D-RAG: Differentiable Retrieval-Augmented Generation for Knowledge Graph Question Answering

Anonymous ACL submission

Abstract

Knowledge Graph Question Answering (KGQA) aims to answer natural language questions based on knowledge graphs. Recent approaches apply the Retrieval-Augmented Generation (RAG) paradigm to incorporate Large Language Models (LLMs) to this task, where a retriever selects a question-related subgraph and an LLM-based generator is then adopted to predict answers based on the retrieved subgraph. However, the subgraph selection process is non-differentiable, preventing end-to-end training of the retriever and the generator in these approaches, which leads to sub-optimal performance. To overcome this limitation, this paper proposes a Differentiable RAG (D-RAG) approach that jointly optimizes the retriever and the generator for KGQA. Firstly, D-RAG reformulates the optimization objective as an expectation over a subgraph distribution with respect to answer generation likelihood, making the joint optimization feasible. Secondly, it designs a differentiable subgraph sampling and prompting module based on Gumbel-Softmax reparameterization, which achieves end-to-end optimization and allows the retriever to discover latent graph patterns that actively facilitate the generator's reasoning process. Experimental results on WebQSP and CWQ show that D-RAG outperforms the state-of-the-art approaches by 2.3% and 3.4% on the F1 scores, respectively, demonstrating its effectiveness.

1 Introduction

Knowledge Graph Question Answering (KGQA) aims to automatically answer natural language questions via well-structured fact information stored in Knowledge Graphs (KGs). It is an essential task in Natural Language Processing (NLP) and is vital in various applications such as information retrieval and intelligent assistance (Potdar et al., 2025; Liang et al., 2024). However, KGQA poses challenges to existing approaches, as it requires a deep understanding of natural language questions and the ability to perform complex reasoning over KGs. Considering that Large Language Models (LLMs) (DeepSeek, 2025; OpenAI, 2024; Meta, 2024) have shown strong capabilities in natural language understanding and reasoning, some recent approaches (Peng et al., 2024; Luo et al., 2024; He et al., 2024) incorporate LLMs into KGQA via the Retrieval-Augmented Generation (RAG) paradigm (Lewis et al., 2020). Specifically, they adopt a retriever to select a question-relevant subgraph from the KG. Then, they serialize the subgraph into the prompt and adopt LLMs as the generator to reason for answers.

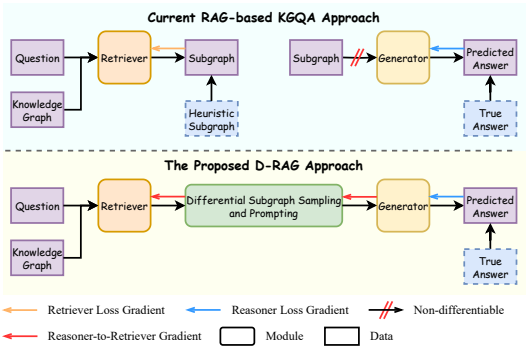

Figure 1: Comparison between the current RAG-based KGQA approaches and the proposed D-RAG approach. The red arrows highlight the end-to-end gradient flow.

Despite the promising performance of these RAG-based KGQA approaches, significant challenges remain in optimizing both the retriever and the generator. As illustrated in Figure 1, the core challenge stems from the non-differentiable nature of discrete subgraph selection, which prevents direct gradient flow from the generator to the retriever. While current approaches (Luo et al., 2024; Mavromatis and Karypis, 2024) typically adopt a sequential optimization paradigm, where the retriever is

trained using heuristic supervision signals, and the generator is subsequently optimized with the retriever frozen, this optimization paradigm has the following limitations: 1) The isolated optimization of individual modules leads to sub-optimal performance of the complete system rather than joint optimization; 2) The heuristic supervision signals used for training the retriever may not align well with the actual requirements of the generation task; 3) The system fails to leverage the generator's semantic understanding capabilities to enhance retriever performance.

To address these limitations in the existing RAG-based KGQA approaches, we propose the **D**ifferentiable **R**etrieval-**A**ugmented **G**eneration (D-RAG) for KGQA. Our approach introduces several innovations. First, we reformulate the optimization objective as a tractable expectation over a subgraph distribution with respect to answer generation likelihood, making the joint optimization mathematically feasible. Second, we design a differentiable subgraph sampling and prompting module that operates in two steps. The first step transforms discrete subgraph selection into differentiable fact-level sampling using the Gumbel-Softmax reparameterization trick (Jang et al., 2017; Maddison et al., 2017). The second step converts the sampled facts into LLM-compatible prompts while maintaining proper gradient flow throughout the entire pipeline. The combination of these innovations enables end-to-end optimization between the retriever and the generator, allowing the whole system to leverage the generator's semantic understanding to guide retrieval. Experimental results on WebQSP and CWQ show that D-RAG outperforms the state-of-the-art approaches by 2.4% and 1.0% on Hits@1, and by 2.3% and 3.4% on the F1 scores, respectively.

The main contributions of this work are as follows:

- We propose D-RAG, the first differentiable RAG-based KGQA approach, to the best of our knowledge, that enables end-to-end optimization with gradient flow from the generator to the retriever.
- We reformulate the objective as a tractable expectation over a subgraph distribution and design a differentiable subgraph sampling and prompting module based on Gumbel-Softmax reparameterization, achieving end-to-end joint optimization of the KGQA system.
- Comprehensive experiments on two widely used benchmark datasets, i.e., WebQSP and CWQ, demonstrate that D-RAG outperforms state-of-the-art performance, validating the effectiveness of our differentiable approach.

2 Related Works

2.1 Knowledge Graph Question Answering

KGQA approaches can be broadly categorized into Semantic Parsing-based (SP-based) and Information Retrieval-based (IR-based) ones (Lan et al., 2023). Since this work belongs to the IR-based category, we focus on IR-based approaches that retrieve question-specific subgraphs and either rank candidate answers or directly generate answers (Sun et al., 2018; He et al., 2021; Zhang et al., 2022).

With the powerful reasoning capabilities of LLMs, directly generating answers with text decoder in IR-based approaches has become increasingly promising, leading to RAG-based approaches. These approaches can be further divided into two groups based on how they retrieve question-specific subgraphs: graph-LLM approaches that leverage specialized graph-based techniques (e.g. GNNs) during subgraph retrieval (He et al., 2024; Li et al., 2025; Mavromatis and Karypis, 2024; Liu et al., 2024a), and LLM reasoning methods that primarily rely on LLMs to understand and reason over graph structure (Luo et al., 2024; Jiang et al., 2023a; Sun et al., 2024; Ma et al., 2024).

Current RAG-based KGQA approaches lack end-to-end training capabilities. Although SR (Zhang et al., 2022) achieves end-to-end KGQA by constructing tree-structured subgraphs from multi-hop paths, their posterior approximation requires computing answer generation probability for each top-k path independently, which would incur prohibitive computational costs when LLMs serve as the generator.

2.2 End-to-End Training in RAG

Most RAG systems follow a pipeline paradigm (Gao et al., 2023), where separate modules for retrieval, prompting, and generation are optimized separately. Several works have explored end-to-end trainable approaches for text retrieval, including REALM (Guu et al., 2020), EMDR^2 (Sachan et al., 2021), VOD (Liévin et al., 2023), and StochasticRAG (Zamani and Bendersky, 2024). However, these text-centric

methods cannot be directly applied to KGQA due to the structured nature of graph data and the need for specialized graph retrieval mechanisms.

StochasticRAG (Zamani and Bendersky, 2024) is the most similar one to D-RAG, as both methods leverage Gumbel tricks for discrete sampling, whether for documents or subgraphs. However, D-RAG differs in two key aspects: (1) StochasticRAG retrieves a fixed number of documents, which is not suitable for KGQA. In contrast, our approach transforms subgraph sampling into independent sampling of facts, allowing for flexible subgraph sizes; (2) Unlike documents that can be directly fed to LLMs, we employ a differentiable prompting step to bridge the gap between graph structures and LLM reasoning.

3 Preliminary

Knowledge Graph Question Answering. In this paper, the knowledge graph is composed of multiple facts, where each fact $\tau = (h, r, t)$ represents a triple consisting of a head entity h, a relation r, and a tail entity t. Formally, the KG can be represented as $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ denotes the set of all entities and $\mathcal{R}$ represents the set of all relation types, with each entity and relation type typically corresponding to a natural language form. Given a knowledge graph $\mathcal{G}$, the KGQA task takes a natural language question q as input and outputs an answer a corresponding to one or more entities in $\mathcal{G}$. The ultimate goal is to maximize the likelihood of the correct answer, which can be formulated as $\mathbb{E}_{(q,\, a)} \left[\log p(a|q, \mathcal{G})\right]$.

RAG-based KGQA. The RAG paradigm in KGQA involves two independent modules: a retriever R_β that identifies the question-relevant subgraph g_{sub} with probability $p_\beta(g_{sub}|\mathcal{G}, q)$, and a generator G_γ that generates the answer a with probability $p_\gamma(a|g_{sub}, q)$.

The overall answer generation probability can be formulated as:

$$p_\theta(a|q, \mathcal{G}) = \sum_{g_{sub} \subseteq \mathcal{G}} p_\gamma(a|q, g_{sub}) p_\beta(g_{sub}|q, \mathcal{G}), \tag{1}$$

where β and γ denote the parameters of the retriever and the generator, respectively. θ denotes all parameters in the above two modules.

Current RAG-based KGQA approaches face a fundamental challenge in joint optimization - the discrete nature of subgraph retrieval creates a non-differentiable barrier between the retriever and the generator. Existing approaches circumvent this by separately training the retriever (using heuristic subgraph labels) and freezing it during generator training. The decoupled paradigm inevitably causes error propagation and suboptimal performance.

4 The Proposed D-RAG Approach

This section presents **D**ifferentiable **R**etrieval-**A**ugmented-**G**eneration (D-RAG), as illustrated in Figure 2. Our approach integrates a GNN-based retriever and an LLM-based generator through a differentiable subgraph sampling and prompting module, enabling end-to-end training. Below, we detail these modules and the training strategy.

4.1 GNN-based Retriever

The graph retriever in D-RAG employs fact-wise probability factorization to model subgraph selection through independent fact selections. This transforms the complex subgraph probability into a product of simple binary selection probabilities for all facts:

$$p(g_{sub}) = \prod_{\tau_i \in g_{sub}} p(\tau_i) \prod_{\tau_j \notin g_{sub}} (1 - p(\tau_j)), \tag{2}$$

where each τ_i represents a fact with corresponding selection probability $p(\tau_i)$. A detailed derivation of this factorization is given in Appendix A.

To compute the selection probability, we design a triplet scoring mechanism that combines entity representations from ReaRev's GNN architecture (Mavromatis and Karypis, 2022) with a relation encoder through feature concatenation:

$$p(\tau_i) = \sigma(\text{MLP}(f_e(h_i) \,||\, f_r(r_i) \,||\, f_e(t_i))), \tag{3}$$

where $f_e(\cdot)$ is the entity encoder, $f_r(\cdot)$ is the relation encoder, and $\sigma(\cdot)$ denotes the sigmoid activation. Detailed specifications are provided in Appendix B.

4.2 LLM-based Generator

The LLM-based generator predicts answers through autoregressive decoding:

$$p_\gamma(a|g_{sub}, q) = \prod_{i=1}^{L_a} p_\gamma(a_i|a_{<i}, g_{sub}, q) \tag{4}$$

where L_a is the length of the ground-truth answer, and the subgraph and question are formatted through a structured prompt template:

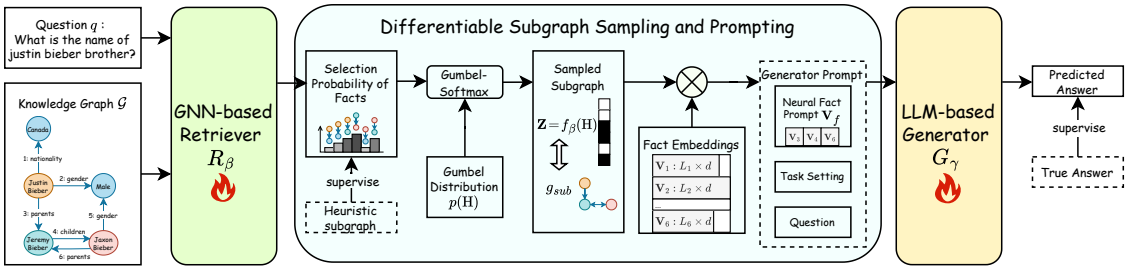

Figure 2: The overall approach of DRAG. 1) The GNN-based Retriever R_β processes the input knowledge graph $\mathcal{G}$ and assigns a selection probability to each fact with respect to the given question. 2) To enable a differentiable subgraph retrieval and prompting process, the approach first employs Gumbel-Softmax reparameterization to sample facts, forming a subgraph. This subgraph is then transformed into fact embeddings, which serve as prompts. 3) The LLM-based Generator G_γ processes the subgraph information, task setting, and question to infer the final answer.

```
Answer the question based
on the provided facts.
Question: <question>
Provided facts: <fact1><fact2> ...
Answer:
```

The ground-truth answer is represented as a list separated by vertical bars: `<Ans1>|<Ans2>|...|<AnsN>`. During inference, the generator produces answers in the same format, which can be easily parsed to obtain the final answer set. Complete prompt examples are detailed in Appendix C.

4.3 Differentiable Subgraph Sampling and Prompting

Our key innovation lies in constructing differentiable bridges across the retriever-generator interface through a differentiable formulation of the optimization objective and a novel subgraph sampling and prompting module. This end-to-end optimization enables the retriever to discover latent graph patterns that actively facilitate the generator's reasoning process.

4.3.1 Differentiable Formulation

The optimization objective of maximizing Equation 1 involves a summation problem with combinatorial complexity, which is generally intractable. To address this, we can alternatively optimize its evidence lower bound (ELBO) (Hoffman et al., 2013), formulated as:

$$\begin{aligned}\log p_\theta(a|q,\mathcal{G}) &= \mathbb{E}_{g_{sub}\sim r}\left[\log \frac{p_\theta(a, g_{sub}|q,\mathcal{G})}{r(g_{sub})}\right] \\ &\quad + D_{KL}(r(g_{sub}) \,||\, p_\theta(g_{sub}|a,q,\mathcal{G})) \\ &\geq \mathbb{E}_{g_{sub}\sim r}\left[\log \frac{p_\theta(a, g_{sub}|q,\mathcal{G})}{r(g_{sub})}\right],\end{aligned} \tag{5}$$

where $r(g_{sub})$ represents the variational distribution of the subgraph, and the inequality holds because the Kullback-Leibler divergence is non-negative. We specify the variational distribution $r(g_{sub})$ as the retriever's distribution $p_\beta(g_{\text{sub}}|q,\mathcal{G})$. The ELBO can be simplified as:

$$\begin{aligned}\log p_\theta(a|q,\mathcal{G}) &\geq \mathbb{E}_{g_{sub}\sim p_\beta}\left[\log \frac{p_\theta(a, g_{sub}|q,\mathcal{G})}{p_\beta(g_{sub}|q,\mathcal{G})}\right] \\ &= \mathbb{E}_{g_{sub}\sim p_\beta}\left[\log \frac{p_\gamma(a|g_{sub},q)p_\beta(g_{sub}|q,\mathcal{G})}{p_\beta(g_{sub}|q,\mathcal{G})}\right] \\ &= \mathbb{E}_{g_{sub}\sim p_\beta}\left[\log p_\gamma(a|g_{sub},q)\right],\end{aligned} \tag{6}$$

where p_β is modeled by the GNN-based retriever and p_γ is modeled by the LLM-based generator. This choice not only simplifies the optimization objective but also ensures the retriever operates without access to the answer information.

The optimization objective of our approach is formulated as the ELBO in Equation 6. When the subgraph distribution $p_\beta(g_{sub}|q,\mathcal{G})$ retrieved by the retriever closely approximates the posterior distribution $p_\theta(g_{sub}|q,a,\mathcal{G})$, the inequality in Equation 6 approaches equality, thereby tightening the bound to the original objective. Since the GNN-based retriever is jointly learned during training, this bound can theoretically achieve equality. The computation of this objective can be decomposed into two steps: subgraph sampling via p_β and prompt-based generation via p_γ. The following two sections present the refinements to make these steps differentiable.

4.3.2 Differentiable Subgraph Sampling

This section focuses on the differentiability of subgraph sampling. To enable gradient computation, we adopt the Gumbel-Softmax reparameterization trick (Jang et al., 2017; Maddison et al., 2017),

which approximates discrete sampling with a continuous relaxation. For the i-th fact in the KG, the retriever outputs a Bernoulli parameter $p_i = p_\beta(\tau_i)$, representing the probability of selecting this fact. To make the discrete Bernoulli sampling differentiable, we apply the Gumbel-Softmax trick:

$$\mathbf{z}_i^{\text{soft}} = \text{softmax}\begin{pmatrix} (\log p_i + \eta_{i1}) \,/\, t \\ (\log(1-p_i) + \eta_{i2}) \,/\, t \end{pmatrix}^T, \tag{7}$$

where η_{i1}, η_{i2} are independent samples from Gumbel(0,1)[1] and t is the temperature coefficient. Here, $z_i^{\text{soft}} \in \mathbb{R}^{1\times 2}$ represents the relaxed selection probability for fact τ_i.

The final binary selection indicator z_i is obtained through:

$$\mathbf{z}_i = \text{onehot}(\text{argmax}(\mathbf{z}_i^{\text{soft}})) + \mathbf{z}_i^{\text{soft}} - \text{SG}(\mathbf{z}_i^{\text{soft}}), \tag{8}$$

where SG denotes the stop-gradient operation. This formulation maintains differentiability while producing discrete one-hot vectors during forward propagation.

Let $\mathbf{Z} = f_\beta(\text{H}) = \left[\mathbf{z}_1; \mathbf{z}_2; \ldots; \mathbf{z}_{N_f}\right] \in \{0,1\}^{N_f \times 2}$ denote the complete subgraph selection matrix, where N_f is the total number of facts and H represents Gumbel noise samples. The selected subgraph can be equivalently represented as $g_{sub} = \{\tau_i | \mathbf{z}_i = [1,0]\}$. The training objective becomes:

$$\mathbb{E}_{\text{H}\sim p(\text{H})}\left[\log p_\gamma(a | f_\beta(\text{H}), q)\right], \tag{9}$$

where f_β denotes the complete sampling process parameterized by β. This reparameterization not only allows gradients to flow through the sampling process, but also transforms the expectation from a complex parameterized distribution p_β to a simple fixed distribution, making the optimization more tractable.

4.3.3 Differentiable Prompt Construction

After obtaining the parameterized subgraph representation $f_\beta(\text{H})$, where the first column of the output matrix encodes the selection probabilities, the subgraph needs to be transformed into LLM-compatible inputs while maintaining differentiability.

To construct differentiable prompts, we first convert each fact into its textual representation following the template `<head name>, <relation name>, <tail name>`. These textual forms are then encoded into token embeddings $\mathbf{V}_i \in \mathbb{R}^{L_i \times d}$, where L_i is the token length and $d$is the embedding dimension of the LLM-based generator.

The binary selection indicator $\mathbf{Z}_{i1}$ (the first element of the i-th row in $\mathbf{Z}$ determines whether fact τ_i is selected. We multiply each embedding $\mathbf{V}_i$ with its corresponding $\mathbf{Z}_{i1}$. When $\mathbf{Z}_{i1} = 0$, the corresponding embedding is completely masked out. All weighted embeddings are concatenated to form the neural prompt $\mathbf{V}_f$, allowing gradients from the LLM-based generator to propagate back to the GNN-based retriever parameters β through the chain:

$$\frac{\partial L}{\partial \beta} = \frac{\partial L}{\partial \mathbf{V}_f}\frac{\partial \mathbf{V}_f}{\partial \mathbf{Z}}\frac{\partial \mathbf{Z}}{\partial \beta}, \tag{10}$$

where L is the autoregressive loss of the LLM.

For multi-hop reasoning in KGQA, where retrieved facts should form structured reasoning paths, we arrange facts in ascending order of their selection probabilities to preserve the logical dependencies in the linearized prompt sequence.

4.4 Training Strategy

With the differentiable subgraph sampling and prompting module proposed above, D-RAG supports end-to-end training in principle. However, directly optimizing the full model from random initialization often leads to poor convergence, as the retriever might retrieve an irrelevant subgraph that misleads the generator. To address this, we adopt a two-phase training strategy.

In the first phase, the GNN-based retriever is pre-trained using heuristically constructed subgraphs g_{heur} to establish a reasonable initialization, preventing the retriever from retrieving irrelevant subgraphs. The pre-training loss is defined as:

$$L_1 = D_{KL}(p_{heur}(g_{sub}) \,||\, p_\beta(g_{sub})), \tag{11}$$

where p_{heur} represents the heuristic subgraph distribution (typically in one-hot form), and p_β is the retriever's predicted distribution.

In the second phase, both the retriever and the generator are trained jointly. Following Equation 9, the generation loss is defined as:

$$L_2 = -\mathbb{E}_{\text{H}\sim p(\text{H})}\left[\log p_\gamma(a | f_\beta(\text{H}), q)\right]. \tag{12}$$

The overall joint loss combines the retriever pre-training objective and the generation loss:

$$L_{\text{joint}} = \lambda L_1 + (1-\lambda) L_2, \tag{13}$$

[1]The cumulative distribution function of Gumbel(0,1) is $F(x) = \exp(-\exp(-x))$.

where the hyperparameter λ balances the retriever's adherence to heuristic subgraphs and its ability to predict answers.

5 Experiments

In our experiments, we aim to answer the following three research questions: **RQ1**. What is the overall performance of the proposed approach? **RQ2**. Is the end-to-end optimization of the retriever and the generator effective? **RQ3**. How do the various design details within the proposed approach influence the outcomes?

5.1 Experiment Settings

Datasets. The experimental evaluation was conducted on two benchmark datasets: WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018), both built upon the Freebase (Bollacker et al., 2008) knowledge graph. These datasets represent classical benchmarks for complex logical reasoning in KGQA. WebQSP contains relatively straightforward questions that typically require 1-2 hop reasoning chains, while CWQ presents more challenging scenarios involving 3-4 hop reasoning chains. Detailed specifications of the datasets are provided in Appendix D.

Baselines. D-RAG is compared with 15 baselines across three categories: 1) Graph reasoning methods that leverage graph structure for scoring-based answer inference; 2) LLM reasoning methods that perform reasoning with LLMs while not utilizing graph structure during retrieval; and 3) Graph-LLM methods that maintain dedicated graph-based retrieval while leveraging LLMs for reasoning. The details of each baseline are described in Appendix E.

Evaluation Metrics. Following previous works (Luo et al., 2024; Sun et al., 2024), D-RAG employs Hits@1 and F1 metrics for evaluation on WebQSP and CWQ. The evaluation process first parses LLM-generated answers into a list for comparison with the ground truth answers. The Hits@1 metric (also commonly denoted as Hit in LLM-based methods) measures whether any correct answer appears in the model's response, while F1 provides a balanced measure of precision and recall for comprehensive quality assessment. A detailed discussion of the evaluation metrics is provided in Appendix F.

Implementations. D-RAG employs the ReaRev (Mavromatis and Karypis, 2022) model with a fact prediction head as the GNN and utilizes the Llama3-8B-Instruct (Meta, 2024) as the LLM. Consistent with prior work (Mavromatis and Karypis, 2022; Luo et al., 2024), we assume that the entities mentioned in the questions (referred to as topic entities) have already been linked to the knowledge graph through entity linking (Yih et al., 2015). Based on these linked entities, heuristic subgraphs are extracted via two methods: shortest path between topic and answer entities, and SPARQL query parsing. We limit the maximum number of retrieved facts to 50 to fit the LLM's context window. Full implementation details are in Appendix G.

5.2 Main Results

To evaluate the overall effectiveness of D-RAG (**RQ1**), we compare it with state-of-the-art baselines on KGQA tasks. Table 1 presents the results, where "-" indicates the corresponding method does not report results for that metric. The D-RAG approach achieves state-of-the-art performance across both datasets. Specifically, on the WebQSP dataset, D-RAG achieves a 2.4% improvement in Hits@1 over the best-performing baseline SubgraphRAG, and outperforms DECAF by 2.3% in the F1 score. While some baselines like RoG achieve competitive Hits@1 (85.7%), their F1 scores (70.8%) lag substantially behind, suggesting they may achieve high recall at the cost of precision. For the more complex CWQ dataset, the proposed approach demonstrates a 1.0% advantage in Hits@1 compared to the state-of-the-art ToG approach, while surpassing GNN-RAG by 3.4% in the F1 score. Notably, methods like SubgraphRAG suffer from a significant performance drop on CWQ (F1 decreases from 70.6% to 47.2%). In contrast, D-RAG maintains robust performance across both datasets, demonstrating its superior generalization capability and balanced precision-recall trade-off.

Among other approaches, graph reasoning approaches (without LLM integration) underperform due to lacking advanced reasoning capabilities. While LLM reasoning methods leverage generative power for knowledge exploration, they suffer from hallucination and structural misinterpretation. Graph-LLM approaches demonstrate superior performance by integrating structural patterns with LLM inference. This empirically validates two observations: (1) LLM integration fundamentally

Type	Method	WebQSP		CWQ	
		Hits@1	F1	Hits@1	F1
Graph Reasoning	Graftnet (Sun et al., 2018)	66.4	-	32.8	-
	NSM (He et al., 2021)	68.7	62.8	47.6	42.4
	SR+NSM (Zhang et al., 2022)	68.9	64.1	50.2	47.1
	ReaRev (Mavromatis and Karypis, 2022)	76.4	70.9	52.9	-
	UniKGQA (Jiang et al., 2023b)	75.1	70.2	50.7	48.0
	NuTrea (Choi et al., 2023)	77.4	72.7	53.6	49.5
LLM Reasoning	LLama3-8B (Meta, 2024)	59.8	45.7	30.8	27.6
	StructGPT (Jiang et al., 2023a)	72.6	-	-	-
	DECAF (DPR + FiD-large) (Yu et al., 2023)	80.7	77.1	67.0	-
	ToG (GPT4) (Sun et al., 2024)	82.6	-	68.5	-
	RoG (joint) (Luo et al., 2024)	85.7	70.8	62.6	56.2
Graph-LLM	G-Retriever (He et al., 2024)	70.1	-	-	-
	EtD (ChatGPT) (Liu et al., 2024a)	82.5	-	62.0	-
	GNN-RAG (Mavromatis and Karypis, 2024)	85.7	71.3	66.8	59.4
	SubgraphRAG (Llama3.1-8B) (Li et al., 2025)	86.6	70.6	57.0	47.2
	D-RAG	**89.0**	**79.4**	**69.5**	**62.8**

Table 1: Performance comparison with different baselines on WebQSP and CWQ.

enhances reasoning capacity, and (2) structural-semantic synergy produces superior outcomes compared to isolated approaches.

5.3 Ablation Study

Training Method	WebQSP		CWQ	
	Hits@1	F1	Hits@1	F1
D-RAG	**88.0**	**78.6**	**69.5**	**62.8**
REINFORCE	84.5	69.5	62.6	56.2
D-RAG w/o e2e				
Dynamic Cascade	86.9	76.7	69.0	62.0
Static Cascade	87.1	77.0	67.7	61.1
Isolation	85.3	73.1	63.9	28.1

Table 2: Ablation studies of different training methods.

To answer **RQ2** regarding the effectiveness of end-to-end optimization between the retriever and the generator, we conduct an ablation study with four variants: 1) *REINFORCE*, which optimizes both modules jointly using the REINFORCE algorithm (Williams, 1992) with variance reduction; 2) *Dynamic Cascade*, where both modules are trained simultaneously with the generator using real-time retriever outputs, but without gradient backpropagation; 3) *Static Cascade*, where we train the retriever first and then optimize the generator with the frozen retriever; 4) *Isolation*, where both modules are trained independently with the generator using only heuristic subgraphs.

The experimental results are presented in Table 2. From the results, three key observations emerge. First, D-RAG outperforms all baseline approaches, demonstrating the superiority of end-to-end joint training. Second, in *Dynamic Cascade* where gradient flow from the generator to the retriever is disabled, we observe approximately 1% degradation in Hits@1 and F1 across both datasets. This indicates that our end-to-end design enables direct supervision of the retriever using ground-truth answer signals, effectively mitigating noise inherent in heuristic subgraphs. Third, both *REINFORCE* and *Isolation* show significant performance drops, but for different reasons. *REINFORCE* suffers from the inherent instability of policy gradient, while *Isolation* demonstrates the necessity of module interaction. In contrast, D-RAG enables stable and efficient end-to-end optimization. Detailed analyses of module interaction patterns, training dynamics, qualitative case studies, and training efficiency analysis can be found in Appendices H, I, J, and K, respectively.

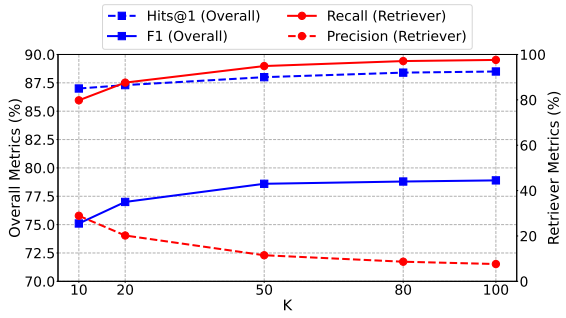

Figure 3: The impact of the number of retrieved facts (K) on D-RAG on the WebQSP dataset.

5.4 Detail Analysis

To better understand how different design choices affect our approach's effectiveness (**RQ3**), we conduct a series of analyses on three critical factors: the number of retrieved facts, variations in heuristic subgraphs, and the order of facts.

Number of Retrieved Facts. The number of retrieved facts (K) serves as a crucial hyperparameter. Figure 3 shows how K affects both the overall performance (Hits@1 & F1) and the retriever's metrics (recall & precision). As K increases, recall improves steadily (reaching 97.6%) while precision declines (to 7.63%). The overall performance metrics show substantial gains from $K = 10$ to $K = 50$, but plateau beyond $K = 50$, suggesting that retrieving 50 facts achieves a good balance between coverage and efficiency.

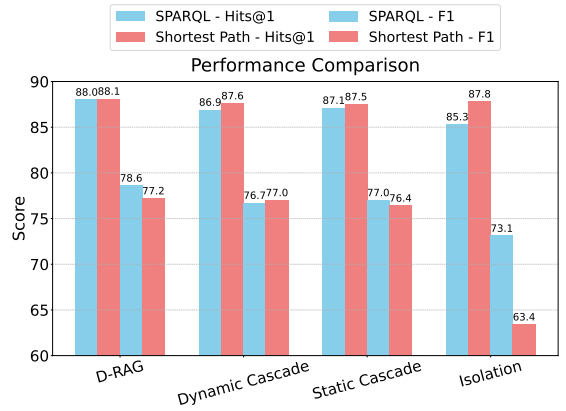

Figure 4: Performance comparison of different training methods under two heuristic subgraph acquisition approaches on the WebQSP dataset.

Different Heuristic Subgraphs. To analyze the impact of different heuristic subgraph supervision signals, we compare two methods for generating them: *SPARQL* and *Shortest Path*. As shown in Figure 4, D-RAG consistently outperforms all variants under both supervision signals, achieving the highest scores in both Hits@1 and F1. The minimal performance gap between *SPARQL* and *Shortest Path* methods further demonstrates D-RAG's robustness to different types of supervision signals.

Fact Order	Hits@1	F1
ascent	**89.0**	**79.4**
descent	88.0	78.6
default	88.0	77.3
random	86.7	76.6

Table 3: The impact of fact ordering on D-RAG on the WebQSP dataset.

Fact Ordering. Since the order of input facts can influence LLM generation (Liu et al., 2024b), we compare four ordering strategies: 1) *Ascent*: Facts are arranged in ascending order of retrieval probabilities; 2) *Descent*: The reverse of ascent, with facts ordered from high to low probabilities; 3) *Default*: Facts are arranged in a fixed order, regardless of their probabilities; 4) *Random*: Facts are shuffled randomly during both training and inference.

As shown in Table 3, ascending order performs best, while random ordering yields the poorest results. This suggests that deliberately structuring facts is more effective than pursuing order-agnostic generation, though more sophisticated ordering strategies remain to be explored in future work.

Conclusion

In this paper, we presented D-RAG, a novel differentiable approach for KGQA that enables end-to-end optimization between the retriever and the generator. By reformulating the optimization objective as a tractable expectation over a subgraph distribution and designing a differentiable subgraph sampling and prompting module based on Gumbel-Softmax reparameterization, D-RAG achieves superior performance on standard KGQA benchmarks. Experimental results on WebQSP and CWQ demonstrated that our approach outperforms state-of-the-art methods with substantial improvements, validating the effectiveness of end-to-end optimization in RAG-based KGQA systems.

Limitations

Despite the effectiveness of D-RAG, we acknowledge several limitations of our current approach. First, our approach relies on entity linking results without considering potential errors in this preprocessing step. Second, our end-to-end optimization approach is limited to open-source language models and cannot be directly applied to closed-source API-based models.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Hyeong Kyu Choi, Seunghun Lee, Jaewon Chu, and Hyunwoo J. Kim. 2023. Nutrea: Neural tree search for context-guided multi-hop KGQA. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

DeepSeek. 2025. Deepseek-r1 release.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Improving multi-hop knowledge base question answering by learning intermediate supervision signals. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pages 553–561. ACM.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *CoRR*, abs/2402.07630.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John W. Paisley. 2013. Stochastic variational inference. *J. Mach. Learn. Res.*, 14(1):1303–1347.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023a. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Xin Zhao, and Ji-Rong Wen. 2023b. Unikgqa: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Complex knowledge base question answering: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(11):11196–11215.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Mufei Li, Siqi Miao, and Pan Li. 2025. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *Preprint*, arXiv:2410.20724.

Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. 2024. KAG: boosting llms in professional domains via knowledge augmented generation. *CoRR*, abs/2409.13731.

Valentin Liévin, Andreas Geert Motzfeldt, Ida Riis Jensen, and Ole Winther. 2023. Variational open-domain question answering. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 20950–20977. PMLR.

Zhutian Lin, Junwei Pan, Shangyu Zhang, Ximei Wang, Xi Xiao, Shudong Huang, Lei Xiao, and Jie Jiang. 2024. Understanding the ranking loss for recommendation with sparse user feedback. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 5409–5418. ACM.

Guangyi Liu, Yongqi Zhang, Yong Li, and Quanming Yao. 2024a. Explore then determine: A gnn-llm

synergy framework for reasoning over knowledge graph. *Preprint*, arXiv:2406.01145.

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

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, and Jian Guo. 2024. Think-on-graph 2.0: Deep and interpretable large language model reasoning with knowledge graph-guided retrieval. *CoRR*, abs/2407.10805.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Costas Mavromatis and George Karypis. 2022. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Costas Mavromatis and George Karypis. 2024. GNN-RAG: graph neural retrieval for large language model reasoning. *CoRR*, abs/2405.20139.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

OpenAI. 2024. Learning to reason with llms.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *CoRR*, abs/2408.08921.

Saloni Potdar, Daniel Lee, Omar Attia, Varun Embar, De Meng, Ramesh Balaji, Chloe Seivwright, Eric Choi, Mina H. Farid, Yiwen Sun, and Yunyao Li. 2025. Comprehensive evaluation for a large scale knowledge graph question answering service. *Preprint*, arXiv:2501.17270.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Devendra Singh Sachan, Siva Reddy, William L. Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 25968–25981.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. 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 (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. 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)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases.

In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Hamed Zamani and Michael Bendersky. 2024. Stochastic RAG: end-to-end retrieval-augmented generation through expected utility maximization. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2641–2646. ACM.

Jing Zhang, Xiaokang Zhang, Jifan Yu, Jian Tang, Jie Tang, Cuiping Li, and Hong Chen. 2022. Subgraph retrieval enhanced model for multi-hop knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5773–5784, Dublin, Ireland. Association for Computational Linguistics.

A Probability Factorization Analysis

In this section, we first prove the validity of Equation 2, followed by a discussion on the rationale behind fact-wise factorization.

The factorization of subgraph probability represents an approximation of the complex probability distribution, with an underlying assumption that the selection of each fact is independent. Consider a knowledge graph with N_f facts, where each fact has two possible states (selected or not selected), resulting in 2^{N_f} possible subgraphs. The sum of probabilities over all possible subgraphs can be expressed as:

$$
\begin{aligned}
&\sum_{g_{sub}} p(g_{sub}) \\
&= \sum_{g_{sub}} \prod_{\tau_i \in g_{sub}} p(\tau_i) \prod_{\tau_j \notin g_{sub}} (1 - p(\tau_j)) \\
&= \sum_{\tau_1} \sum_{\tau_2} \cdots \sum_{\tau_{N_f}} \prod_{i=1}^{N_f} p(\tau_i)^{\mathbb{I}(\tau_i)} (1 - p(\tau_i))^{1-\mathbb{I}(\tau_i)} \\
&= \prod_{i=1}^{N_f} \sum_{\mathbb{I}(\tau_i) \in \{0,1\}} p(\tau_i)^{\mathbb{I}(\tau_i)} (1 - p(\tau_i))^{1-\mathbb{I}(\tau_i)} \\
&= \prod_{i=1}^{N_f} (p(\tau_i) + (1 - p(\tau_i))) = 1
\end{aligned}
$$

where the third row follows from the fact that summing over all subgraphs is equivalent to considering both possibilities (selected or not selected) for each fact independently. $\mathbb{I}(\tau_i)$ is an indicator function that equals 1 when fact τ_i is included in the subgraph and 0 otherwise. The final result of 1 validates the probability formulation in Equation 2.

Beyond fact-wise factorization, node-level and path-wise granularities are also common choices for probability decomposition. Path-wise granularities, however, face combinatorial complexity challenges, which explains why direct modeling of subgraph probability is computationally intractable. Node-wise granularity, on the other hand, disregards relation information between entities and fails to handle multi-edge scenarios. These limitations motivate our choice of fact-wise factorization. To address the potential dependencies between fact selections that may be overlooked by the independence assumption implicit in factorization, we employ a GNN-based retriever. The inherent capability of GNNs to capture graph structural information helps mitigate the independence assumption, as the internal parameters of GNN can effectively encode the correlations between facts.

B Specific design of GNN-based Retriever

B.1 Module

For the GNN-based retriever, D-RAG adopts ReaRev (Mavromatis and Karypis, 2022) as the core architecture, which consists of three primary modules:

- The Instruction Module employs Sentence-BERT (Reimers and Gurevych, 2019) as its Language Model (LM) encoder to transform queries into instructions;
- The Graph Reasoning Module initializes and updates node representations through message passing, considering the relationship between instructions and nodes;
- The Instruction Update Module refines instructions based on the node representations and predicted terminal node distributions.

In our implementation, the node encoder corresponds to the output of the Graph Reasoning Module, while the relation encoder refers to the LM encoder and MLP projection components used in the node initialization process.

B.2 Loss design of graph neural network

As shown in Equation 11 of the main text, the loss function L_1 for training the GNN-based retriever is formulated as:

$$D_{KL}(p_{heur}(g_{sub})|p_\beta(g_{sub})) = -\sum_{\tau \in g_{sub}} \log p_\beta(\tau) = L_{\text{BCE}} \quad (14)$$

This can be implemented using PyTorch's BCE (Binary Cross Entropy) weighted loss [2]. Inspired by the work of (Lin et al., 2024), to address the sparsity of positive examples in knowledge graph link classification tasks, we further incorporate a rank loss:

$$L_{\text{rank}} = -\frac{1}{N_+ N_-} \sum_{i=1}^{N_+} \sum_{j=1}^{N_-} \log \sigma(p(\tau_i) - p(\tau_j)), \quad (15)$$

where N_+ and N_- denote the number of positive and negative examples, respectively, τ_i represents a positive example, τ_j represents a negative example, and $\sigma(\cdot)$ is the sigmoid function. This ranking loss generates larger gradients on sparse samples, effectively complementing the BCE loss and enhancing the model's classification capability.

The total loss of the GNN-based retriever is a weighted combination of these two losses:

$$L_1 = \rho L_{\text{BCE}} + (1 - \rho) L_{\text{Rank}}, \quad (16)$$

where we empirically set $\rho = 0.7$ to balance between the BCE loss and the ranking loss.

C Prompts

Figure 5 illustrates the full input prompt received by the LLM-based generator, which consists of three components: task configuration, question, and subgraph. The task configuration and question components are presented in natural language text format, while the subgraph is represented in embedding form, corresponding to the neural fact prompt in D-RAG.

D Datasets

D-RAG evaluates on two benchmark KGQA datasets: WebQuestionSP (WebQSP) (Yih et al., 2016) and Complex WebQuestions (CWQ) (Talmor and Berant, 2018). Following previous works (Luo et al., 2024; He et al., 2021), the same train and test splits are adopted for fair comparison. The datasets are analyzed from two perspectives: basic statistics and reasoning complexity.

[2] https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html

The overall statistics of both datasets are summarized in Table 4, including the number of samples in training, validation and test sets.

Table 5 shows the distribution of reasoning hops required for answering questions, indicating the logical complexity of questions in each dataset. The hop counting method analyzes the path length from topic entities to answer entities in SPARQL queries. For WebQSP, hop counts are determined precisely as most questions involve single topic entities with equal path lengths from topic to answer entities. For CWQ, we compute fuzzy hop counts due to frequent multi-topic scenarios. When SPARQL queries represent constrained graphs rather than simple chains, we take the maximum path length among all topic-to-answer paths as the final hop count.

Datasets	#Train	#Validate	#Test
WebQSP	2826	246	1,628
CWQ	27,639	3519	3531

Table 4: Statistics of the datasets.

Datasets	1-hop	2-hop	3-hop	≥4-hop
WebQSP	2906	1776	8	8
CWQ	6743	19408	5911	2627

Table 5: Statistics of reasoning hop distribution in WebQSP and CWQ.

E Baselines

The D-RAG approach is compared with the 15 baselines grouped into three categories: 1) Graph reasoning methods; 2) LLM reasoning methods; and 3) Graph-LLM methods. The details of each baseline are described as follows:

Graph Reasoning Methods.

- Graftnet (Sun et al., 2018) performs question answering by propagating features through a heterogeneous graph that fuses knowledge bases and text documents.
- NSM (He et al., 2021) leverages language models' bidirectional reasoning capabilities for multi-hop question answering.
- SR+NSM (Zhang et al., 2022) introduces a trainable path-wise subgraph retriever that decouples retrieval from reasoning.

Complete Generator Prompt

Answer the question based on the provided facts.
Question: what does jamaican people speak
Provided facts:
Jamaica, location.country.official_language, Jamaican English
Jamaica, location.country.languages_spoken, Jamaican English
Jamaica, location.country.languages_spoken, Jamaican Creole English Language
Jamaica, location.country.currency_used, Jamaican dollar
Jamaica, location.country.form_of_government, Democracy
Jamaica, location.country.form_of_government, Parliamentary system
Jamaica, base.locations.countries.continent, North America
Jamaica, location.country.form_of_government, Constitutional monarchy
Grenada, location.country.official_language, English Language
Bermuda, location.country.official_language, English Language
Belize, location.country.official_language, English Language
Turks and Caicos Islands, location.country.official_language, English Language
Bahamas, location.country.official_language, English Language
Cayman Islands, location.country.official_language, English Language
Puerto Rico, location.country.official_language, English Language
Grenada, location.country.languages_spoken, English Language
Bermuda, location.country.languages_spoken, English Language
Costa Rica, location.country.languages_spoken, Jamaican Creole English Language
Belize, location.country.languages_spoken, English Language
Turks and Caicos Islands, location.country.languages_spoken, English Language
Answer:

Figure 5: The complete input prompt for the LLM-based generator, incorporating 20 facts.

- ReaRev (Mavromatis and Karypis, 2022) adaptively refines reasoning instructions using knowledge graph context and executes them through a BFS-guided neural network.

- UniKGQA (Jiang et al., 2023b) unifies retrieval and reasoning stages in KGQA through a shared PLM-based architecture and joint pre-training strategy.

- NuTrea (Choi et al., 2023) utilizes tree search-based message passing to explore future paths with RF-IEF node embeddings that capture global KG context.

LLM Reasoning Methods.

- LLama3-8B (Meta, 2024) performs direct reasoning without fact retrieval by leveraging its pre-trained knowledge.

- StructGPT (Jiang et al., 2023a) enhances LLM reasoning by iteratively collecting evidence from structured data through specialized interfaces before performing reasoning steps.

- DECAF (DPR + FiD-large) (Yu et al., 2023) improves KB question answering by combining logical form generation with direct answer prediction, while simplifying the process through text-based retrieval.

- ToG (GPT4) (Sun et al., 2024) enables LLMs to perform traceable reasoning by iteratively exploring knowledge graphs through beam search.

- RoG (joint) (Luo et al., 2024) enhances LLM reasoning by leveraging KG structure to generate faithful reasoning paths through a planning-retrieval-reasoning framework.

Graph-LLM Methods.

- G-Retriever (He et al., 2024) enables conversational graph interaction by combining GNNs, LLMs, and RAG through Prize-Collecting Steiner Tree optimization.

- EtD (ChatGPT) (Liu et al., 2024a) combines GNNs for efficient knowledge exploration with frozen LLMs for final answer determination, creating a resource-efficient framework for KGQA.

- GNN-RAG (Mavromatis and Karypis, 2024) combines GNNs for subgraph reasoning and path extraction with LLMs for natural language understanding in a RAG framework.

- SubgraphRAG (Llama3.1-8B) (Li et al., 2025) enhances KG-based RAG by implementing efficient subgraph retrieval with flexible size control and directional structural encoding.

F Discussion on Evaluation Metrics

The evaluation procedure varies across different methods. While node prediction and graph query approaches produce direct answers requiring no additional processing, LLM-based methods often generate responses containing multiple predicted answers. This characteristic of LLMs explains why many recent works prefer the term Hit over Hits@1, as the evaluation focuses on the presence of correct answers within the complete generated response rather than strictly the first position.

G Implementation Details

We train separate models for CWQ and WebQSP datasets. The training process consists of two stages: GNN pre-training and joint training. For the GNN pre-training stage, we train the model for 20 epochs and select the checkpoint with the lowest validation loss. In the joint training stage, we train for 18 epochs and select the final model based on the highest Hits@1 score on the validation set.

For model optimization, we apply different strategies to the GNN and LLM. The GNN undergoes full parameter fine-tuning with a learning rate of 5e-5, while the LLM is fine-tuned using LoRA with a learning rate of 1e-5. The LoRA hyperparameters are set as: lora_r=8, lora_alpha=16, and dropout=0.05, targeting the q_proj and v_proj modules. We use AdamW optimizer with weight decay of 0.001. Other training hyperparameters include a batch size of 16, one warmup epoch, and a cosine learning rate scheduler. In the joint training objective, we set both the loss weight λ and the Gumbel-Softmax temperature to 0.5.

During inference, we restrict the maximum number of retrieved facts to 50. The model selects the top-K facts with probabilities higher than 0.5, arranging them in ascending order of probability for LLM generation. The maximum length for LLM-generated responses is limited to 128 tokens.

All experiments are conducted on two NVIDIA A100-80GB GPUs. During each training epoch, we process the entire training set for WebQSP, while for CWQ we randomly sample 5,000 examples from its training set. The joint training stage takes approximately 16.5 hours for CWQ and 9.5 hours for WebQSP.

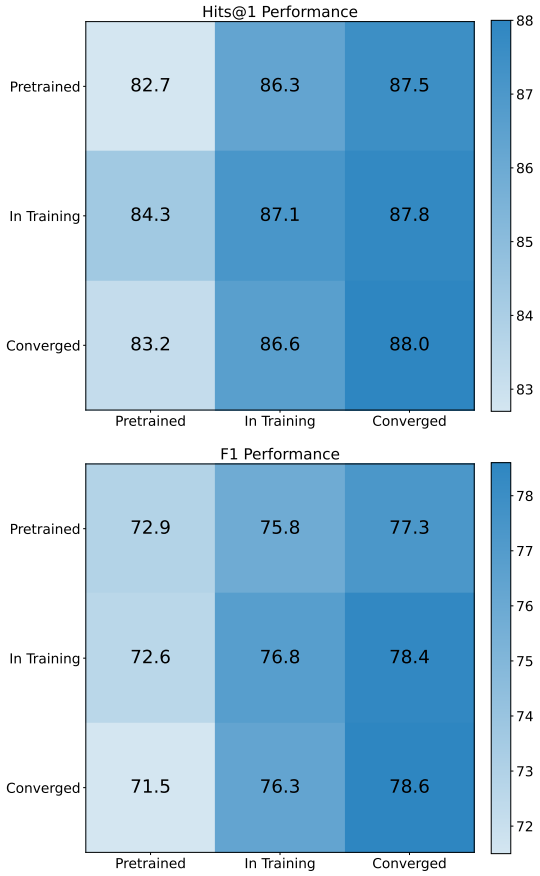

Figure 6: Performance comparison of different retriever-generator checkpoint combinations on the WebQSP test set. The x-axis represents LLM-based generator checkpoints and y-axis represents GNN-based retriever checkpoints. "Pretrained" indicates the initial state (pretrained GNN and base LLM), "In Training" represents checkpoints at epoch 8, and "Converged" represents the final checkpoints at epoch 18.

H Collaborative Performance of Retriever and Reasoner Under Different Checkpoints

Figure 6 shows how the retriever and the generator work together at different training stages, with all results measured on the WebQSP test set. The performance consistently improves throughout the training process, with Hits@1 increasing from 82.7% to 88.0% and F1 from 72.9% to 78.6%, demonstrating the effectiveness of end-to-end training. The converged generator (epoch 18) demonstrates remarkable robustness across different retriever checkpoints, which we attribute to its exposure to varied retrieval results during end-to-end training. Additionally, we observe that each gener-

ator checkpoint achieves its optimal performance when paired with its corresponding retriever checkpoint, indicating the development of complementary capabilities between the two modules. Furthermore, the results suggest that the generator plays a more crucial role in the overall performance: improvements in the generator checkpoint lead to more substantial gains compared to retriever improvements, while maintaining good performance even with less optimal retrieval results.

I Dynamics of Performance Metrics During Training Iterations

Figure 7 illustrates the training dynamics of different training methods on the WebQSP validation set. The *Isolation* method achieves the highest performance due to its use of high-quality heuristic supervision signals with minimal noise. While theoretically the GNN-based retriever could surpass these heuristic subgraphs, current GNN-based module inevitably introduces more noise compared to the carefully constructed heuristic supervision, making the *Isolation* performance reflect a practical upper bound for D-RAG's achievable performance. In contrast, *REINFORCE* exhibits significant instability, a common challenge of policy gradient methods. This instability is particularly pronounced in the early stages where the generator's limited capabilities exacerbate the issue, and although the performance improves in later epochs as the generator becomes more capable, the overall convergence remains slower than other training methods. This highlights the inherent challenges of reinforcement learning in such scenarios. Among the remaining training methods, D-RAG demonstrates both stability and superior performance compared to cascade variants, reaching higher final scores than *Dynamic Cascade* and *Static Cascade*. This advantage can be attributed to two factors: first, the end-to-end training allows the retriever to leverage the LLM's semantic understanding for more generation-friendly pattern retrieval; second, compared to *Static Cascade*, the dynamic nature of both D-RAG and *Dynamic Cascade* enables the generator to adapt to varied retrieval qualities during training, enhancing its robustness. These results collectively demonstrate that D-RAG successfully combines training stability with strong performance.

J Case Studies

Table 6 further demonstrates the specific retrieval and generation results of a 2-hop question under different experimental settings. We conduct comparative analyses across five configurations: D-RAG, D-RAG w/o e2e, Cascade, Isolation, and LLM-only (direct LLM generation).

All configurations except LLM-only retrieve facts from the KG, with the table presenting the top-8 facts with highest selection probabilities in descending order. The D-RAG configuration successfully retrieves all relevant facts and generates correct answers through its generation process. In contrast, D-RAG w/o e2e only captures partial facts (specifically second-hop facts), resulting in an incomplete answer. Both Cascade and Isolation approaches completely fail to retrieve pertinent facts.

Interestingly, we observe that D-RAG's retrieval results group together the corresponding 1-hop and 2-hop facts, suggesting that D-RAG can effectively identify subgraph patterns that facilitate LLM-based generation. This structural organization in the retrieved evidence appears crucial for successful multi-hop reasoning.

K Training Efficiency Analysis

The time complexity analysis illustrated in Table 7 shows that D-RAG requires approximately 52 minutes per epoch, which is about 13-14% more time than both cascade training methods and the sum of isolated training times (all around 45-46 minutes). This moderate increase in computational cost is primarily attributed to the additional overhead of maintaining end-to-end gradient flow between the retriever and the generator modules. The similar time requirements between cascade variants and combined isolated training suggest that the primary computational bottleneck lies in the basic operations of both modules, rather than in their interaction patterns.

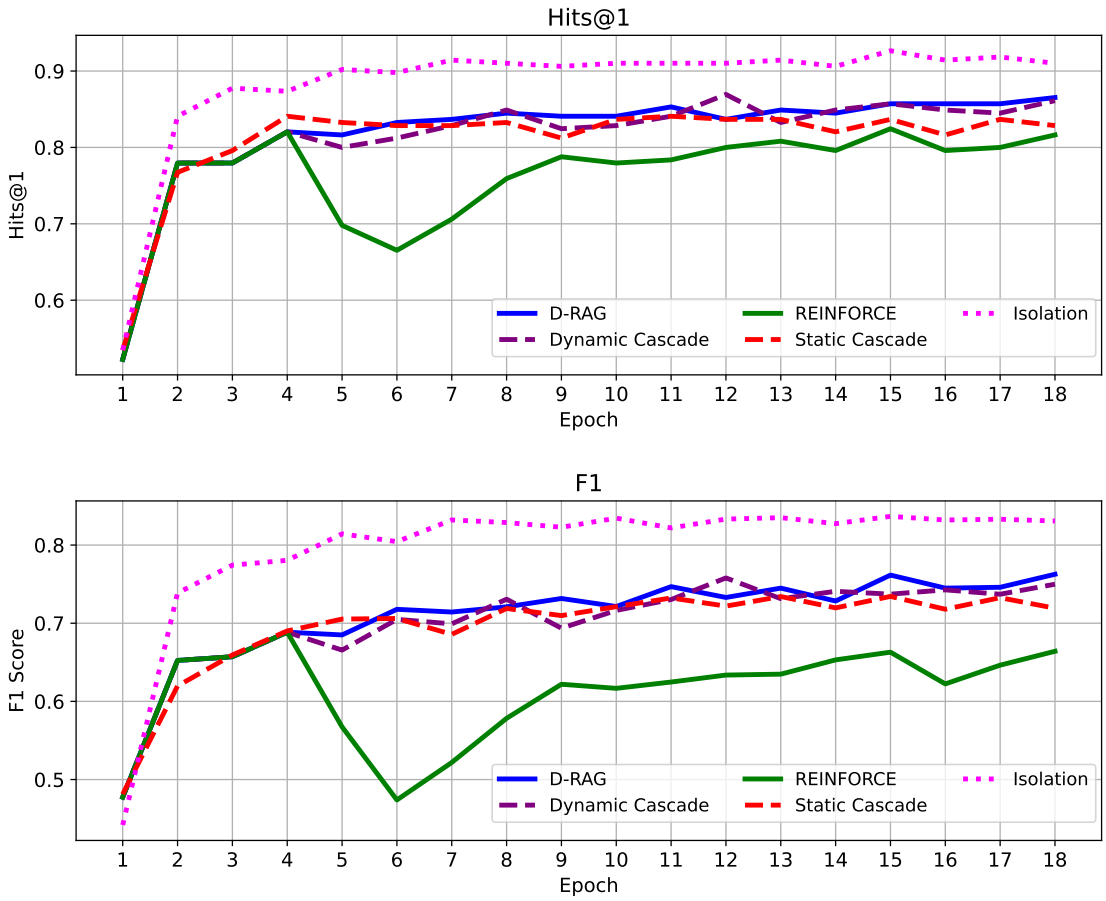

Figure 7: Training dynamics comparison of different training methods on WebQSP validation set.

Question	Which of the following does australia export the most?
True Answers	Energy industry | Agriculture
True Inference Chain	location.statistical_region.major_exports -> location.imports_exports_by_industry.industry
D-RAG	Provided facts: m.0cnqyp5, location.imports_exports_by_industry.industry, Energy industry m.0cnqyp9, location.imports_exports_by_industry.industry, Agriculture Australia, location.statistical_region.major_exports, m.0cnqyp5 Australia, location.statistical_region.major_exports, m.0cnqyp9 m.04c6slk, location.imports_and_exports.exported_to, Kiribati m.04bcv6n, location.imports_and_exports.exported_to, Tuvalu Australia, location.statistical_region.places_exported_to, m.04bcv6n Australia, location.statistical_region.places_exported_to, m.04c6slk Answer: Agriculture | Energy industry
D-RAG w/o e2e	Provided facts: Australia, location.country.currency_used, Australian dollar m.0cnqyp5, location.imports_exports_by_industry.industry, Energy industry m.0cnqyp9, location.imports_exports_by_industry.industry, Agriculture m.04c6slk, location.imports_and_exports.exported_to, Kiribati m.04bcv6n, location.imports_and_exports.exported_to, Tuvalu Australia, location.statistical_region.religions, m.02_75rz Australia, location.statistical_region.religions, m.02_75sg Australia, location.statistical_region.religions, m.02_75rb Answer: Agriculture
Cascade	Provided facts: m.02n_b7n, location.adjoining_relationship.adjoins, Timor-Leste m.02n_b76, location.adjoining_relationship.adjoins, New Caledonia m.02n_b6b, location.adjoining_relationship.adjoins, Vanuatu m.02n_b6s, location.adjoining_relationship.adjoins, Papua New & Guinea m.02n_b7f, location.adjoining_relationship.adjoins, Indonesia m.02n_b6k, location.adjoining_relationship.adjoins, Solomon Islands m.02n_b6_, location.adjoining_relationship.adjoins, New Zealand Australia, location.country.currency_used, Australian dollar Answer: Papua New Guinea
Isolation	The provided facts are exactly the same as cascade scenario Answer: Agriculture
LLM-only	According to the Australian Bureau of Statistics (ABS), in 2020-21, Australia's top exports by value were: 1. Iron ore: $63.4 billion 2. Coal: $24.4 billion 3. Gold: $14.4 billion 4. Liquefied natural gas (LNG): $12.4 billion 5. Refined petroleum products: $6.4 billion So, Australia exports the most iron ore.

Table 6: The Comparison of results from different training methods. For each method, we show both the retrieved facts and the final answer. The answers are highlighted in red background. In the retrieved facts, we use color coding to indicate the facts that align with the correct reasoning path: green for the first hop and yellow for the second hop.

Training Method	Time (minutes)
D-RAG	52.1 ± 5.6
Dynamic Cascade	46.1 ± 3.4
Static Cascade	$44{:}9 \pm 2.2$
Isolation (retriever)	27.0 ± 0.9
Isolation (generator)	19.0 ± 1.2

Table 7: Training time per epoch on CWQ with 5,000 random samples. Time variations ($\pm$) indicate the observed ranges across multiple epochs.