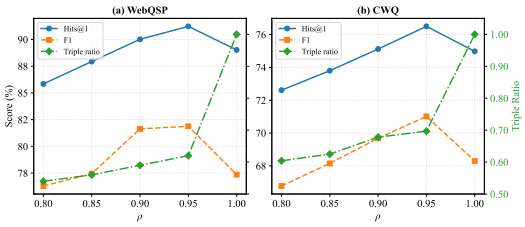

Figure 2: KGQA performance (%) with varying ρ.

Impact of varying λ. Fig. 3 illustrates how λ mediates the trade-off between retrieval supervision and generation flexibility. Setting $\lambda = 0$, which removes the retrieval loss, degrades both metrics, indicating that generation supervision alone cannot induce compact evidence selection. Performance peaks at $\lambda = 0.1$, where the retrieval and generation objectives are most effectively balanced. Larger values place excessive emphasis on retrieval fidelity and consequently weaken answer generation. The resulting non-monotonic trend underscores the need to coordinate the two objectives rather than optimize either one in isolation.

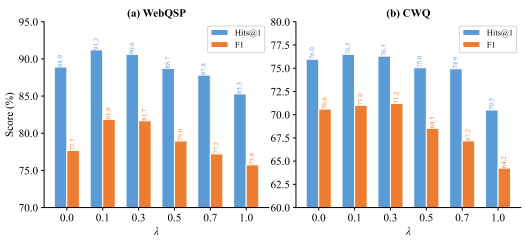

Figure 3: KGQA performance (%) with varying λ .

Effects of Core SPTS Designs. We assess four core designs of SPTS: *full-prefix conditioning*, *hierarchical relation–entity selection*, *positional embeddings*, and *type embeddings*. As reported in Table 5, full SPTS attains the highest TR and AR on both datasets. Removing full-prefix conditioning causes the largest overall deterioration, reducing TR/AR from 0.912/0.954 to 0.816/0.847 on WebQSP and from 0.926/0.972 to 0.835/0.863 on CWQ. They highlight the central role of complete trajectory context in coherent evidence expansion. Replacing hierarchical relation–entity selection with direct triple-level scoring likewise impairs retrieval. Hierarchical action decomposition therefore enables more effective discrimination among candidate expansions. The embedding ablations further reveal the value of structural cues. Removing positional embeddings produces moderate TR/AR reductions of 0.011/0.017 on WebQSP and 0.013/0.011 on CWQ. In contrast, removing type embeddings yields larger declines, particularly on CWQ. Eliminating both embeddings produces the weakest results among the embedding variants. Together, these results show that positional and type embeddings provide complementary structural information, with entity–relation type distinctions contributing more strongly than positional cues, especially on CWQ.

Table 5: Effects of core SPTS designs on retrieval performance.

	WebQSP		CWQ	
	TR(↑)	AR(↑)	TR(↑)	AR(↑)
Full SPTS	**0.912**	**0.954**	**0.926**	**0.972**
w/o full-prefix conditioning	0.816	0.847	0.835	0.863
w/o hierarchical relation–entity selection	0.884	0.912	0.897	0.936
w/o positional embedding	0.901	0.937	0.913	0.961
w/o type embedding	0.894	0.924	0.885	0.932
w/o both embeddings	0.882	0.913	0.862	0.920

Effect of varying β. Fig. 4 shows that SPTS underperforms at $\beta = 0$ (KL-only), demonstrating that the pairwise ranking term is essential for separating valid from spurious expansions. WebQSP performs best at $\beta = 0.3$, whereas the deeper and more ambiguous search space of CWQ favors stronger regularization at $\beta = 0.5$. Excessively large values, such as $\beta = 1.0$, degrade performance on both datasets, suggesting that overemphasizing pairwise ranking distorts the target distribution. The hybrid KL+ranking objective must therefore be balanced according to dataset characteristics.

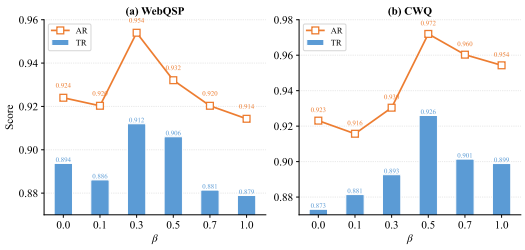

Figure 4: Retrieval performance with varying β.

5 Conclusion

We presented SPARK, a KGQA framework that integrates structure-aware evidence retrieval with graph-enhanced LLM generation. Its hierarchical relation–entity decisions enable coherent path retrieval without modeling a large flat action space, while graph-aware KGL prompting compactly combines symbolic paths with fine-grained graph representations. A two-stage strategy further coordinates retrieval and generation through a shared graph encoder. Extensive experiments demonstrate the effectiveness of SPARK in unifying structured graph exploration with LLM reasoning. This work provides a promising avenue for integrating structured graph exploration with large language models in knowledge-intensive reasoning tasks.

AI USE STATEMENT

We used GPT and DeepSeek to assist with translation and English-language polishing of the manuscript. The authors manually reviewed all AI-assisted text for accuracy and consistency with the intended meaning.

The training and evaluation data used in this work are the existing WebQSP and CWQ benchmarks. We did not use generative AI to generate synthetic datasets, create or annotate training examples, or produce reference answers. The open-source LLMs Llama3.1-8B-Instruct, Llama3.2-1B/3B-Instruct, and Qwen3-1.7B/4B/8B are used as answer generators within the proposed SPARK method; this use is part of the experimental setup and is described in the main paper.

Generative AI was not used for method design, analysis or interpretation of experimental results, or figure creation. The authors take full responsibility for the final manuscript and the research.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Tu Ao, Yanhua Yu, Yuling Wang, Yang Deng, Zirui Guo, Liang Pang, Pinghui Wang, Tat-Seng Chua, Xiao Zhang, and Zhen Cai. Lightprof: A lightweight reasoning framework for large language model on knowledge graph. In *AAAI*, volume 39, pp. 23424–23432, 2025.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, 2008.

Jialin Chen, Houyu Zhang, Seongjun Yun, Alejandro Mottini, Rex Ying, Xiang Song, Vassilis_N Ioannidis, Zheng Li, and Qingjun Cui. Gril: Knowledge graph retrieval-integrated learning with large language models. Association for Computational Linguistics, 2025.

Hyeong Kyu Choi, Seunghun Lee, Jaewon Chu, and Hyunwoo J. Kim. Nutrea: neural tree search for context-guided multi-hop kgqa. *NeurIPS*, 36:12, 2023.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 6491–6501, 2024.

Rong Fu, Yemin Wang, Tianxiang Xu, Yongtai Liu, Weizhi Tang, Wangyu Wu, Xiaowen Ma, and Simon Fong. S-path-rag: Semantic-aware shortest-path retrieval augmented generation for multi-hop knowledge graph question answering. In *Proceedings of the ACM Web Conference 2026*, pp. 4057–4068, 2026.

Guangze Gao, Zixuan Li, Chunfeng Yuan, Jiawei Li, Wu Jianzhuo, Yuehao Zhang, Xiaolong Jin, Bing Li, and Weiming Hu. D-rag: Differentiable retrieval-augmented generation for knowledge graph question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35386–35405, 2025a.

Zengyi Gao, Yukun Cao, Hairu Wang, Ao Ke, Yuan Feng, S Kevin Zhou, and Xike Xie. Frag: A flexible modular framework for retrieval-augmented generation based on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6178–6192, 2025b.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Lingbing Guo, Zhongpu Bo, Zhuo Chen, Yichi Zhang, Jiaoyan Chen, Yarong Lan, Mengshu Sun, Zhiqiang Zhang, Yangyifei Luo, Qian Li, et al. Mkgl: mastery of a three-word language. *NeurIPS*, 37:140509–140534, 2024.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *NeurIPS*, 37:132876–132907, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Xinke Jiang, Ruizhe Zhang, Yongxin Xu, Rihong Qiu, Yue Fang, Zhiyuan Wang, Jinyi Tang, Hongxin Ding, Xu Chu, Junfeng Zhao, et al. Hykge: A hypothesis knowledge graph enhanced rag framework for accurate and reliable medical llms responses. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 11836–11856, 2025.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kun Li, Tianhua Zhang, Xixin Wu, Hongyin Luo, James Glass, and Helen Meng. Decoding on graphs: Faithful and sound reasoning on knowledge graphs through generation of well-formed chains. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 24349–24364, 2025a.

Moxin Li, Yong Zhao, Wenxuan Zhang, Shuaiyi Li, Wenya Xie, See-Kiong Ng, Tat-Seng Chua, and Yang Deng. Knowledge boundary of large language models: A survey. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 5131–5157, 2025b.

Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2025, pp. 6061–6089, 2025c.

Xujian Liang and Zhaoquan Gu. Fast think-on-graph: Wider, deeper and faster reasoning of large language model on knowledge graph. In *AAAI*, volume 39, pp. 24558–24566, 2025.

Ben Liu, Jihai Zhang, Fangquan Lin, Cheng Yang, Min Peng, and Wotao Yin. Symagent: A neural-symbolic self-learning agent framework for complex reasoning over knowledge graphs. In *Proceedings of the ACM on Web Conference 2025*, pp. 98–108, 2025.

Jinliang Liu, Jiale Bai, and Shaoning Zeng. Think parallax: Solving multi-hop problems via multi-view knowledge-graph-based retrieval-augmented generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 26637–26655, 2026.

Xiao Long, Liansheng Zhuang, Aodi Li, Minghong Yao, and Shafei Wang. Eperm: An evidence path enhanced reasoning model for knowledge graph question and answering. In *AAAI*, volume 39, pp. 12282–12290, 2025.

Kangyang Luo, Yuzhuo Bai, Cheng Gao, Shuzheng Si, Zhu Liu, Yingli Shen, Zhitong Wang, Cunliang Kong, Wenhao Li, Yufei Huang, Ye Tian, Xuantang Xiong, Lei Han, and Maosong Sun. GLTW: Joint improved graph transformer and LLM via three-word language for knowledge graph completion. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 11328–11344, 2025a.

Shijie Luo, Xinyuan Lu, Qinpei Zhao, and Weixiong Rao. Bridging the gap between knowledge graphs and llms for multi-hop question answering. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 5006–5010, 2025b.

Jie Ma, Zhitao Gao, Qi Chai, Wangchun Sun, Pinghui Wang, Hongbin Pei, Jing Tao, Lingyun Song, Jun Liu, Chen Zhang, et al. Debate on graph: a flexible and reliable reasoning framework for large language models. In *AAAI*, volume 39, pp. 24768–24776, 2025a.

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 *ICLR*, volume 2025, pp. 52782–52806, 2025b.

Costas Mavromatis and George Karypis. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2447–2458, December 2022. URL https://aclanthology.org/2022.findings-emnlp.181.

Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16682–16699, 2025.

Yunfeng Ning, Mayi Xu, Jintao Wen, Qiankun Pi, Yuanyuan Zhu, Ming Zhong, Jiawei Jiang, and Tieyun Qian. Privacy-protected retrieval-augmented generation for knowledge graph question answering. In *AAAI*, volume 40, pp. 32573–32581, 2026.

OpenAI. Hello gpt-4o, 2024.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, 2024.

Minbae Park, Hyemin Yang, Jeonghyun Kim, Kunsoo Park, and Hyunjoon Kim. Prograg: Hallucination-resistant progressive retrieval and reasoning over knowledge graphs. In *AAAI*, volume 40, pp. 32674–32682, 2026.

Joaquín Polonuer, Lucas Vittor, Iñaki Arango, Ayush Noori, David A Clifton, Luciano Del Corro, and Marinka Žitnik. Autonomous knowledge graph exploration with adaptive breadth-depth retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 15694–15709, 2026.

Yongxue Shan, Jie Peng, Zixuan Dong, Fei Hu, and Xiaodong Wang. Reasoning with ontology graph: Toward type-constrained knowledge graph question answering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 10684–10697, 2026.

Tiesunlong Shen, Jin Wang, Xuejie Zhang, and Erik Cambria. Reasoning with trees: faithful question answering over knowledge graph. In *ICLR*, pp. 3138–3157, 2025.

Xiaozhuang Song, Shufei Zhang, and Tianshu Yu. Rekg-mcts: Reinforcing llm reasoning on knowledge graphs via training-free monte carlo tree search. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9288–9306, 2025.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *EMNLP*, pp. 4231–4242, 2018.

Alon Talmor and Jonathan Berant. 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*, pp. 641–651, 2018.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. Hydrarag: Structured cross-source enhanced large language model reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14442–14470, 2025.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Kaehyun Um, KyuHwan Yeom, Haerim Yang, Minyoung Choi, Hyeongjun Yang, and Kyong-Ho Lee. Rpo-rag: Aligning small llms with relation-aware preference optimization for knowledge graph question answering. In *Proceedings of the ACM Web Conference 2026*, pp. 2513–2523, 2026.

Yingxu Wang, Shiqi Fan, Mengzhu Wang, Siyang Gao, Chao Wang, and Nan Yin. Damr: Efficient and adaptive context-aware knowledge graph question answering with llm-guided mcts. *arXiv preprint arXiv:2508.00719*, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 35: 24824–24837, 2022.

Jiasheng Xu, Mingda Li, Yongqiang Tang, Peijie Wang, and Wensheng Zhang. Towards open-world retrieval-augmented generation on knowledge graph: A multi-agent collaboration framework. In *Proceedings of the ACM Web Conference 2026*, pp. 2171–2182, 2026a.

Xiaotong Xu, Yizhao Wang, Yunfei Liu, and Shengyang Li. SKRAG: A retrieval-augmented generation framework guided by reasoning skeletons over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 13983–13994, 2025.

Yuanye Xu, Linyi Guo, Yue Zhang, and Fu Ning. Query-aware graph attention for precise subgraph retrieval in knowledge-augmented reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8127–8148, 2026b.

Tianjun Yao, Haoxuan Li, Zhiqiang Shen, Pan Li, Tongliang Liu, and Kun Zhang. Learning efficient and generalizable graph retriever for knowledge-graph question answering. *arXiv preprint arXiv:2506.09645*, 2025.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 1321–1331, 2015.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 201–206, 2016.

Peng Yu, En Xu, Bin Chen, Haibiao Chen, and Yinfei Xu. Stem: Structure-tracing evidence mining for knowledge graphs-driven retrieval-augmented generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, pp. 7252–7285, 2026.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):58, 2024.

APPENDIX

A RELATED WORK

KG Retrieval for LLMs. Existing LLM-enhanced KGQA methods generally follow a "*retrieve-then-reason*" paradigm. Some efforts employ an LLM as a retrieval agent that interacts with the KG via hop-by-hop exploration, evidence-sufficiency assessment, or reasoning path planning Liang & Gu (2025); Ma et al. (2025b); Liu et al. (2025); Tan et al. (2025); Li et al. (2025a); Ma et al. (2025a); Polonuer et al. (2026); Xu et al. (2026a); Shan et al. (2026). Although these methods offer strong semantic planning capabilities, they often require multiple rounds of LLM calls, incurring substantial inference costs. Another line of work utilizes lightweight models Yao et al. (2025); Li et al. (2025c); Jiang et al. (2025); Mavromatis & Karypis (2025); Yu et al. (2026); Park et al. (2026); Xu et al. (2026b); Liu et al. (2026) or specialized tree search algorithms to retrieve graph evidence Shen et al. (2025); Wang et al. (2025); Song et al. (2025); Gao et al. (2025b). Despite improved retrieval efficiency and accuracy, many of them mainly score nodes, triples, or complete paths, with limited modeling of the dependency between the complete path prefix and the current expansion action. More recently, some studies leverage continuous relaxation or stochastic reparameterization to establish gradient pathways from generation signals to retrieval modules Chen et al. (2025); Gao et al. (2025a). However, continuous approximations of discrete graph structures can complicate optimization, while selecting individual triples does not inherently preserve evidence connectivity or ordering.

Aligning KG Structures with LLMs. Most existing methods serialize the retrieved triples or paths as text and append the resulting sequences to the LLM input context Long et al. (2025); Xu et al. (2025); Shan et al. (2026); Um et al. (2026). Although broadly compatible with different LLMs, this design can weaken explicit graph connectivity and leads to increasing context overhead as more evidence is retrieved. To exploit graph structure more directly, some studies inject subgraph-, path-, or triple-level representations into LLMs as soft prompts or latent features He et al. (2024); Luo et al. (2025b); Chen et al. (2025); Ao et al. (2025); Gao et al. (2025a); Fu et al. (2026). Unlike these coarse-grained representations, SPARK injects entity- and relation-level graph embeddings at KGL symbol positions, preserving symbolic semantics, path ordering, and fine-grained graph structure within a compact input.

B DETAILS OF QUESTION-CONDITIONED GRAPH REPRESENTATION LEARNING

This section details how the question representation $\mathbf{q}$, entity representation matrix $\mathbf{E}_{\mathcal{E}}$, and relation representation matrix $\mathbf{E}_{\mathcal{R}}$ in Eq. (1) are obtained. Overall, we follow the encoding pipeline of NuTrea Choi et al. (2023), which takes the natural language question q, the knowledge graph $\mathcal{G}$, and the seed entity set $\mathcal{S}_q \subseteq \mathcal{E}$ obtained via entity linking as input, and produces question-aware graph representations. Because SPARK additionally requires a single question vector for path-conditioned search, we aggregate the expansion and backup instructions produced by NuTrea into $\mathbf{q}$.

First, NuTrea generates two sets of question-conditioned instruction vectors from q via two language model-based instruction generators, denoted as IG_{exp} and IG_{bak}:

$$\{\mathbf{q}_{exp}^{(i)}\}_{i=1}^{N} = \text{IG}_{exp}(q), \quad \{\mathbf{q}_{bak}^{(j)}\}_{j=1}^{M} = \text{IG}_{bak}(q), \tag{11}$$

where $\mathbf{q}_{exp}^{(i)}, \mathbf{q}_{bak}^{(j)} \in \mathbb{R}^{d_n}$ denote the expansion and backtracking instructions, respectively. The former guides the model to propagate messages along the graph structure, while the latter introduces question-relevant context from local subtrees. NuTrea retains the two instruction sets separately for expansion and backup. To obtain the single question vector required by the SPARK path encoder, we additionally aggregate them as

$$\mathbf{q} = \frac{1}{2}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{q}_{\text{exp}}^{(i)} + \frac{1}{M}\sum_{j=1}^{M}\mathbf{q}_{\text{bak}}^{(j)}\right). \tag{12}$$

For implementation details of the instruction generators, we refer readers to Mavromatis & Karypis (2022); Choi et al. (2023).

Second, for the set of relation types $\mathcal{R}$, we encode the natural language description of each relation type using BERT[2] to obtain the initial relation embeddings:

$$\mathbf{r}_j^{(0)} = \mathbf{W}_r \, \mathrm{BERT}(\mathrm{desc}(r_j)), \qquad \mathbf{E}_{\mathcal{R}}^{(0)} = [\mathbf{r}_j^{(0)}]_{j=1}^{|\mathcal{R}|}, \tag{13}$$

where $\mathbf{W}_r \in \mathbb{R}^{d_n \times d_{\mathrm{BERT}}}$ is a learnable parameter matrix. During training, the question and relation embeddings are optimized jointly with the graph encoder parameters. For notational simplicity, we denote the final question embedding and relation embedding matrix as $\mathbf{q}$ and $\mathbf{E}_{\mathcal{R}}$, respectively.

Third, since entity surface forms often provide limited semantic cues, NuTrea initializes entity representations based on neighboring relations via an RF-IEF mechanism. Specifically, RF-IEF computes a relation weight matrix $\mathbf{F} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{R}|}$ according to the local frequency and global distribution of relation types in the entity's neighborhood. The initial entity representations are obtained by weighted aggregation of relation embeddings:

$$\mathbf{E}_{\mathcal{E}}^{(0)} = \mathbf{F}\mathbf{E}_{\mathcal{R}}^{(0)}\mathbf{W}_h, \tag{14}$$

where $\mathbf{W}_h$ is a learnable parameter matrix. This scheme down-weights frequent yet less discriminative relation types and highlights more distinctive relational semantics.

Finally, NuTrea takes $\mathbf{E}_{\mathcal{E}}^{(0)}$, $\mathbf{E}_{\mathcal{R}}^{(0)}$, the seed entity set $\mathcal{S}_q$, and the question instructions as input, and performs L layers of neural tree search encoding. Each layer propagates question-related information via the expansion mechanism and incorporates local subtree context via the backtracking mechanism, thereby updating entity node representations. This process can be abstractly expressed as

$$\begin{aligned} \mathbf{E}_{\mathcal{E}}^{(\ell)} =& \text{NuTreaLayer}^{(\ell)}(\mathbf{E}_{\mathcal{E}}^{(\ell-1)}, \mathbf{E}_{\mathcal{R}}^{(\ell-1)}, \\ & \{\mathbf{q}_{exp}^{(i,\ell-1)}\}_{i=1}^{N}, \{\mathbf{q}_{bak}^{(j,\ell-1)}\}_{j=1}^{M}), \quad \ell = 1, \ldots, L. \end{aligned} \tag{15}$$

After L layers of encoding, the entity representations from the final layer serve as the question-aware entity embedding matrix:

$$\mathbf{E}_{\mathcal{E}} = \mathbf{E}_{\mathcal{E}}^{(L)}. \tag{16}$$

Thus, $\mathbf{q}$, $\mathbf{E}_{\mathcal{E}}$, and $\mathbf{E}_{\mathcal{R}}$ are the resulting representations of the question, entities, and relations, respectively. For a candidate triple $t = (e_v, r, e) \in \mathcal{T}(e_v, \chi_v)$, the corresponding graph representations are obtained as $\mathbf{E}_{\mathcal{E}}[e_v]$, $\mathbf{E}_{\mathcal{R}}[r]$, and $\mathbf{E}_{\mathcal{E}}[e]$. These representations are subsequently used for path encoding and graph-aware answer generation.

C Knowledge Graph Language-Based Path Reasoning Prompt Template

D Complete Experimental Settings

Datasets and Baselines. We conduct experiments on the widely used KGQA datasets, WebQSP Yih et al. (2016) and CWQ Talmor & Berant (2018), both built upon the Freebase KG Bollacker et al. (2008), which contains roughly 88M entities, 20K relations, and 126M factual triples. WebQSP consists of 4,737 questions requiring up to 2-hop reasoning, whereas CWQ contains 34,689 complex questions that demand up to 4-hop reasoning with compositional or constraint-based patterns. We adopt the standard training, valid, and test splits from prior work Shan et al. (2026) and summarize the dataset statistics in Table 6. For evaluation, we compare our retrieval method SPTS against existing baselines including G-Retriever He et al. (2024), GNN-RAG Mavromatis & Karypis (2025), SubgraphRAG Li et al. (2025c), and QSRAG Xu et al. (2026b), and compare our KGQA method SPARK against strong baselines, which we group into four categories (*GNN Reasoning*, *LLM Reasoning*, *KG+LLM Reasoning*, and *GNN/Lightweight Retriever+LLM Reasoning*) detailed in Table 2. For SPARK, we employ multiple open-source LLMs of varying scales as generators, specifically Llama3.1-8B-Instruct Grattafiori et al. (2024), Llama3.2-1B/3B-Instruct [3], and Qwen3-1.7B/4B/8B Team (2025).

[2]https://huggingface.co/google-bert/bert-base-uncased

[3]https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct, https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Task Instruction:
You are an expert assistant for Knowledge Graph Question Answering (KGQA). Your core capability lies in deeply understanding natural language questions, the semantics of the specific entities and relations from the knowledge graph as listed in the Dictionary below, and the given reasoning paths, in order to infer the correct answer.
Dictionary (Entities and Relations):

Input	Type	Description
<kgl:Pakistan>	entity	Pakistan
<kgl:English Language>	entity	English Language
<kgl:Urdu Language>	entity	Urdu Language
<kgl:location.country.languages_spoken>	relation	location.country.languages_spoken
<kgl:location.country.official_language>	relation	location.country.official_language

Reasoning Paths:
<kgl:Pakistan> → <kgl:location.country.languages_spoken> → <kgl:English Language>
<kgl:Pakistan> → <kgl:location.country.languages_spoken> → <kgl:Urdu Language>
<kgl:Pakistan> → <kgl:location.country.official_language> → <kgl:English Language>
<kgl:Pakistan> → <kgl:location.country.official_language> → <kgl:Urdu Language>
Question:
what language do the speak in pakistan

Answer:
English Language|Urdu Language

Figure 5: Knowledge Graph Language-Based Path Reasoning Prompt Template

Table 6: Dataset statistics. We set the maximum search depth D to 2 for WebQSP and 4 for CWQ.

Datasets	#Train	#Valid	#Test	Max #hop (i.e., D)
WebQSP	2826	283	1628	2
CWQ	27639	3519	3531	4

Metrics and Configurations. Following prior work Li et al. (2025c); Xu et al. (2026b), we evaluate performance in terms of retrieval and reasoning. For retrieval, we adopt Triple Recall (TR) and Answer Recall (AR) as evaluation metrics. The former measures the proportion of triples in the retrieval results that lie on the shortest paths from seed entities to answer entities; the latter measures the proportion of answer entities covered by the subgraph induced by the retrieved triples. For reasoning (KGQA), we use Hits@1 and F1 as evaluation metrics. Hits@1 denotes the proportion of questions for which the correct answer is ranked first in the output; F1 combines precision and recall, measuring both accuracy and coverage. Throughout all experiments, unless otherwise specified, the hyperparameters are set as follows: for WebQSP and CWQ, we set $\beta = \beta_r = \beta_e = 0.3$ and 0.5, respectively, with $\rho = \rho_r = \rho_e = 0.95$ and $\lambda = 0.1$ for both datasets. The output dimension of f_{enc} is $d_n = 256$. We adopt the Adam optimizer Kingma & Ba (2014) with a two-stage learning rate schedule: in the first stage, both NuTrea_ω and f_θ use a learning rate of $5e{-}4$ on WebQSP and $2.5e{-}4$ on CWQ; in the second stage, the learning rates for NuTrea_ω, f_θ, and the generator f_ϕ are all set to $1e{-}5$, while the projector Proj_ψ uses $5e{-}4$ and $2.5e{-}4$ for the two datasets, respectively. During the second stage, we apply gradient accumulation every 4 steps with a gradient clipping threshold of 1.0, and employ the ExponentialLR scheduler with a decay rate of 0.97. We use LoRA Hu et al. (2022) (rank $r = 8$, scaling factor $\alpha = 16$) to train f_ϕ. We perform validation every 2 epochs, and adopt early stopping with a patience of 20 epochs, selecting the final model based on the F1 score on the validation set. All experiments are implemented using PyTorch. The models are trained for 100 epochs with early stopping and run on 8 A100 GPUs (40GB each).