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

Guangze Gao[1,2], Zixuan Li[3], Chunfeng Yuan[1,2], Jiawei Li[1,2], Jianzhuo Wu[1,2], Yuehao Zhang[1,2], Xiaolong Jin[3], Bing Li [1,2,4], Weiming Hu[1]

[1]Beijing Key Laboratory of Super-Intelligent Security of Multi-Modal Information, MAIS, CASIA

[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

[3]Key Laboratory of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences [4] People AI, Inc.

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, 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. Via reformulating the optimization objective as an expectation over a subgraph distribution with respect to answer generation likelihood, D-RAG makes the joint optimization feasible. Specifically, it implements this joint optimization through a differentiable subgraph sampling and prompting module that integrates Gumbel-Softmax reparameterization for sampling and a neural prompt construction process that fuses semantic and structural information. Experimental results on WebQSP and CWQ demonstrate that D-RAG not only outperforms state-of-the-art approaches, but also effectively reduces retrieval noise in the process.

1 Introduction

Knowledge Graph Question Answering (KGQA) aims to automatically answer natural language questions via well-structured facts 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, 2024b; Meta, 2024) have shown strong capabilities in natural language understanding and reasoning, some recent approaches (Peng et al., 2024; Luo et al., 2024a; 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.

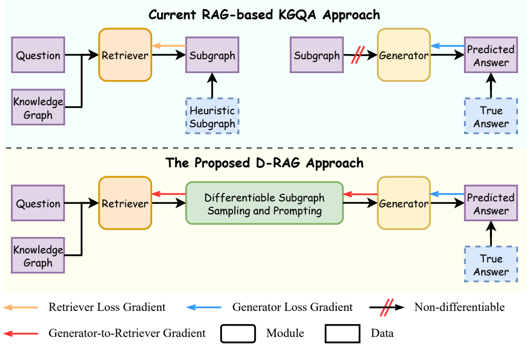


Figure 1: Comparison between the current RAG-based KGQA approaches and the proposed D-RAG approach. The red arrows highlight the end-to-end gradient flow.

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, current approaches (Luo et al., 2024a; 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 sequential optimization leads to sub-optimal performance for the complete system. Specifically,

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 35387–35406
November 4-9, 2025 ©2025 Association for Computational Linguistics

the generator's semantic understanding capabilities cannot guide the retriever, while the retriever cannot effectively communicate structural knowledge in a way the generator can optimally utilize.

To address above limitations, we propose the **D**ifferentiable **R**etrieval-**A**ugmented **G**eneration (D-RAG) for KGQA. First, we reformulate the optimization objective as an expectation over a subgraph distribution with respect to answer generation likelihood, making the joint optimization tractable. Second, we develop a differentiable subgraph sampling and prompting module that achieves end-to-end training. In the subgraph sampling step, D-RAG transforms discrete subgraph selection into differentiable fact-level sampling using the Gumbel-Softmax reparameterization trick (Jang et al., 2017; Maddison et al., 2017). In the prompt construction step, D-RAG converts the sampled subgraph into LLM-compatible prompts that fuse both semantic and structural information while maintaining gradient flow throughout the entire pipeline. This end-to-end optimization creates a synergistic relationship where the generator's semantic understanding informs retrieval quality, and the retriever provides structurally meaningful information that enhances the generator's reasoning capabilities.

Experimental results on WebQSP and CWQ show that D-RAG outperforms the state-of-the-art approaches by 2.5% and 1.8% on Hits@1, and by 3.4% and 4.4% on the F1 scores, respectively. These improvements stem from the end-to-end optimization strategy, which effectively reduces retrieval noise and enhances answer generation quality.

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 optimization objective as a tractable expectation over subgraph distributions and develop a differentiable subgraph sampling and prompting module. This module combines Gumbel-Softmax reparameterization for differentiable sampling with neural prompt construction that integrates both semantic and structural information, establishing an effective end-to-end optimization framework for KGQA.

- 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 the proposed 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). While SP-based methods parse questions into formal queries for execution, IR-based methods retrieve relevant subgraphs for answer ranking or generation. D-RAG falls into the latter category.

Traditional IR-based approaches typically learn entity and relation representations for answer ranking using network architectures such as graph neural networks (Sun et al., 2018; He et al., 2021; Zhang et al., 2022), which we categorize as Graph Reasoning methods. The emergence of LLMs has led to RAG-based approaches that leverage LLMs' powerful reasoning capabilities for answer generation. These RAG-based approaches can be divided into two groups: LLM Reasoning methods that primarily rely on LLMs for both subgraph retrieval and answer generation (Luo et al., 2024a; Jiang et al., 2023a; Sun et al., 2024; Ma et al., 2024; Luo et al., 2024b), and Graph-LLM methods that address LLMs' limitations in processing graph-structured data (Guo et al., 2023; Guan et al., 2025) by incorporating graph-specific techniques during retrieval while using LLMs for reasoning (He et al., 2024; Li et al., 2025; Mavromatis and Karypis, 2024; Liu et al., 2024a).

Despite the promise of these RAG-based approaches, a critical limitation is their lack of end-to-end training capabilities. While 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 the proposed approach, 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 for 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 $p_\gamma(a|g_{sub}, q)$. β and γ denote the parameters of the retriever and the generator, respectively.

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 θ denotes all parameters in the above two modules.

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 graph neural network (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 GNN-based retriever encodes the knowledge graph to identify question-relevant facts. Given a question and a knowledge graph, it outputs fact representations that serve multiple purposes in D-RAG.

Fact Representation. For each fact τ_i in the knowledge graph, we construct a representation by concatenating its constituent elements:

$$\mathbf{F}_i = [\mathbf{h}_i \parallel \mathbf{r}_i \parallel \mathbf{t}_i] \in \mathbb{R}^{D_{\text{GNN}}}, \tag{2}$$

where $\mathbf{h}_i$, $\mathbf{r}_i$, and $\mathbf{t}_i$ are representations of the head entity, relation, and tail entity, respectively, derived from a GNN module based on ReaRev (Mavromatis and Karypis, 2022).

From these fact representations, we compute the selection probability for each fact using a linear layer followed by a sigmoid function: $p(\tau_i) = \sigma(\mathbf{W}\mathbf{F}_i + \mathbf{b})$, where $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters.

Subgraph Selection Probability Factorization. For subgraph sampling, computing the exact probability of a specific subgraph is combinatorially complex. Therefore, we employ a factorization approach that decomposes the subgraph selection probability into independent binary selections for each fact:

$$p(g_{sub}) = \prod_{\tau_i \in g_{sub}} p(\tau_i) \prod_{\tau_j \notin g_{sub}} (1 - p(\tau_j)). \tag{3}$$

Detailed derivations of this factorization and specifications of the GNN architecture are provided in Appendix A and B, respectively.

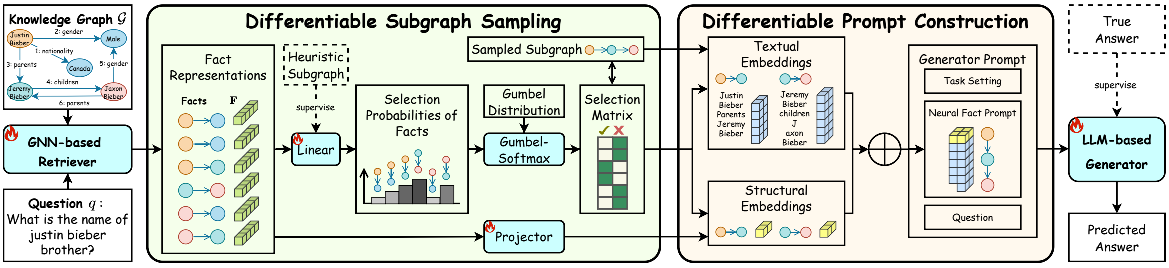


Figure 2: The proposed D-RAG consists of four steps: 1) The GNN-based retriever processes the knowledge graph to obtain fact representations; 2) The differentiable subgraph sampling assigns selection probabilities to facts and uses Gumbel-Softmax reparameterization trick to sample a subgraph; 3) The differentiable prompt construction transforms the sampled subgraph into a neural fact prompt that combines semantic and structural information; 4) The LLM-based generator predicts the final answer.

4.2 LLM-based Generator

The LLM-based generator predicts answers to questions based on the information contained in the retrieved subgraph. It processes the input through autoregressive decoding to generate answers:

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

where L_a is the token length of the ground-truth answer.

The generator receives input comprising three components as shown in Figure 2: the task setting, the question, and the neural fact prompt derived from the retrieved subgraph. These are combined in a structured template:

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

Answers are formatted as a bar-separated list: `<Ans1>|<Ans2>|...|<AnsN>`. Complete prompt examples are provided in Appendix C.

4.3 Differentiable Subgraph Sampling and Prompting

D-RAG constructs differentiable bridges across the retriever-generator interface through two designs: (1) reformulating the optimization objective into a tractable form, and (2) implementing differentiable operations for both subgraph sampling and prompt construction. This end-to-end approach enables joint optimization where the retriever learns to identify graph patterns that enhance the generator's reasoning capabilities.

4.3.1 Differentiable Formulation

The optimization objective of maximizing Equation 1 involves a summation with combinatorial complexity, making it generally intractable. We address this by optimizing its evidence lower bound (ELBO) (Hoffman et al., 2013), formulated as:

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

where $r(g_{sub})$ represents the variational distribution of the subgraph, and the inequality holds because the Kullback-Leibler divergence is non-negative. By specifying the variational distribution $r(g_{sub})$ as the retriever's distribution $p_\beta(g_{\mathrm{sub}}|q, \mathcal{G})$, the ELBO simplifies to:

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

where p_β is modeled by the GNN-based retriever and p_γ by the LLM-based generator. This formulation transforms the original combinatorial objective into a tractable expectation over subgraph distributions. To optimize this expectation through gradient-based methods, two critical challenges need to be addressed: (1) implementing differentiable operations for discrete subgraph sampling from distribution p_β, and (2) constructing differentiable prompts that allow gradients to flow through the generator p_γ.

4.3.2 Differentiable Subgraph Sampling

Sampling a subgraph results a selection matrix $\mathbf{Z} = [\mathbf{z}_1; \mathbf{z}_2; \ldots; \mathbf{z}_{N_f}] \in \{0,1\}^{N_f \times 2}$, where N_f is the total number of facts in the knowledge graph and each row $\mathbf{z}_i$ indicates whether the i-th fact is selected ($[1,0]$) or not ($[0,1]$). Given $\mathbf{Z}$, the sampled subgraph is represented as $g_{sub} = \{\tau_i | \mathbf{z}_i = [1,0]\}$.

To make this subgraph sampling process differentiable, we adopt the Gumbel-Softmax reparameterization trick (Jang et al., 2017; Maddison et al., 2017). For each fact τ_i, the retriever outputs a Bernoulli parameter $p_i = p_\beta(\tau_i)$, representing its selection probability. 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 Gumbel(0,1) noise samples and t is the temperature coefficient.

The final binary selection indicator $\mathbf{z}_i$ 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 combines discrete selection in the forward pass with differentiability in the backward pass.

With this reparameterization, our training objective becomes:

$$\mathbb{E}_{\boldsymbol{\eta} \sim p(\boldsymbol{\eta})} \left[\log p_\gamma(a | g_{sub}, q) \right], \tag{9}$$

which transforms the expectation from a complex parameterized distribution to sampling from a fixed distribution, enabling gradient flow through the discrete sampling process.

4.3.3 Differentiable Prompt Construction

After sampling the subgraph, we transform it into a neural prompt that preserves both semantic and structural information while maintaining end-to-end differentiability.

For semantic information, each fact is converted into natural language using the template `<head name>, <relation name>, <tail name>` and then tokenized and encoded into embeddings $\mathbf{V}_i \in \mathbb{R}^{L_i \times D_{\text{LLM}}}$, where L_i is the token length and D_{LLM} is the LLM embedding dimension. We multiply each embedding $\mathbf{V}_i$ by the corresponding selection indicator $\mathbf{z}_{i1}$ from matrix $\mathbf{Z}$, where z_{i1} denotes the first element of the selection vector z_i, effectively retaining only the embeddings of selected facts.

For structural information, we utilize the fact representations $\mathbf{F} = [\mathbf{F}_1; \mathbf{F}_2; \ldots; \mathbf{F}_{N_f}] \in \mathbb{R}^{N_f \times D_{\text{GNN}}}$ learned by the GNN retriever (defined in Equation 2). These representations capture each fact's position and relevance within the knowledge graph. A two-layer MLP projects these representations to align with the LLM embedding space: $\mathbf{F}' = \text{Projector}(\mathbf{F}) \in \mathbb{R}^{N_f \times D_{\text{LLM}}}$. Similarly, we select only the structural embeddings $\mathbf{F}'_i$ corresponding to facts where $\mathbf{z}_{i1} = 1$.

For each selected fact τ_i, we concatenate its semantic embedding $\mathbf{V}_i$ with its structural embedding $\mathbf{F}'_i$ to form an enriched representation. These combined embeddings are then concatenated to create the complete neural fact prompt $\mathbf{V}_\mathbf{F}$ for the LLM-based generator.

Our approach enables gradient flow from the LLM loss L back to the retriever parameters β through dual pathways:

$$\frac{\partial L}{\partial \beta} = \underbrace{\frac{\partial L}{\partial \mathbf{V}_\mathbf{F}} \frac{\partial \mathbf{V}_\mathbf{F}}{\partial \mathbf{Z}} \frac{\partial \mathbf{Z}}{\partial \beta}}_{\text{Semantic pathway}} + \underbrace{\frac{\partial L}{\partial \mathbf{V}_\mathbf{F}} \frac{\partial \mathbf{V}_\mathbf{F}}{\partial \mathbf{F}'} \frac{\partial \mathbf{F}'}{\partial \beta}}_{\text{Structural pathway}}, \tag{10}$$

where the first term represents gradient flow through the discrete selection process, and the second term captures flow through the fact representations.

For multi-hop reasoning, facts are arranged by their selection probabilities, helping preserve potential logical sequences within the sampled subgraph.

4.4 Training Strategy and Inference

With the differentiable subgraph sampling and prompting module proposed above, D-RAG supports end-to-end training. To accelerate convergence, we adopt a two-phase training strategy.

In the first phase, the GNN-based retriever is pretrained using heuristically constructed subgraphs as guidance:

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

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, the retriever and generator are trained jointly with the generation loss:

$$L_2 = -\mathbb{E}_{\boldsymbol{\eta} \sim p(\boldsymbol{\eta})} \left[\log p_\gamma(a | \mathbf{V}_\mathbf{F}, q) \right], \tag{12}$$

where $\mathbf{V}_\mathbf{F}$ is the neural fact prompt constructed from the sampled subgraph g_{sub} as described in the previous section. Importantly, $\mathbf{V_F}$ depends on both the Gumbel noise $\boldsymbol{\eta}$ and the retriever parameters β.

To balance the significantly different gradient magnitudes between the retriever pre-training and generation objectives, we apply a direct gradient normalization approach:

$$L_{\text{joint}} = \lambda \frac{L_1}{||\nabla_\beta L_1||} + (1-\lambda) \frac{L_2}{||\nabla_\beta L_2||}, \quad (13)$$

where λ is a balancing hyperparameter and $||\nabla_\beta L_i||$ represents the norm of gradients with respect to the retriever parameters.

During inference, D-RAG performs subgraph sampling through three steps: (1) computing fact selection probabilities via the GNN retriever, linear layer, and sigmoid activation without Gumbel-Softmax reparameterization, (2) ranking all facts by their probabilities, and (3) applying a hybrid selection strategy that selects the top-k facts while filtering out those below probability threshold p. The selected facts are then sorted by probability in ascending order before being fed to the generator.

For both WebQSP and CWQ datasets, $k = 100$ due to LLM context length constraints, and $p = 0.01$ as tuned on the WebQSP validation set. This hybrid approach balances comprehensive coverage with relevance quality.

5 Experiments

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, and 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 without utilizing graph structure during retrieval; and 3) Graph-LLM methods that maintain dedicated graph-based retrieval and leverage LLMs for reasoning. The details of each baseline are described in Appendix E.

Evaluation Metrics. Following previous works (Luo et al., 2024a; 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 measures whether any correct answer appears in the model's response, representing a basic retrieval capability. In contrast, F1 provides a more rigorous and comprehensive assessment by balancing precision and recall, thus better reflecting the model's overall answer quality. Further details are provided in Appendix F.

Implementations. D-RAG employs the ReaRev (Mavromatis and Karypis, 2022) model as the GNN and utilizes the Llama3-8B-Instruct (Meta, 2024) as the LLM. Based on entities linked to the knowledge graph, heuristic subgraphs are extracted via SPARQL query parsing. A heuristic subgraph is a set of facts that conform to the intrinsic logic of the SPARQL query, typically forming a tree structure. Full implementation details are provided in Appendix G.

5.2 Main Results

To evaluate the overall effectiveness of D-RAG, 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 among comparable methods. Specifically, on the WebQSP dataset, D-RAG achieves a **2.5%** improvement in Hits@1 over the best-performing baseline SubgraphRAG, and outperforms DECAF by **3.4%** in the F1 score. Although some baselines like RoG achieve competitive Hits@1 (85.7%), their F1 scores (70.8%) lag substantially behind, suggesting they may retrieve some correct answers but with lower precision.

For the more complex CWQ dataset, the proposed approach demonstrates a **1.8%** advantage in Hits@1 compared to the best-performing ToG approach, and surpasses GNN-RAG by **4.4%** 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 consistently superior performance across datasets of varying complexity, achieving the highest scores on both basic retrieval capability (Hits@1) and the more comprehensive

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., 2024a)	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	**89.1**	**80.5**	**70.3**	**63.8**

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

measure of answer quality (F1).

A recent work, GCR (Luo et al., 2024b), using the proprietary GPT4-o-mini (OpenAI, 2024a), achieves substantially lower F1 scores than our D-RAG approach: 6.4% lower on WebQSP (74.1% vs. 80.5%) and 2.1% lower on CWQ (61.7% vs. 63.8%). While GCR reports higher Hits@1 scores with GPT4-o-mini (92.2% on WebQSP and 75.8% on CWQ), its performance drops significantly when using comparable open-source models. With Qwen-2-7B (Yang et al., 2024), which has similar parameter size to D-RAG, GCR's Hits@1 on WebQSP falls to 86.3%, 2.8% below our approach. These results reveal that while proprietary models may excel at Hits@1 through internal knowledge, they still struggle with retrieval precision that impacts F1 scores. D-RAG mitigates this limitation, achieving higher F1 scores using only open-source models.

5.3 Ablation Study

To evaluate the effectiveness of end-to-end optimization between the retriever and the generator, our ablation experiments compare D-RAG with four training method 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 from the generator to the retriever; 3) *Static Cascade*, where the generator is optimized using outputs from the frozen retriever; 4) *Isolation*, where the generator is trained using heuristic subgraphs as input, completely decoupling the two modules. Further details are available in Appendix H.

Impact on Overall Performance. Table 2 presents the performance comparison across different training methods on WebQSP and CWQ. We report both "Full Dataset" performance across the entire test set and "Retrieved Subset" metrics for cases where at least one relevant fact is retrieved.

The results reveal that D-RAG consistently outperforms all variants in most metrics across both datasets, with particularly substantial F1 score improvements. On retrieval success cases, D-RAG achieves up to 6.4% and 8.5% higher F1 scores than the best variant on WebQSP and CWQ respectively, demonstrating that end-to-end optimization enables more effective utilization of retrieved facts.

Dynamic Cascade shows modest improvements over *Static Cascade*, confirming the benefit of continuously updating the retriever during training. While *REINFORCE* generally outperforms *Static Cascade*, it still falls short of D-RAG's per-

Training Method	WebQSP				CWQ			
	Full Dataset		Retrieved Subset		Full Dataset		Retrieved Subset	
	Hits@1	F1	Hits@1	F1	Hits@1	F1	Hits@1	F1
D-RAG	**89.1**	**80.5**	**94.0**	**86.2**	**70.3**	**63.8**	81.7	**75.6**
REINFORCE	85.1	72.9	90.4	78.9	61.7	55.4	73.0	66.7
D-RAG w/o e2e								
Dynamic Cascade	85.3	74.0	90.4	79.8	61.9	55.9	73.5	67.1
Static Cascade	84.8	73.0	90.7	79.4	60.6	54.3	73.2	66.6
Isolation	82.7	53.2	91.1	59.6	63.1	30.0	**85.1**	40.0

Table 2: Ablation study comparing overall performance across different training methods.

formance, suggesting direct gradient propagation is more effective than reward-based optimization. The *Isolation* variant maintains reasonable Hits@1 performance but exhibits significant drops in F1 scores due to the training-inference gap between clean training subgraphs and noisy inference retrieval.

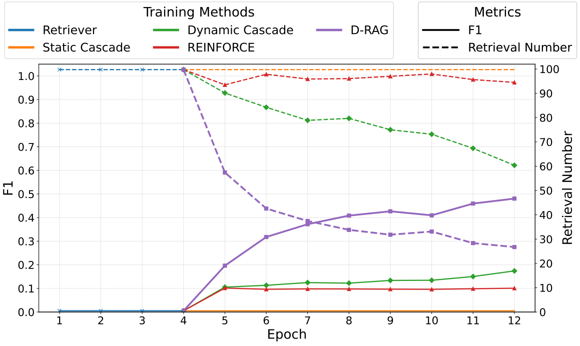


Figure 3: Evolution of retriever F1 (solid lines) and retrieval number (dashed lines) across training epochs on WebQSP. Epochs 1-4 represent retriever-only pretraining, followed by joint training with different methods. Top-100 facts with selection probability >0.01 were used for retrieval evaluation.

Impact on Retrieval Performance. Figure 3 reveals two advantages of D-RAG's retriever optimization. First, D-RAG consistently achieves the highest retriever F1 scores after joint training begins, demonstrating that gradient propagation from the generator effectively refines retrieval quality. Second, D-RAG shows a more significant downward trend in retrieved fact count compared to others, while maintaining high recall performance (95.46% on WebQSP and 95.48% on CWQ), which demonstrates superior noise reduction capability that aligns better with the actual average of 6.4 relevant facts per WebQSP question.

5.4 Relationship Between Retrieval and Generation Performance

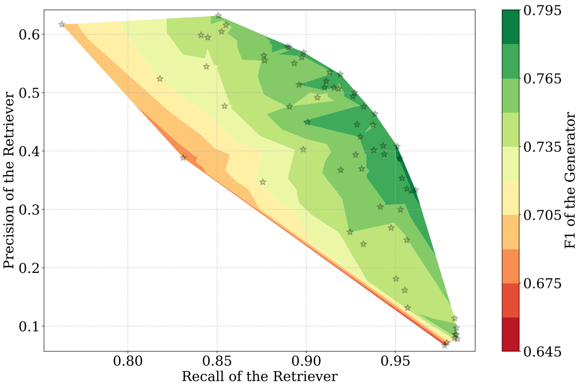


Figure 4: Impact of retriever quality on generator performance (WebQSP dataset). Heatmap shows generator F1 scores (color intensity) as a function of retriever recall (x-axis) and precision (y-axis). Each point represents a model configuration with different retrieval configurations using various probability thresholds (0-0.9) applied to the top-100 retrieved facts.

To understand how different aspects of retrieval quality affect generation performance, we examined various retrieval configurations and their impact on generator performance. As shown in Figure 4, both retrieval recall and precision significantly impact generator performance. Models in the lower-right region (high recall but low precision) perform worse than those with balanced metrics, indicating that retrieving many relevant facts without filtering irrelevant ones leads to suboptimal results. Similarly, models in the upper-left region (high precision but low recall) underperform due to insufficient fact coverage.

To further evaluate the effectiveness and efficiency of D-RAG, we perform additional experiments including overall performance comparison under different situations, evaluation on the

MetaQA dataset, detailed analysis, training efficiency analysis, and a case study in Appendix I.

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. D-RAG achieves this through reformulating the optimization and a differentiable implementation of subgraph sampling and prompt construction. Experimental results demonstrate that D-RAG outperforms state-of-the-art methods with substantial improvements, where the joint optimization significantly reduces noise in the retrieved subgraph while showing that both precision and recall in retrieval impact generator performance.

Limitations

Despite the effectiveness of D-RAG, we acknowledge several limitations of our 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.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. U24A20331), the Beijing Natural Science Foundation (No. L251005, L243015) and the Key Research and Development Program of Xinjiang Uyghur Autonomous Region (No. 2023B01005).

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.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 793–802. PMLR.

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.

Zhong Guan, Likang Wu, Hongke Zhao, Ming He, and Jianpin Fan. 2025. Attention mechanisms perspective: Exploring llm processing of graph-structured data. *Preprint*, arXiv:2505.02130.

Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data ? an empirical evaluation and benchmarking. *Preprint*, arXiv:2305.15066.

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. 2024a. 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.

Linhao Luo, Zicheng Zhao, Chen Gong, Gholamreza Haffari, and Shirui Pan. 2024b. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. *CoRR*, abs/2410.13080.

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. 2024a. Gpt-4o mini: advancing cost-efficient intelligence.

OpenAI. 2024b. 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.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *CoRR*, abs/2407.10671.

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 3, 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 3.

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, and the relation encoder refers to the LM encoder and MLP projection components used in the node initialization process.

B.2 Loss design of GNN-based Retriever

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

$$\begin{aligned}
&D_{KL}(p_{heur}(g_{sub}) | p_\beta(g_{sub})) \\
&= - \sum_{\tau \in g_{sub}} \log p_\beta(\tau) = L_{\text{BCE}}.
\end{aligned} \tag{14}$$

This can be implemented using PyTorch's BCE (Binary Cross Entropy) weighted loss [1]. Inspired by the work of (Lin et al., 2024), to address the sparsity of positive examples in knowledge graph fact classification tasks, we further incorporate a rank loss:

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

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:

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

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

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

C Prompts

Figure 5 illustrates the full input prompt received by the LLM-based generator, which consists of three components: task setting, question, and subgraph. The subgraph is shown in typewriter font, representing the neural fact prompt in D-RAG.

In the subgraph part, each line corresponds to a distinct fact that will be converted to embedding form before being input to the LLM. The `<S-Embedding>` marker at the beginning of each line represents the structural embedding mentioned in the proposed approach. The textual content following this marker contains the semantic information of each fact. Together, these elements constitute the neural fact prompt that enables the model to effectively integrate knowledge during generation.

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., 2024a; 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.

The overall statistics of both datasets are summarized in Table 3, including the number of samples in training, validation and test sets.

Table 4 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 reasoning 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 3: Statistics of the datasets.

Datasets	1-hop	2-hop	3-hop	≥4-hop
WebQSP	62.00%	37.66%	0.17%	0.17%
CWQ	24.66%	64.78%	7.50%	3.06%

Table 4: 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.
- 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 pretraining 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.

Complete Generator Prompt

Answer the question based on the provided facts.
Question: what does jamaican people speak
Provided facts:

```
<S-Embedding> Jamaica, location.country.official_language, Jamaican English
<S-Embedding> Jamaica, location.country.languages_spoken, Jamaican English
<S-Embedding> Jamaica, location.country.languages_spoken, Jamaican Creole English Language
<S-Embedding> Jamaica, location.country.currency_used, Jamaican dollar
<S-Embedding> Jamaica, location.country.form_of_government, Democracy
<S-Embedding> Jamaica, location.country.form_of_government, Parliamentary system
<S-Embedding> Jamaica, base.locations.countries.continent, North America
<S-Embedding> Jamaica, location.country.form_of_government, Constitutional monarchy
<S-Embedding> Grenada, location.country.official_language, English Language
<S-Embedding> Bermuda, location.country.official_language, English Language
<S-Embedding> Belize, location.country.official_language, English Language
<S-Embedding> Turks and Caicos Islands, location.country.official_language, English Language
<S-Embedding> Bahamas, location.country.official_language, English Language
<S-Embedding> Cayman Islands, location.country.official_language, English Language
<S-Embedding> Puerto Rico, location.country.official_language, English Language
<S-Embedding> Grenada, location.country.languages_spoken, English Language
<S-Embedding> Bermuda, location.country.languages_spoken, English Language
<S-Embedding> Costa Rica, location.country.languages_spoken, Jamaican Creole English Language
<S-Embedding> , location.country.languages_spoken, English Language
<S-Embedding> 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.

- 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., 2024a) enhances LLM reasoning by leveraging KG structure to generate faithful reasoning chains 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 (Mavromatis and Karypis, 2024; Li et al., 2025; Luo et al., 2024b) 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 answer.

It is important to note that the above discussion pertains to the overall performance of KGQA systems in terms of answer generation. Throughout this paper, we also report retrieval performance using F1, recall, and precision metrics. These retrieval metrics are calculated by comparing the facts in the retrieved subgraph with those in the heuristic subgraph, which serves as a reference standard. A key consideration is that KGQA benchmarks do not provide ground truth subgraph annotations. The heuristic subgraphs are constructed by parsing SPARQL queries associated with each question, detailed in Appendix G. This parsing ensures that the heuristic subgraphs fully align with the multi-hop reasoning required by the questions, making them relatively reliable reference standards for evaluating retrieval performance.

G Implementation Details

Preprocessing. Consistent with prior work (Mavromatis and Karypis, 2022; Luo et al., 2024a), 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). After identifying entities in the questions, we construct a heuristic subgraph for each question by parsing the SPARQL query. For each SPARQL query, we focus on the logic chain from the topic entity to the answer entity, identifying paths that connect the topic entity to the answer through specific logical chains. All facts along these paths collectively form the heuristic subgraph used in the proposed approach.

Optimization and Hyperparameters. We train separate models for CWQ and WebQSP datasets. The training process follows a two-stage approach: GNN pre-training followed by joint training. During the first training phase (retriever pre-training), we train the model for 10 epochs. In the second training phase (joint training), we train for 18 epochs.

For model optimization, we apply different strategies to the GNN and LLM components. 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 configured as: lora_r=8, lora_alpha=16, and dropout=0.05, specifically targeting the q_proj and v_proj modules. We employ the AdamW optimizer with a weight decay of 0.001, a batch size of 16, and a cosine learning rate scheduler.

Regarding the hyperparameters in our formulations, we set the Gumbel-Softmax temperature coefficient to 0.5 and the loss balancing parameter λ to 0.9. All experiments are conducted on 2 NVIDIA A800-80GB GPUs.

H Details of Ablation Study

As mentioned in Section 4.4, we initially pre-train the retriever using heuristic subgraph labels to prevent it from retrieving completely irrelevant subgraphs. All training method variants describes below, including our proposed D-RAG, are trained based on this pre-trained retriever. Here we elaborate on the four training method variants:

1. *REINFORCE*: We implements the REINFORCE algorithm with variance reduction techniques to jointly optimize both the retriever and the generator. Two reward functions are considered: (i) the negative of the generator's answer loss, and (ii) the recall of retrieved subgraphs compared to heuristic subgraphs. As we observes no significant difference between these reward formulations, the results reported in the main paper correspond to the recall reward.

2. *Dynamic Cascade*: In this approach, both modules are trained simultaneously with the generator using real-time outputs from the retriever during training. However, gradient backpropagation from the generator to the retriever is blocked, meaning the retriever is only optimized using heuristic subgraph labels.

3. *Static Cascade*: The generator is optimized using outputs from the initial fixed retriever (after pre-training). The retriever remains frozen throughout this process and is trained only with heuristic subgraph labels.

4. *Isolation*: The generator is trained using heuristic subgraphs as input, completely decoupling the two modules. Both the retriever and the generator are essentially trained independently.

Table 5 summarizes the key differences between these training methods. The key distinction between D-RAG and the *REINFORCE* variant lies in the granularity of supervision: D-RAG employs fine-grained supervision through direct end-to-end gradient-based optimization, allowing it to analyze the influence of each individual fact on answer generation, while *REINFORCE* uses policy gradient

Training Method	Retriever Supervision	Generator Input	G → R gradient
D-RAG	Retrieval label + Answer label	Real-time retriever output	✓
REINFORCE	Retrieval label + Reward label	Real-time retriever output	✓
Dynamic Cascade	Retrieval label	Real-time retriever output	×
Static Cascade	Retrieval label	Fixed pre-trained retriever output	×
Isolation	Retrieval label	Heuristic subgraph	×

Table 5: Comparison of different training methods highlighting differences in retriever supervision signals, generator inputs, and whether gradients flow from generator to retriever (G → R) during joint training.

methods that provide only coarse-grained, holistic supervision regarding the quality of the retrieved subgraph information.

I Additional Experiment Results

I.1 Performance Comparison Under Different Situations

In this section, we provide a detailed analysis of the performance results presented in Table 6, which compares our proposed D-RAG method against several baseline training methods as described in Section 5.3.

From Table 6, we can draw three important observations:

1) D-RAG's Comprehensive Effectiveness: D-RAG consistently outperforms alternative training methods across almost all metrics and complexity levels. This superiority extends to both generation metrics (Hits@1 and F1) and retrieval metrics (Recall and Precision), demonstrating the holistic effectiveness of the proposed approach.

2) Recall Necessity but Insufficiency: High recall is necessary but not sufficient for strong generation performance. or 3-hop questions, the difference in recall between D-RAG and Dynamic Cascade is 6.9 percentage points (89.8% vs. 82.9%), yet the gap in generation F1 is significantly larger at 15.3% (74.5% vs. 59.2%). This suggests that retrieval precision and effective utilization of retrieved documents also play crucial roles in generation quality.

3) Widening Retrieval-Generation Gap: As question complexity increases, the gap between retrieval performance and generation performance widens. For 4-hop questions, despite D-RAG maintaining high recall (91.2%), its generation Hits@1 drops to 58.0% - a gap of 33.2%. In comparison, for 1-hop questions, this gap is much smaller (92.6% recall vs. 74.4% Hits@1, a difference of 18.2%).

I.2 Performance on MetaQA Dataset

To further evaluate D-RAG's effectiveness across different KGQA benchmarks, we conduct experiments on MetaQA-3hop following RoG's (Luo et al., 2024a) setup with 1,000 training samples. D-RAG is directly fine-tuned on MetaQA-3hop. As shown in Table 7, D-RAG achieves substantial improvements over RoG baselines, with 8.72% and 27.82% gains in Hits@1 and F1 respectively. RoG (from scratch) uses pre-trained models directly, while RoG (transfer from Freebase) applies additional fine-tuning after Freebase pre-training. These results demonstrate D-RAG's consistent effectiveness across different KGQA datasets.

I.3 Detailed Analysis

Due to computational constraints, all experiments in this detailed analysis were conducted with joint training limited to 8 epochs, whereas the main experimental results reported in previous sections used 18 training epochs. This difference in training duration may account for some performance discrepancies between these detailed analyses and our primary results.

Robustness to Loss Balancing Hyperparameter. Figure 6 examines the effect of the loss balancing hyperparameter λ on the overall performance, where λ controls the weighting between retriever and generator losses as defined in Equation 13.

The experimental results demonstrate remarkable stability across the entire range of λ values (0.1 to 0.9). This consistent performance indicates

Training Method	Generation Hits@1				Generation F1			
	1-hop	2-hop	3-hop	4-hop	1-hop	2-hop	3-hop	4-hop
D-RAG	**74.4**	**81.3**	**79.8**	**58.0**	**69.0**	**76.2**	**74.5**	**55.6**
REINFORCE	63.7	73.4	65.9	45.7	59.6	67.4	59.4	45.7
Dynamic Cascade	66.1	74.0	65.9	39.5	62.4	67.8	59.2	39.5
Static Cascade	64.4	71.1	64.3	43.2	60.4	64.9	57.9	42.7

Training Method	Retrieval Recall				Retrieval Precision			
	1-hop	2-hop	3-hop	4-hop	1-hop	2-hop	3-hop	4-hop
D-RAG	**92.6**	**95.5**	**89.8**	**91.2**	**6.9**	**20.2**	**23.1**	13.9
REINFORCE	89.1	90.7	81.1	80.7	4.0	9.0	13.0	14.1
Dynamic Cascade	89.4	91.6	82.9	82.5	4.3	10.9	14.7	**15.1**
Static Cascade	86.7	85.5	78.1	78.7	3.9	9.2	13.6	14.1

Table 6: Performance comparison of D-RAG against different training methods on the CWQ dataset. Results show both overall performance (Hits@1 and F1) and retrieval performance (Recall and Precision) across different complexity levels (1-4 hops).

Method	**Hits@1**	**F1**
RoG (from scratch)	84.81	41.32
RoG (transfer from Freebase)	88.98	50.68
D-RAG	**97.70**	**78.50**

Table 7: Performance comparison on MetaQA-3hop dataset.

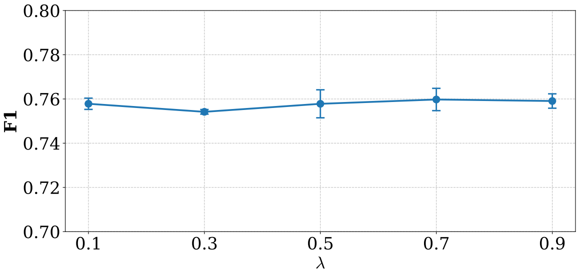


Figure 6: Impact of loss balancing hyperparameter λ on overall performance for WebQSP Dataset. The plot shows the overall F1 scores (y-axis) achieved with different values of λ (x-axis) in the joint loss function 13. Error bars represent standard deviations across three experimental runs.

that the system is largely insensitive to the specific weighting between retriever and generator components. This robustness can be primarily attributed to the gradient normalization mechanism employed in our loss formulation, which effectively prevents either component from dominating the optimization process regardless of the λ value. Future work could explore more sophisticated gradient balancing techniques such as GradNorm (Chen et al., 2018), which builds upon gradient normalization by introducing adaptive weighting strategies that automatically adjust task weights during training based on learning dynamics.

Fact Order	**Training Method**	
	D-RAG	Dynamic Cascade
ascent	76.11 ± 0.09	75.04 ± 0.38
descent	76.33 ± 0.22	74.94 ± 0.10
random	76.66 ± 0.34	73.77 ± 0.77

Table 8: The F1 scores across different fact ordering strategies and training methods. The table compares the performance (F1 score ± standard deviation) of D-RAG and Dynamic Cascade training methods under three fact ordering strategies.

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

Table 8 evaluates the influence of fact ordering on overall performance for both D-RAG and *Dynamic Cascade* in Section 5.3. The results reveal two key findings. First, D-RAG demonstrates remarkable robustness across all ordering strategies. This stability suggests that D-RAG effectively learns to process fact sequences regardless of their

Training Method	Time (minutes)
D-RAG	74.43 ± 0.43
Dynamic Cascade	68.92 ± 1.28
Static Cascade	69.66 ± 0.42

Table 9: Training time per epoch on CWQ with 5,000 random samples. Time variations (±) indicate the standard deviation across multiple epochs.

presentation order, an advantageous property for real-world applications where optimal fact ordering may not be predetermined or existed.

In contrast, the Dynamic Cascade method shows greater sensitivity to fact ordering, with performance declining noticeably under random ordering (73.77%) compared to more structured approaches (ascent: 75.04%, descent: 74.94%). This indicates that consistent, deterministic ordering strategies generally outperform random fact arrangements.

I.4 Training Efficiency Analysis

Table 9 presents the training time per epoch for different training methods on the CWQ dataset using 5,000 random samples.

D-RAG shows a modest increase in training time compared to others, requiring 8.0% more time than *Dynamic Cascade* and 6.9% more than *Static Cascade*. This additional overhead primarily comes from computing gradient norms during loss calculation and the extra backpropagation computations required for end-to-end training.

Interestingly, *Dynamic Cascade* achieves faster training times than *Static Cascade* despite the additional computation needed for retriever updates. As shown in Figure 3, *Dynamic Cascade* retrieves fewer facts than *Static Cascade*, suggesting that the computational benefits from retrieving fewer facts outweigh the cost of training the retriever.

Similarly, the relatively small time difference between D-RAG and the cascade variants can be largely attributed to D-RAG's ability to retrieve fewer and more relevant facts as training progresses (as shown in Figure 3).

I.5 Case Study

To illustrate the advantages of D-RAG, we select representative examples from both WebQSP and CWQ datasets, covering a range of reasoning complexity (1-hop to 3-hop) and answer cardinality (single and multiple answers). Table 10 and 11 present these case studies with visualizations of the retrieved subgraphs and generated answers across different training methods.

The retrieved subgraphs in different cases reveal D-RAG's superior retrieval characteristics compared to other methods. D-RAG consistently produces more focused subgraphs with significantly reduced noise and maintains high recall of relevant facts. This selective retrieval aligns with our goal of providing LLMs with concise yet comprehensive information, as excessive irrelevant facts can distract the generation process and insufficient coverage may miss critical reasoning chains. In the WebQSP example "who inspired Obama" (Table 10, 1-hop), we observe that *Static Cascade* retrieves a sparse and incomplete subgraph leading to an incorrect answer, D-RAG successfully identifies and preserves all three correct answers with minimal extraneous facts.

The 3-hop example from CWQ—"What county is the city that includes the Houston City Council as a part of their government located in?"—particularly highlights D-RAG's effectiveness in complex reasoning scenarios. This question requires following a challenging reasoning chain: `Houston City Council <- governmental body <- governing officials -> county -> Montgomery County`. *Static Cascade* fails to retrieve the complete reasoning chain, resulting in an incorrect answer ("Texas"), and *Dynamic Cascade* suffers from excessive noise that impedes identifying the correct reasoning chain. In contrast, D-RAG effectively prunes irrelevant facts and preserving the critical reasoning chains, enabling the generator to correctly identify "Montgomery County". These cases empirically validate the proposed approach, demonstrating how end-to-end optimization produces cleaner, more focused subgraphs that contain essential reasoning chains and minimize noise. Based on these high-quality subgraphs, the KGQA system generates accurate answers, underscoring the practical benefits of D-RAG in real-world knowledge-based question answering scenarios.

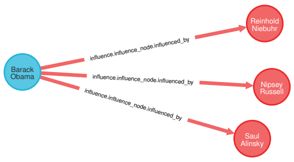
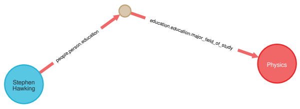
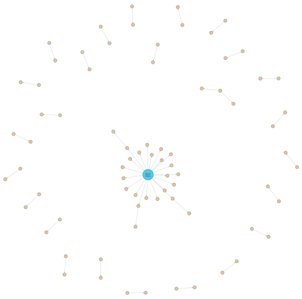
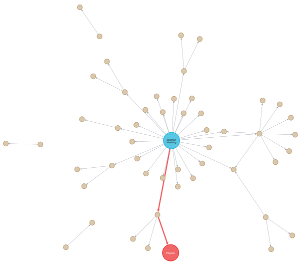
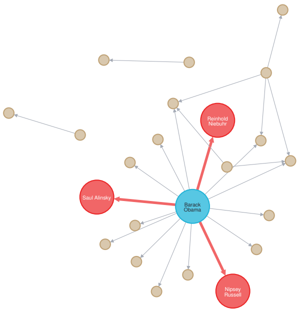
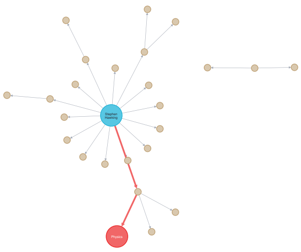
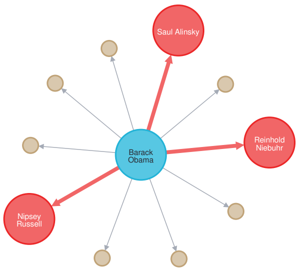
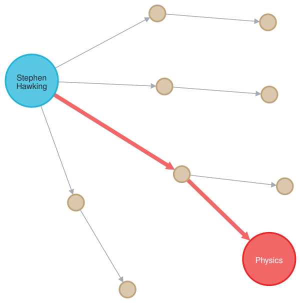
	1-hop Example	2-hop Example
Question	who inspired obama	what did stephen hawking study
True Answers	Reinhold Niebuhr | Nipsey Russell | Saul Alinsky	Physics
Heuristic Subgraph from SPARQL	 Barack Obama influence.influence_node.influenced_by influence.influence_node.influenced_by influence.influence_node.influenced_by Reinhold Niebuhr Nipsey Russell Saul Alinsky	 Stephen Hawking people.person.education education.education.major_field_of_study Physics
Static Cascade (Subgraph)		
Static Cascade (Answers)	Martin Luther King Jr.	Astrophysicist | Cosmologist| Physicist...
Dynamic Cascade (Subgraph)	 Barack Obama Reinhold Niebuhr Saul Alinsky Nipsey Russell	
Dynamic Cascade (Answers)	Saul Alinsky	Cosmologist | Physicist | Astrophysicist...
D-RAG (Subgraph)	 Saul Alinsky Barack Obama Reinhold Niebuhr Nipsey Russell	
D-RAG (Answers)	Reinhold Niebuhr | Nipsey Russell | Saul Alinsky	Physics

Table 10: Case studies on WebQSP dataset. Comparison of retrieved subgraphs and generated answers across different methods on 1-hop and 2-hop questions. Blue nodes represent question entities, red nodes represent answer entities, and yellow nodes are intermediate entities.

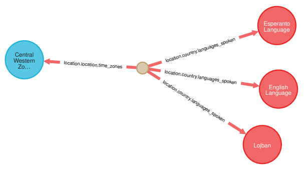
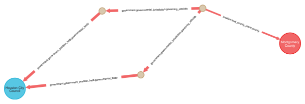
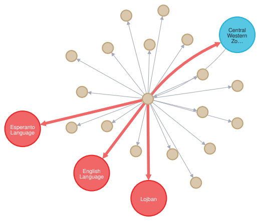
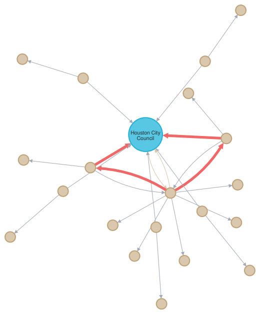
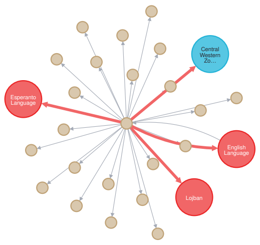
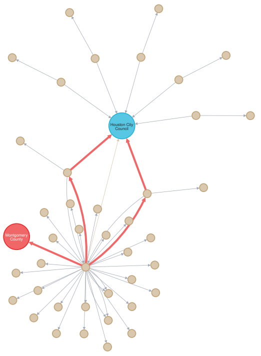
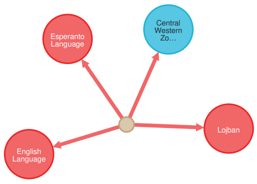
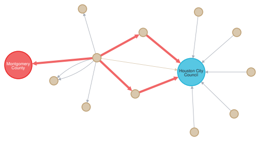
	2-hop Example	3-hop Example
Question	What languiages are spoken by residents of the Central Western Time Zone?	What county is the city that includes the Houston City Council as a part of their government located in?
True Answers	Esperanto Language | Lojban | English Language	Montgomery County
Heuristic Subgraph from SPARQL	 Central Western Zo... location.location.time_zones location.country.languages_spoken location.country.languages_spoken location.country.languages_spoken Esperanto Language English Language Lojban	 government.governmental_jurisdiction.governing_officials government.government_position_held.governmental_body government.governmental_jurisdiction.governing_officials location.hud_county_place.county government.government_position_held.governmental_body Houston City Council Montgomery County
Static Cascade (Subgraph)	 Central Western Zo... Esperanto Language English Language Lojban	
Static Cascade (Answers)	English Language	Texas
Dynamic Cascade (Subgraph)	 Central Western Zo... Esperanto Language English Language Lojban	
Dynamic Cascade (Answers)	English Language	Harris County
D-RAG (Subgraph)	 Esperanto Language Central Western Zo... Lojban English Language	
D-RAG (Answers)	Esperanto Language | Lojban | English Language	Montgomery County

Table 11: Case studies on CWQ dataset. Comparison of retrieved subgraphs and generated answers across different methods on 2-hop and 3-hop questions. Blue nodes represent question entities, red nodes represent answer entities, and yellow nodes are intermediate entities.