SimGRAG: Leveraging Similar Subgraphs for Knowledge Graphs Driven Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Recent advancements in large language models (LLMs) have shown impressive versatility across various tasks. To eliminate its hallucinations, retrieval-augmented generation (RAG) has emerged as a powerful approach, leveraging external knowledge sources like knowledge graphs (KGs). In this paper, we study the task of KG-driven RAG and propose a novel *Similar Graph Enhanced Retrieval-Augmented Generation* (SimGRAG) method. It effectively addresses the challenge of aligning query texts and KG structures through a two-stage process: (1) query-to-pattern, which uses an LLM to transform queries into a desired graph pattern, and (2) pattern-to-subgraph, which quantifies the alignment between the pattern and candidate subgraphs using a graph semantic distance (GSD) metric. We also develop an optimized retrieval algorithm that efficiently identifies the top-k subgraphs within 1-second latency on a 10-million-scale KG. Extensive experiments show that SimGRAG outperforms state-of-the-art KG-driven RAG methods in both question answering and fact verification.

1 Introduction

Pre-trained large language models (LLMs) are popular for diverse applications due to their generality and flexibility (Zhao et al., 2023; Minaee et al., 2024; Wang et al., 2024a). To avoid the hallucinations or outdated knowledge of LLMs (Zhang et al., 2023; Baek et al., 2023), Retrieval-Augmented Generation (RAG) (Zhao et al., 2024; Gao et al., 2023) integrates LLMs with external knowledge sources to produce grounded outputs, where knowledge graphs (KGs) (Ji et al., 2022) have emerged as a valuable option (Peng et al., 2024).

For many KG-driven tasks, their KG schemas align with human cognition and can be read by humans. In other words, a non-specialist can describe the knowledge using an intuitive graph structure. In this paper, we follow existing KG-driven RAG methods (Baek et al., 2023; Kim et al., 2023a; Liu et al., 2024) and focus on such human-understandable KGs to enable the mimicking of human reasoning. As shown in Figure 1, an ideal approach should address the following features.

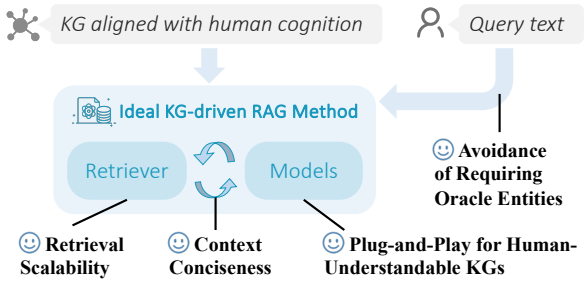


Figure 1: Ideal features for KG-driven RAG methods.

Plug-and-Play on Human-Understandable KGs. To fully leverage the inherent generalization power of LLMs, an ideal approach should be easily deployable without additional training or fine-tuning for KGs that align with human cognition and can be interpreted by LLMs. Otherwise, training a smaller and task-specific model on such KGs would be a more cost-effective alternative.

Avoidance of Requiring Oracle Entities. In real applications, users might not always know the precise entity IDs related to their queries. Thus, it would be better if a method naturally does not require users to specify the oracle entities.

Context Conciseness. The retrieved subgraphs should focus on the most relevant and essential nodes and edges, ensuring clear contexts for LLMs.

Retrieval Scalability. An ideal algorithm should scale to large KGs with tens of millions of nodes and edges while maintaining acceptable latency.

Existing approaches typically follow a paradigm of retrieving subgraphs from the KG and feeding them into LLMs to generate the final response. The critical challenge lies in effectively aligning query texts with the structural knowledge encoded in KGs. Figure 2 summarizes different mechanisms of ex-

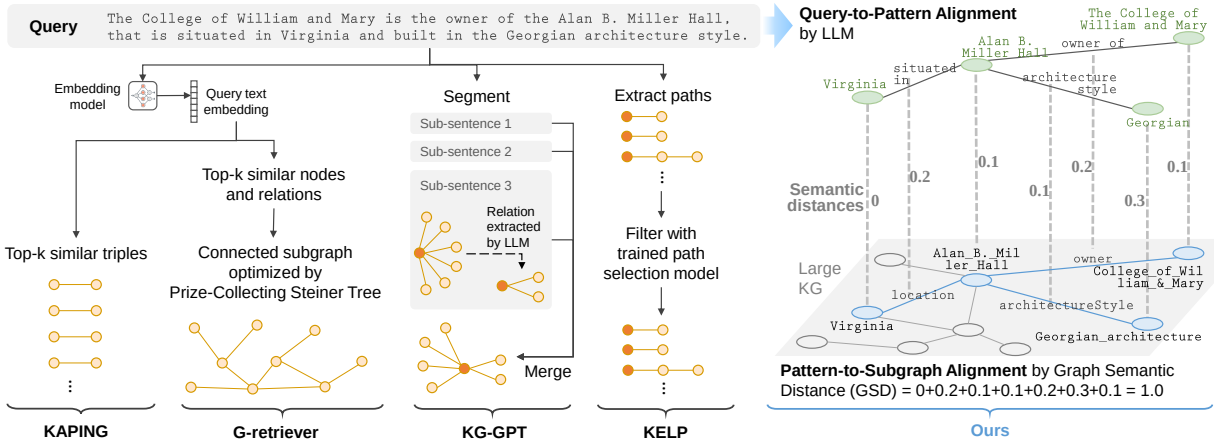


Figure 2: Comparison of mechanisms for aligning query text with KG structures. The example task is fact verification, where the query comes from FactKG dataset (Kim et al., 2023b) with DBpedia (Lehmann et al., 2015).

isting approaches. Specifically, *(i)* KAPING (Baek et al., 2023) employs query text to directly retrieve isolated triples using their semantic embedding similarity, which struggles with multi-hop queries as the query embedding captures excessive information. *(ii)* G-retriever (He et al., 2024) uses query text embeddings to retrieve similar entities and relations, then extracts a connected components in KG, which potentially cannot guarantee the best *conciseness* of the retrieved subgraphs. *(iii)* KG-GPT (Kim et al., 2023a) segments the query into sub-sentences but depends on the LLM to decide relations in KG that can match each sub-sentence, compromising *scalability* as the number of candidate relations increases. *(iv)* KELP (Liu et al., 2024) trains a path selection model to identify paths that align with the query text, lacking the *plug-and-play* usability even on human-understandable KGs.

In this paper, we introduce a novel approach for aligning query text with KG structures. We first utilize an LLM to generate a pattern graph that aligns with the query text. Then, to retrieve the best subgraphs from KG that semantically align with the pattern graph, we introduce a novel metric termed *Graph Semantic Distance* (GSD). It quantifies the alignment by summing the semantic distances between corresponding nodes and relations in the pattern graph and the candidate isomorphic subgraphs. For example, in Figure 2, the LLM generates a star-shaped pattern graph aligning with the query. And the highlighted subgraph with the smallest GSD is considered as the best-aligned subgraph in KG.

Different from KG-GPT (Kim et al., 2023a) that leverages LLMs to filter relations within large KG, we only ask LLMs to generate a small pattern graph. Also, our method targets subgraphs structurally and semantically aligned with the pattern, fundamentally differing from KAPING (Baek et al., 2023) and G-retriever (He et al., 2024) that do not explicitly constrain subgraph structure or size. Our method can support more complex pattern graph structures, diverging from KELP (Liu et al., 2024) that trains a path selection model limited to 1-hop or 2-hop paths. Moreover, to retrieve the top-k similar subgraphs w.r.t. the pattern graph with the smallest GSD, we further develop an optimized algorithm with an average retrieval time of less than one second per query on a 10-million-scale KG.

Figure 3 presents the overview of the proposed *Similar Graph Enhanced Retrieval-Augmented Generation* (SimGRAG) method.

Our contributions are summarised as follows.

- We propose the query-to-pattern and pattern-to-subgraph alignment paradigm, ensuring the plug-and-play usability on human-understandable KGs and the context conciseness for LLMs.
- We define the graph semantic distance and develop an optimized subgraph retrieval algorithm to avoid requiring oracle entities and ensure retrieval scalability on million-scale KGs.
- Extensive experiments across different KG-driven RAG tasks confirm that SimGRAG outperforms state-of-the-art baselines.

2 Related Work

Knowledge Graph Meets Large Language Models. Recently, the pre-trained large language models have shown the ability to understand and handle knowledge graph (KG) related tasks (Pan et al., 2023; Jin et al., 2024; Pan et al., 2024; Yang et al., 2024; Li et al., 2024b), such as KG construction (Zhu et al., 2024b), KG completion (Xie et al., 2022; Li et al., 2024a), KG embedding (Zhang

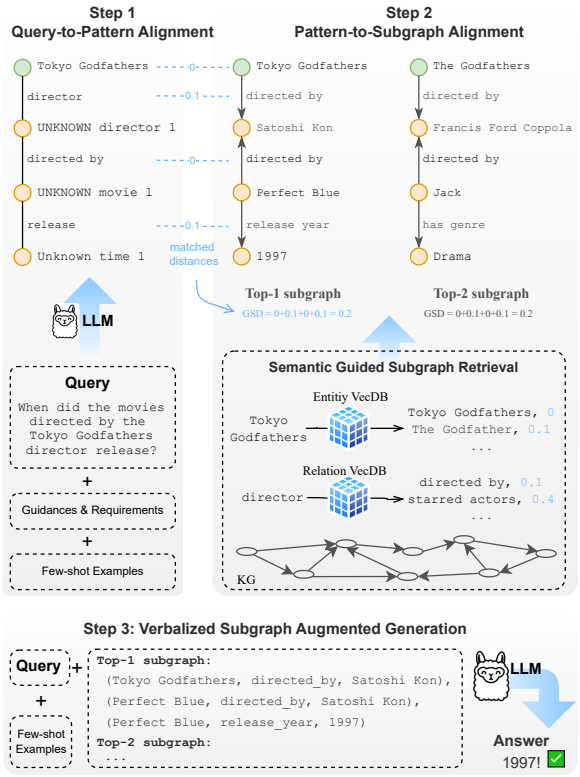


Figure 3: Overview of the SimGRAG method.

et al., 2020), and so on. Furthermore, existing studies (Zhu et al., 2024a; Mao et al., 2024; Fan et al., 2024; Wang et al., 2024b) have tried to integrate LLMs with Graph Neural Networks (GNNs) to enhance modeling capabilities for graph data.

Retrieval-Augmented Generation. In practice, LLMs may produce unsatisfactory outputs due to their hallucination or inner outdated knowledge (Baek et al., 2023). Retrieval-Augmented Generation (RAG) (Gao et al., 2023; Zhao et al., 2024) is a promising solution that retrieves related information from external databases to assist LLMs. Driven by documents, naive RAG approaches divide them into text chunks, which are embedded into dense vectors for retrieval. There are a bunch of studies and strategies optimizing each step of the RAG process (Zhao et al., 2024), including chunk division (Gao et al., 2023), chunk embedding (Li and Li, 2023; Chen et al., 2023), query rewriting (Ma et al., 2023), document reranking (Gao et al., 2023), and LLM fine-tuning (Cheng et al., 2023).

Knowledge Graph Driven Retrieval-Augmented Generation. The intricate structures of knowledge graphs (KGs) present significant challenges to traditional RAG pipelines, prompting the development of various techniques for graph-based indexing, retrieval, and generation (Peng et al., 2024). As depicted in Figure 2, KAPING (Baek et al., 2023) retrieves KG triples most relevant to the query directly. KG-GPT (Kim et al., 2023a) segments the query and presents LLMs with all candidate relations in the KG for decision-making. KELP (Liu et al., 2024) trains a model to encode paths in the KG for selecting relevant paths, although it struggles to scale to structures more complex than 2-hop paths. G-Retriever (He et al., 2024) first retrieves similar entities and relations, then constructs a connected subgraph optimized via the prize-collecting Steiner tree algorithm, and employs a GNN to encode the subgraph for prompt tuning with the LLM.

3 Preliminaries

A knowledge graph (KG) $\mathcal{G}$ is defined as a set of triples, i.e., $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{V}, r \in \mathcal{R}\}$, where $\mathcal{V}$ represents the set of entity nodes and $\mathcal{R}$ denotes the set of relations.

Given a knowledge graph $\mathcal{G}$ and a user query $\mathcal{Q}$, the task of *Knowledge Graph Driven Retrieval-Augmented Generation* is to generate an answer $\mathcal{A}$ by leveraging both large language models and the retrieved evidence from $\mathcal{G}$. This task is general and encompasses a variety of applications, including but not limited to Knowledge Graph Question Answering (KGQA) and Fact Verification (Kim et al., 2023a; Liu et al., 2024).

An embedding model (EM) transforms a textual input x to an n-dimensional embedding vector z that captures its semantic meaning, i.e., $z = \text{EM}(x) \in \mathbb{R}^n$. And the L2 distance between two vectors z_1 and z_2 is denoted by $\|z_1 - z_2\|_2 \in \mathbb{R}$.

4 The SimGRAG Approach

Effectively aligning query text with the KG structures is a critical challenge. In this section, we introduce a novel strategy that decomposes this alignment task into two distinct phases: query-to-pattern alignment and pattern-to-graph alignment.

4.1 Query-to-Pattern Alignment

Given a query text $\mathcal{Q}$, we prompt the LLM to generate a pattern graph $\mathcal{P}$ consisting of a set of triples $\{(h_1, r_1, t_1), (h_2, r_2, t_2), \ldots\}$ that align with the query semantics. We expect the LLM to interpret the user query thoughtfully, but we do not expect it to produce the exact same entities or relations appeared in the KG.

To guide the LLM in generating the desired patterns, our prompt first asks for the segmented phrases for each triple before generating all the triples. As shown in Table 18, it also includes a

few explicit requirements. To facilitate in-context few-shot learning (Agarwal et al., 2024), we further manually construct a few examples (typically 12-shots) based on the characteristics of each KG, guiding the LLM to generate desired patterns.

Such query-to-pattern alignment leverages the inherent understanding and instruction-following capabilities of LLMs. Based on our experiments detailed in Section 6, the accuracy of the alignment can be defined as the proportion of queries that conform to the expected pattern under manual verification. For queries involving up to 3 hops in the MetaQA (Zhang et al., 2018) and FactKG (Kim et al., 2023b) datasets, Llama 3 70B (Dubey et al., 2024) achieves the accuracies of 98% and 93%, respectively. Thus, on KGs following human cognition which can be understood by humans, such alignment could be effectively performed by the LLM without the need for additional training, ensuring plug-and-play usability. But for certain KGs with specialized structures, it may be inevitable to further fine-tune the LLMs for mimicing domain-specific specialists, as discussed in Section 8.

4.2 Pattern-to-Subgraph Alignment

Given the generated pattern graph $\mathcal{P}$, our objective is to assess the overall similarity between $\mathcal{P}$ and a subgraph $\mathcal{S}$ in the knowledge graph $\mathcal{G}$. Since the pattern $\mathcal{P}$ defines the expected structure of a subgraph, we leverage graph isomorphism to enforce structural constraints on the desired subgraph.

Definition 1 (Graph Isomorphism) *The pattern graph $\mathcal{P}$ has a node set $V_{\mathcal{P}}$, while the subgraph $\mathcal{S}$ has a node set $V_{\mathcal{S}}$. We say that $\mathcal{P}$ and $\mathcal{S}$ are isomorphic if there exists a bijective mapping $f : V_{\mathcal{P}} \rightarrow V_{\mathcal{S}}$ s.t. an edge $\langle u, v\rangle$ exists in $\mathcal{P}$ iff the edge $\langle f(u), f(v)\rangle$ exists in $\mathcal{S}$.*

Figure 2 presents an isomorphism example. Note that when checking graph isomorphism, we do not consider the edge direction, as different KGs may vary for the same relations. For instance, some KGs may express a relation such as "person A directs movie B", while others may use the reversed direction, "movie B is directed by person A".

After aligning the subgraph structure through graph isomorphism, we proceed to consider the semantic information of the nodes and relations. Similar to traditional text-driven RAG pipelines, for each entity node v and relation r in both the pattern graph $\mathcal{P}$ and the subgraph $\mathcal{S}$, we obtain the corresponding embedding vectors z as follows:

$$z_v = \text{EM}(v), \quad z_r = \text{EM}(r) \tag{1}$$

Keyword: Georgian

Rank	Entity in KG
1	Georgian
2	Atlanta_Georgian
…	…
112	Georgian_architecture

Keyword: architectural style

Rank	Relation in KG
1	architecture
2	buildingType
3	architecturalStyle
4	architect

Figure 4: Semantic L2 distance rankings of a given keyword with entities (relations) in DBpedia (Lehmann et al., 2015), computed using the embeddings generated by the Nomic model (Nussbaum et al., 2024).

For a subgraph $\mathcal{S}$ isomorphic to $\mathcal{P}$, the nodes and edges in $\mathcal{S}$ have a one-to-one mapping with those in $\mathcal{P}$. By computing the L2 distance between their embeddings, we use the pairwise matching distance (Blumenthal, 1953) to derive the following overall graph semantic distance.

Definition 2 (Graph Semantic Distance, GSD) *Given the isomorphic mapping $f : V_{\mathcal{P}} \rightarrow V_{\mathcal{S}}$ between the pattern graph $\mathcal{P}$ and the KG subgraph $\mathcal{S}$, Graph Semantic Distance (GSD) is defined as follows, where $r_{\langle u,v\rangle}$ denotes the relation of the edge $\langle u, v\rangle$.*

$$GSD(\mathcal{P}, \mathcal{S}) = \sum_{\text{node } v \in \mathcal{P}} \|z_v - z_{f(v)}\|_2 + \sum_{\text{edge } \langle u,v\rangle \in \mathcal{P}} \left\| z_{r_{\langle u,v\rangle}} - z_{r_{\langle f(u),f(v)\rangle}} \right\|_2, \tag{2}$$

Example 1 *As illustrated in Figure 2, the highlighted subgraph in KG is isomorphic to the pattern graph. By computing the text similarity (i.e., embedding distance) between the matched nodes and edges, the resulting GSD is 1.0.*

Focusing exclusively on isomorphic subgraphs guarantees conciseness. Section 5 will provides algorithms to efficiently retrieve the top-k isomorphic subgraphs with the smallest GSD in KG.

Furthermore, the joint use of graph isomorphism and semantic similarity effectively reduces noise. In practice, KGs are often noisy, and even semantically similar entities or relations may not always constitute suitable evidence. Figure 4 presents the distance rankings over the 10-million-scale DBpedia for the pattern graph in Figure 2. There are numerous entities related to "Georgian", but only the entity ranked 112 contributes to the final subgraph. Similarly, for the relation "architecture style", only the relation ranked 3 is useful. The proposed GSD metric can effectively incorporate somewhat distant entities or relations that still contribute valuable evidence to the overall subgraph, thereby eliminating the need for oracle entities.

4.3 Generalization to Unknown Entities or Relations

In practice, some queries like "Who is the director of the movie Her?" may involve unknown entities. To address this, we extend the query-to-pattern alignment process by allowing the LLM to represent unknown entities or relations with unique identifiers such as "UNKNOWN director 1", as illustrated by the pattern graph $\mathcal{P}$ in Figure 3.

In such cases, we further generalize the Graph Semantic Distance (GSD). Specifically, since the unknown entities or relations are ambiguous and difficult to match with corresponding entities or relations in the KG, we exclude them from the GSD computation. Given the isomorphic mapping $f : V_{\mathcal{P}} \rightarrow V_{\mathcal{S}}$ between the pattern graph $\mathcal{P}$ and the KG subgraph $\mathcal{S}$, we generalize GSD to:

$$GSD(\mathcal{P}, \mathcal{S}) = \sum_{\substack{\text{node } v \in \mathcal{P} \\ \text{s.t.} v \text{ is known}}} \|z_v - z_{f(v)}\|_2 + \sum_{\substack{\text{edge } \langle u,v \rangle \in \mathcal{P} \\ r_{\langle u,v \rangle} \text{ is known}}} \|z_{r_{\langle u,v \rangle}} - z_{r_{\langle f(u),f(v) \rangle}}\|_2 \quad (3)$$

Example 2 *As illustrated in Figure 3, the top-1 subgraph from the KG yields a GSD of* 0.2.

4.4 Verbalized Subgraph-Augmented Generation

Given the top-k subgraphs with the smallest Graph Semantic Distance (GSD) from the KG, we now expect the LLM to generate answers to the original query based on these evidences. To achieve this, we append each retrieved subgraph $\mathcal{S}$ to the query text in the prompt. Each subgraph is verbalized as a set of triples $\{(h_1, r_1, t_1), (h_2, r_2, t_2), \dots\}$, as illustrated in Figure 3. Additionally, to facilitate in-context learning, we also manually curate a few example queries (typically 12-shots) with their corresponding subgraphs and expected answers in the prompt. Please refer to Appendix B for details.

5 Semantic Guided Subgraph Retrieval

Performing a brute-force search over all candidate subgraphs and computing the Graph Semantic Distance (GSD) for each one is computationally prohibitive. To address this, we propose a practical retrieval algorithm in Section 5.1, which is further optimized for efficiency in Section 5.2.

5.1 Top-k Retrieval Algorithm

Recent subgraph isomorphism algorithms often follow a *filtering-ordering-enumerating* paradigm (Lee et al., 2012; Sun and Luo, 2020; Zhang et al., 2024). To narrow down the potential search space,

Algorithm 1: Top-k Retrieval Algorithm

Input: Pattern graph $\mathcal{P}$, knowledge graph $\mathcal{G}$, node candidates $C^{(n)}$, relation candidates $C^{(r)}$, and the parameter k.
Output: The top-k subgraphs from $\mathcal{G}$ with the smallest GSD.

1 Select start node $v^*_{\mathcal{P}}$ in $\mathcal{P}$ with the fewest candidates;
2 $L \leftarrow$ all triples of $\mathcal{P}$ in DFS traversal order from $v^*_{\mathcal{P}}$;
3 $res \leftarrow$ a priority queue maintaining the top-k subgraphs with the smallest GSD;
4 **foreach** $v_{\mathcal{G}} \in C^{(n)}[v^*_{\mathcal{P}}]$ **do**
5 ⌊ Expand(1, $\{v^*_{\mathcal{P}} : v_{\mathcal{G}}\}$);
6 **return** res;
7 **Function** Expand(i, f):
8 ⌊ **if** *f is a valid isomorphism mapping for $\mathcal{P}$* **then**
9 ⌊ Push the mapped subgraph $\mathcal{S}$ to res;
10 ⌊ **return**;
11 ⌊ $(h_{\mathcal{P}}, r_{\mathcal{P}}, t_{\mathcal{P}}) \leftarrow$ the ith triple in L ;
12 ⌊ $h_{\mathcal{G}} \leftarrow f(h_{\mathcal{P}})$;
13 ⌊ **foreach** (r_G, t_G) *s.t.* $(h_G, r_G, t_G) \in \mathcal{G}$ **do**
14 ⌊ **if** $r_{\mathcal{G}} \in C^{(r)}[r_{\mathcal{P}}] \wedge t_{\mathcal{G}} \in C^{(n)}[t_{\mathcal{P}}]$ **then**
15 ⌊ **if** *no contradiction for $t_{\mathcal{P}}$ in f* **then**
16 ⌊ Expand($i + 1, f \cup \{t_{\mathcal{P}} : t_{\mathcal{G}}\}$);

we first apply semantic embeddings to filter out unlikely candidate nodes and relations. For each node $v_{\mathcal{P}}$ in the pattern graph $\mathcal{P}$, we retrieve the top-$k^{(n)}$ most similar entities from the knowledge graph $\mathcal{G}$, forming a candidate node set $C^{(n)}[v_{\mathcal{P}}]$. Similarly, for each relation $r_{\mathcal{P}}$, we extract the top-$k^{(r)}$ similar relations to form the candidate relation set $C^{(r)}[r_{\mathcal{P}}]$. Figure 3 illustrates an example of the candidate nodes and relations for the pattern graph node "Tokyo Godfathers" and the relation "director". For unknown nodes or relations, as discussed in Section 4.3, we treat all nodes or relations in $\mathcal{G}$ as candidates with a semantic distance of 0.

The retrieval process is described in Algorithm 1. Initially, lines 1-2 organize all edges in $\mathcal{P}$ according to a DFS traversal order. For each candidate node $v_{\mathcal{G}}$ in the set $C^{(n)}[v^*_{\mathcal{P}}]$, we start an isomorphic mapping in lines 4-5 and iteratively expand the mapping using the Expand function until a valid mapping is found. In function Expand, when matching the ith triple $(h_{\mathcal{P}}, r_{\mathcal{P}}, t_{\mathcal{P}})$ in the ordered triple list L, the node $h_{\mathcal{P}}$ is mapped to the corresponding node $h_{\mathcal{G}}$ in $\mathcal{G}$ via the partial mapping f. Then, lines 13-16 check each neighboring relation $r_{\mathcal{G}}$ and node $t_{\mathcal{G}}$ for $h_{\mathcal{G}}$ to see if they are valid candidates and do not contradict the existing mapping f.

5.2 Optimized Retrieval Algorithm

Despite the filtering approach, the above algorithm still suffers from a large search space, especially

when there are too many candidate nodes and relations. As we only need the top-k subgraphs with the smallest GSD, we propose an optimized strategy that can prune unnecessary search branches.

Assume that during the expansion of the ith edge in L, the partial mapping from $\mathcal{P}$ to the knowledge graph $\mathcal{G}$ is represented by f. Suppose there exists an isomorphic mapping f' that can be completed by future expansion, resulting in a subgraph $\mathcal{S}$ with $GSD(\mathcal{P}, \mathcal{S})$. It can be decomposed into four terms, where $L[1:i]$ denotes the first $i-1$ triples in L and $L[i:]$ denotes the remaining triples.

$$GSD(\mathcal{P}, \mathcal{S}) = \Delta^{(n)}_{\text{mapped}} + \Delta^{(n)}_{\text{remain}} + \Delta^{(r)}_{\text{mapped}} + \Delta^{(r)}_{\text{remain}}, \quad (4)$$

$$\Delta^{(n)}_{\text{mapped}} = \sum_{\substack{\text{node } v_{\mathcal{P}} \in \mathcal{P} \\ \text{mapped in } f}} \|z_{v_{\mathcal{P}}} - z_{f(v_{\mathcal{P}})}\|_2, \quad (5)$$

$$\Delta^{(n)}_{\text{remain}} = \sum_{\substack{\text{node } v_{\mathcal{P}} \in \mathcal{P} \\ \text{not mapped in } f}} \|z_{v_{\mathcal{P}}} - z_{f'(v_{\mathcal{P}})}\|_2, \quad (6)$$

$$\Delta^{(r)}_{\text{mapped}} = \sum_{(h_{\mathcal{P}}, r_{\mathcal{P}}, t_{\mathcal{P}}) \in L[1:i]} \|z_{r_{\mathcal{P}}} - z_{r_{\langle f(h_{\mathcal{P}}), f(t_{\mathcal{P}}) \rangle}}\|_2, \quad (7)$$

$$\Delta^{(r)}_{\text{remain}} = \sum_{(h_{\mathcal{P}}, r_{\mathcal{P}}, t_{\mathcal{P}}) \in L[i:]} \|z_{r_{\mathcal{P}}} - z_{r_{\langle f'(h_{\mathcal{P}}), f'(t_{\mathcal{P}}) \rangle}}\|_2. \quad (8)$$

For Equations (6) and (8), notice that

$$\Delta^{(n)}_{\text{remain}} \geq \sum_{\substack{\text{node } v_{\mathcal{P}} \in \mathcal{P} \\ \text{not mapped in } f}} \min_{v_{\mathcal{G}} \in C^{(n)}[v_{\mathcal{P}}]} \|z_{v_{\mathcal{P}}} - z_{v_{\mathcal{G}}}\|_2 \triangleq X. \quad (9)$$

$$\Delta^{(r)}_{\text{remain}} \geq \sum_{(h_{\mathcal{P}}, r_{\mathcal{P}}, t_{\mathcal{P}}) \in L[i:]} \min_{r_{\mathcal{G}} \in C^{(r)}[r_{\mathcal{P}}]} \|z_{r_{\mathcal{P}}} - z_{r_{\mathcal{G}}}\|_2 \triangleq Y. \quad (10)$$

Combining Equations (4), (9), and (10), we have

$$GSD(\mathcal{P}, \mathcal{S}) \geq \Delta^{(n)}_{\text{mapped}} + \Delta^{(r)}_{\text{mapped}} + X + Y \triangleq B. \quad (11)$$

When the lower bound B exceeds the largest GSD of the top-k subgraphs in current priority queue res, any subgraph $\mathcal{S}$ completed through future expansion will never become the desired top-k subgraphs. That is, the current partial mapping f can be safely discarded, effectively pruning subsequent unnecessary search branches.

Moreover, to reduce the largest GSD in the top-k priority queue res for more pruning opportunities, we adopt a greedy strategy that prioritizes matching more promising subgraphs earlier. Specifically, for lines 4-5, we can process the nodes $v_{\mathcal{G}} \in C^{(n)}[v^*_{\mathcal{P}}]$ in ascending order of their distances. In line 13 of the `Expand` function, the neighboring relation and node (r_G, t_G) with the smaller sum of $\|z_{t_P} - z_{t_G}\|_2 + \|z_{r_P} - z_{r_G}\|_2$ will be expanded earlier.

By combining the pruned and greedy expansion strategies, the optimized algorithm is guaranteed to produce the same results as the top-k retrieval algorithm without any loss in solution quality. The experiments in Section 6.6 show that the optimized algorithm significantly accelerates retrieval.

6 Experiments

We conduct experiments on tasks of *Knowledge Graph Question Answering (KGQA)* and *Fact Verification*. Our code is available at https://anonymous.4open.science/r/SimGRAG.

6.1 Tasks and Datasets

Knowledge Graph Question Answering. We use the MoviE Text Audio QA dataset (MetaQA) (Zhang et al., 2018) related to the field of movies. All the queries in the test set are adopted for evaluation, consisting of Vanilla 1-hop, 2-hop, and 3-hop question-answering in the same field.

Fact Verification. We adopt the FactKG dataset (Kim et al., 2023b), in which colloquial and written style claims can be verified using the DBpedia (Lehmann et al., 2015). All statements in the test set are used in the evaluation, and a method should return *Supported* or *Refuted* after verification.

Please refer to Appendix A for detailed statistics and examples of the tasks and datasets.

6.2 Baselines

The included baselines are briefly introduced as follows. Please refer to Appendix C for more details.

Supervised task-specific models. State-of-the-art models for KGQA include EmbedKGQA (Saxena et al., 2020), NSM (He et al., 2021), and UniKGQA (Jiang et al., 2022). They are trained on the MetaQA training set and evaluated by the test accuracy. For the task of fact verification, the KG version of GEAR (Zhou et al., 2019) is included and trained on the FactKG training set.

Pre-trained LLMs. For both tasks, we evaluate two popular LLMs, ChatGPT (OpenAI, 2024) and Llama 3 70B (Dubey et al., 2024), using 12-shots without any provided evidence.

KG-driven RAG with training. Recent method KELP (Liu et al., 2024) trains the retriever over the training set, while G-retriever (He et al., 2024) trains a graph neural network (GNN) to integrate query texts and subgraph evidences.

KG-driven RAG without training. Both KAPING (Baek et al., 2023) and KG-GPT (Kim et al., 2023a) only require retrieval subgraphs from the KGs without any training or fine-tuning.

6.3 Comparative Results

As summarized in Table 1, supervised task-specific methods outperform KG-driven RAG approaches that require additional training. Notably, supervised task-specific methods generally require

Method	MetaQA (Hits@1)			FactKG
	1-hop	2-hop	3-hop	(Accuracy)
Supervised task-specific methods				
EmbedKGQA	97.5	98.8	94.8	-
NSM	97.1	99.9	98.9	-
UniKGQA	97.5	99.0	99.1	-
GEAR	-	-	-	77.7
Pre-trained LLMs				
ChatGPT	60.0	23.0	38.7	68.5
Llama 3 70B	56.7	25.2	42.3	68.4
KG-driven RAG with training (Llama 3 70B)				
KELP†	94.7	96.0	-	73.3
G-Retriever†	98.5	87.6	54.9	61.4
KG-driven RAG without training (Llama 3 70B)				
KAPING	90.8	71.2	43.0	75.5
KG-GPT†	93.6	93.6	88.2	69.5
SimGRAG (ours)	98.0	98.4	97.8	86.8

Table 1: Performance comparison of different approaches, where † denotes we provide oracle entities as it is the default setting of a method. Each reported value serves as an upper bound for the result obtained without oracle entities. Appendix D presents more discussions.

Step	MetaQA			FactKG
	1-hop	2-hop	3-hop	
Query-to-pattern	89%	36%	31%	49%
Pattern-to-subgraph	0%	0%	0%	24%
Augmented generation	11%	64%	69%	27%

Table 2: The statistics of errors from different steps.

	MetaQA (Hits@1)			FactKG
	1-hop	2-hop	3-hop	(Accuracy)
shot=4	98.6	96.5	92.8	84.0
shot=8	98.3	96.4	98.8	87.9
shot=12	98.0	98.4	97.8	86.8
$k = 1$	95.2	98.2	97.0	88.1
$k = 2$	98.0	97.9	97.6	87.6
$k = 3$	98.0	98.4	97.8	86.8

Table 3: Performance of the SimGRAG method by varying the number of few-shot examples and the parameter k for semantic guided subgraph retrieval.

smaller model sizes and lower training costs, making them a more cost-effective option in practice.

Directly using LLMs leads to the poorest performance. As for KG-driven RAG methods without additional training, SimGRAG shows substantially higher Hits@1 and accuracy in most cases. In fact, SimGRAG performs comparably to supervised task-specific models and even outperforms the supervised GEAR method on the FactKG dataset.

Moreover, the performance gap between SimGRAG and other RAG approaches becomes larger as the complexity of the questions increases on the MetaQA dataset. As discussed in Section 4.2, the combined use of graph isomorphism and semantic similarity effectively reduces noise and ensures conciseness, thus benefiting the performance of SimGRAG for 2-hop and 3-hop questions.

6.4 Error Analysis

Table 2 summarizes the error distribution across the three main steps of the SimGRAG method. For detailed error examples, please refer to Appendix E.

Many errors occur during the query-to-pattern alignment step, where the LLM fails to follow the given instructions and examples, thereby generating the undesired pattern graphs. Generally, both 2-hop and 3-hop queries roughly have consistent proportion of errors. But there are more errors on 1-hop queries, since we use the same few-shot examples for all MetaQA queries, which are all about 2-hop or 3-hop patterns. They make the LLM sometimes generate 2-hop patterns for 1-hop queries.

As the complexity of the queries increases in the MetaQA dataset, we also observe a higher incidence of errors in the subgraph-augmented generation step, since it is more difficult for the LLM to accurately extract relevant information for a complex question from the retrieved subgraphs.

On the FactKG dataset, errors are also encountered during the pattern-to-subgraph alignment. In these cases, while the LLM generates reasonable subgraphs in line with the guidance, mismatches occur because the ground-truth subgraphs have different structures and thus cannot be successfully aligned, as illustrated in Appendix E.

6.5 Ablation Studies

Few-shot in-context learning. Table 3 evaluates the proposed SimGRAG method by varying the number of examples in the prompts for the LLM, used in both pattern-to-graph alignment and verbalized subgraph-augmented generation. For the simplest MetaQA 1-hop questions, performance is not sensitive to the number of shots. In contrast, for more complex queries like those in the MetaQA 3-hop and FactKG datasets, we observe significant improvements when increasing from 4 to 8 shots.

Parameter k for semantic guided subgraph retrieval. Table 3 reports the impact of parameter k for retrieving top-k subgraphs with the smallest graph semantic distance. For MetaQA 1-hop questions, setting $k = 1$ leads to a significant drop in Hits@1, since many movies share exactly the same title, and retrieving fewer subgraphs makes it more difficult to cover the ground-truth answer. For MetaQA 2-hop and 3-hop questions, the choice of k has a negligible impact on performance. Conversely, increasing k leads to a slight decrease in accuracy on the FactKG dataset, since the top-1 subgraph is often sufficient and including more subgraphs will introduce noise for LLM.

Dataset	Path			Conjunction		Star
	1-hop	2-hop	3-hop	2-hop	3-hop	3-hop
MetaQA 1-hop	100%	0	0	0	0	0
MetaQA 2-hop	0	100%	0	0	0	0
MetaQA 3-hop	0	0	100%	0	0	0
PQ 2-hop	0	100%	0	0	0	0
PQ 3-hop	0	0	100%	0	0	0
WC2014	64%	14%	0	22%	0	0
FactKG	32%	28%	5%	17%	8%	10%

Table 4: Distribution of query pattern structures.

Method (Hits@1)	**PQ 2-hop**	**PQ 3-hop**	**WC2014**
KAPING	41.0	52.1	88.1
G-Retriever	61.8	46.7	67.5
SimGRAG (ours)	88.7	78.6	98.1

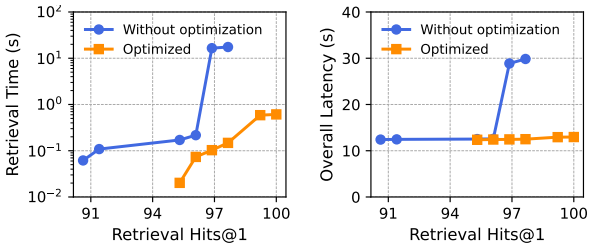
Table 5: Performance comparison on new datasets.

Query pattern structure. We introduce two additional KGQA datasets: PathQuestions (PQ) (Zhou et al., 2018) developed from Freebase (Bollacker et al., 2008), and WorldCup2014 (WC2014) (Zhang et al., 2016) with sports-domain KGs, as detailed in Appendix A. As outlined in Appendix F, we categorize query pattern structures into six classes and show the distributions in Table 4. Both Tables 1 and 5 confirm that SimGRAG outperforms RAG baselines on multi-hop path queries, and it is also better on WC2014 dataset that contains 22% 2-hop conjunction queries. By further experiments on each category of queries for FactKG dataset, SimGRAG achieves the accuracies of 89%, 88%, and 85% on 2-hop conjunction, 3-hop conjunction, and 3-hop star queries, respectively.

6.6 Retrieval Efficiency

As discussed in Section 5, we first perform a vector search to obtain the top-$k^{(n)}$ candidate nodes and top-$k^{(r)}$ candidate relations. Table 6 reports the average retrieval time per query, in which the vector search time dominates the total time. On the 10-million-scale DBpedia KG from the FactKG dataset, the overall retrieval time is 0.74 seconds per query, highlighting the efficiency and scalability of the optimized retrieval algorithm.

Additionally, we conduct a grid search over the parameters $k^{(n)}$ and $k^{(r)}$ to compare the top-k retrieval and the optimized algorithms. Please refer to Appendix G for detailed setups. Figure 5(a) presents the Pareto optimal curves, which plot the trade-off between average retrieval time and retrieval Hits@1. The results clearly show that the optimized retrieval algorithm significantly improves the performance, particularly in scenarios where a higher retrieval Hits@1 is desired in practice. Also, Figure 5(b) shows the overall latency for the proposed SimGRAG method, in which the optimized algorithm guarantees reasonable latency.



(a) Retrieval Time in SimGRAG (b) Overall Latency of SimGRAG

Figure 5: Pareto optimal curves for retrieval.

	MetaQA			FactKG
	1-hop	2-hop	3-hop	
Vector search	0.02	0.02	0.02	0.59
Optimized retrieval	0.0006	0.0007	0.002	0.15
Total	0.02	0.02	0.02	0.74

Table 6: Semantic guided subgraph retrieval time (s).

Method	MetaQA			FactKG
	1-hop	2-hop	3-hop	
KELP	3.6	4.8	-	5.4
G-Retriever	4.1	4.3	4.4	5.4
KAPING	3.3	5.9	8.8	10.6
KG-GPT	10.1	12.3	13.1	13.3
Ours (4-shot)	5.5	5.9	7.3	10.2
Ours (12-shot)	9.0	9.1	11.9	14.2

Table 7: Comparison of average query latency (s).

6.7 Overall Latency

We run each method on a NVIDIA A6000 GPU using Ollama 4-bit quantization for Llama3 70B. Table 7 reports the average latency for answering each query. Generally, our method has similar latency compared with others using the default 12-shot in-context learning. It could be much faster with 4-shot learning while still providing competitive performance, as confirmed by Table 3.

7 Conclusion

In this paper, we investigate the problem of KG-driven RAG and introduce a novel SimGRAG approach that effectively aligns query texts with KG structures. For query-to-pattern alignment, we employ an LLM to generate a pattern graph that aligns with the query text. For pattern-to-subgraph alignment, we introduce the Graph Semantic Distance (GSD) metric to quantify the alignment between the desired pattern and the underlying subgraphs in the KG. Additionally, we propose an optimized algorithm to retrieve the top-k similar subgraphs with the smallest GSD, improving retrieval efficiency and scalability. Extensive experiments demonstrate that SimGRAG consistently outperforms existing KG-driven RAG approaches.

8 Limitations

The performance of SimGRAG method is closely tied to the underlying capabilities of the large language model (LLM). Specifically, the method relies heavily on the ability of LLMs to understand and follow instructions effectively in both steps of the query-to-pattern alignment and verbalized subgraph-augmented generation. Thus, the performance of SimGRAG can be substantially degraded when utilizing lower-quality or less capable LLMs, especially in scenarios involving more complex queries that demand advanced reasoning skills.

Furthermore, following the characteristics of KGs used by existing studies (Baek et al., 2023; Kim et al., 2023a; Liu et al., 2024; He et al., 2024), we also assume that our input KG aligns with human cognition. It is a key requirement for the plug-and-play usability for the SimGRAG method. However, when using industrial domain-specific KGs which diverge significantly from commonly used schemas, it is challenging for LLMs to predict the desired nodes, edges or pattern structures during the query-to-pattern alignment stage. It is still under exploration how effectively fine-tuning LLMs can help to align generated patterns with such special KG structures.

Additionally, for domain-specific KGs, linking query entities to corresponding candidate entities in the KG could be challenging, particularly when the embedding model has not been trained on such data. Therefore, rather than relying on a plug-and-play embedding model, future work may fine-tune the embedding models on domain-specific data or explore alternative entity linking approaches.

References

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. 2024. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106, Toronto, Canada. Association for Computational Linguistics.

Leonard M. Blumenthal. 1953. *Theory and Applications of Distance Geometry*. Oxford, at the Clarendon Press, Oxford, UK.

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*, pages 1247–1250.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2309.07597.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2023. Lift yourself up: Retrieval-augmented text generation with self-memory. In *Advances in Neural Information Processing Systems*, volume 36, pages 43780–43799. Curran Associates, Inc.

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

Wenqi Fan, Shijie Wang, Jiani Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Hui Liu, Xiaorui Liu, Dawei Yin, et al. 2024. Graph machine learning in the era of large language models (llms). *arXiv preprint arXiv:2404.14928*.

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

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 *Proceedings of the 14th ACM international conference on web search and data mining*, pages 553–561.

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. *arXiv preprint arXiv:2402.07630*.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. 2022. UniKGQA: Unified retrieval and reasoning for solving multi-hop question answering over knowledge graph. *arXiv preprint arXiv:2212.00959*.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023a. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. *arXiv preprint arXiv:2310.11220*.

Jiho Kim, Sungjin Park, Yeonsu Kwon, Yohan Jo, James Thorne, and Edward Choi. 2023b. FactKG: Fact verification via reasoning on knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16190–16206. Association for Computational Linguistics.

Jinsoo Lee, Wook-Shin Han, Romans Kasperovics, and Jeong-Hoon Lee. 2012. An in-depth comparison of subgraph isomorphism algorithms in graph databases. *Proc. VLDB Endow.*, 6(2):133–144.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Jinpeng Li, Hang Yu, Xiangfeng Luo, and Qian Liu. 2024a. COSIGN: Contextual facts guided generation for knowledge graph completion. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1669–1682. Association for Computational Linguistics.

Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *ArXiv*, abs/2309.12871.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2024b. A survey of graph meets large language model: Progress and future directions. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 8123–8131. International Joint Conferences on Artificial Intelligence Organization.

Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. 2024. Knowledge graph-enhanced large language models via path selection. *arXiv preprint arXiv:2406.13862*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *Preprint*, arXiv:2305.14283.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Qiheng Mao, Zemin Liu, Chenghao Liu, Zhuo Li, and Jianling Sun. 2024. Advancing graph representation learning with large language models: A comprehensive survey of techniques. *arXiv preprint arXiv:2402.05952*.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *Preprint*, arXiv:2402.06196.

Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*.

OpenAI. 2024. Chatgpt: A language model by openai. https://openai.com/index/chatgpt/.

Jeff Z. Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, Russa Biswas, Gerard de Melo, Angela Bonifati, Edlira Vakaj, Mauro Dragoni, and Damien Graux. 2023. Large Language Models and Knowledge Graphs: Opportunities and Challenges. *Transactions on Graph Data and Knowledge*, 1(1):2:1–2:38.

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

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

Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4498–4507.

Shixuan Sun and Qiong Luo. 2020. In-memory subgraph matching: An in-depth study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, page 1083–1098.

Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 2614–2627. Association for Computing Machinery.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6).

Yaoke Wang, Yun Zhu, Wenqiao Zhang, Yueting Zhuang, Liyunfei Liyunfei, and Siliang Tang. 2024b.

Bridging local details and global context in text-attributed graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14830–14841, Miami, Florida, USA. Association for Computational Linguistics.

Xin Xie, Ningyu Zhang, Zhoubo Li, Shumin Deng, Hui Chen, Feiyu Xiong, Mosha Chen, and Huajun Chen. 2022. From discrimination to generation: Knowledge graph completion with generative transformer. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 162–165. Association for Computing Machinery.

Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2024. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *IEEE Transactions on Knowledge and Data Engineering*.

Liwen Zhang, John Winn, and Ryota Tomioka. 2016. Gaussian attention model and its application to knowledge base embedding and question answering. *arXiv preprint arXiv:1611.02266*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's song in the ai ocean: A survey on hallucination in large language models. *Preprint*, arXiv:2309.01219.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Zhijie Zhang, Yujie Lu, Weiguo Zheng, and Xuemin Lin. 2024. A comprehensive survey and experimental study of subgraph matching: Trends, unbiasedness, and interaction. *Proc. ACM Manag. Data*, 2(1).

Zhiyuan Zhang, Xiaoqian Liu, Yi Zhang, Qi Su, Xu Sun, and Bin He. 2020. Pretrain-KGE: Learning knowledge representation from pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 259–266. Association for Computational Linguistics.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *Preprint*, arXiv:2402.19473.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: Graph-based evidence aggregating and reasoning for fact verification. *arXiv preprint arXiv:1908.01843*.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. *arXiv preprint arXiv:1801.04726*.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024a. Efficient tuning and inference for large language models on textual graphs. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 5734–5742. International Joint Conferences on Artificial Intelligence Organization.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024b. LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *Preprint*, arXiv:2305.13168.

Dataset	Underlying KG	# Entity nodes	# Relation edges	# Entity type	# Relation type
MetaQA	MetaQA	43,234	269,482	-	9
PathQuestions	PathQuestions	2,215	3,321	-	14
WC2014	WC2014	1,127	6,482	-	6
FactKG	DBpedia	9,912,183	42,879,918	467	522

Table 8: Statistics of the knowledge graphs.

Head	Relation	Tail
Champagne for Caesar	has genre	Comedy
High Risk	starred actors	Lindsay Wagner
Married to It	directed by	Arthur Hiller
The Adventures of Huckleberry Finn	directed by	Michael Curtiz
The Amazing Spider-Man 2	directed by	Marc Webb
The Eiger Sanction	starred actors	Clint Eastwood
The Exterminating Angel	has tags	luis buñuel
The Life and Times of Hank Greenberg	has genre	Documentary
The Slumber Party Massacre	directed by	Amy Holden Jones
Tokyo Godfathers	release year	2003

Table 9: Example triples in the knowledge graph of MetaQA dataset.

Dataset	Example questions
MetaQA 1-hop	1. what films did Michelle Trachtenberg star in? 2. what are some words that describe movie Lassie Come Home? 3. who is the director of The Well-Digger's Daughter?
MetaQA 2-hop	1. which movies have the same actor of Jack the Bear? 2. which movies share the same director of I Wanna Hold Your Hand? 3. what were the release dates of Eric Mandelbaum written films?
MetaQA 3-hop	1. who wrote movies that share directors with the movie Unbeatable? 2. what genres do the movies that share directors with Fish Story fall under? 3. who acted in the films written by the screenwriter of The Man Who Laughs?

Table 10: Example questions in the MetaQA dataset.

A Details of Tasks and Datasets

Table 8 summarizes the statistics of the underlying knowledge graph used for each dataset.

A.1 Knowledge Graph Question Answering

For the task of Knowledge Graph Question Answering, we use the MoviE Text Audio QA dataset (MetaQA), which is designed for research on question-answering systems on knowledge graphs (Zhang et al., 2018). It provides a knowledge graph about movies, where entities include movie names, release years, directors, and so on, while the relations include starred actors, release year, written by, directed by, and so on. The queries are composed of Vanilla 1-hop, 2-hop, and 3-hop question answering in the field of movies. For the test set of MetaQA dataset, there are 9,947 questions for 1-hop, 14,872 for 2-hop, and 14,274 for 3-hop. Table 9 shows some example triples in the knowledge graph provided in the MetaQA (Zhang et al., 2018) dataset, while Table 10 are some example questions in the dataset. The MetaQA dataset is released under the Creative Commons Public License.

To investigate the impact of query pattern structures in Section 6.5, we further introduce PathQuestions and WC2014 datasets. *(i)* PathQuestions (Zhou et al., 2018) is built on Freebase KG (Bollacker et al., 2008), which contains 1,908 2-hop path queries and 5,198 3-hop path queries. Table 11 shows some example triples in the PathQuestions knowledge graph, while Table 12 are some example questions. *(ii)* WC2014 (Zhang et al., 2016) contains a knowledge graph about football players that participated in FIFA World Cup 2014. There are 10,162 queries of WC2014, which is a mixture of 6,482 single-relation questions, 1,472 two-hop path questions, and 2,208 conjunctive questions. Table 13 shows some example triples in the WC2014 knowledge graph, while Table 14 are some example questions. Both datasets are under a Creative Commons Attribution 4.0 International Licence.

A.2 Fact Verification

For the task of fact verification, we use the FactKG dataset (Kim et al., 2023b) that contains 5 different types of fact verification: One-hop, Conjunction, Existence, Multi-hop, and Negation, while all of them can be verified using the DBpedia knowledge

Head	Relation	Tail
eleanor_of_provence	children	beatrice_of_england
manuel_i_of_portugal	gender	male
joan_crawford	spouse	phillip_terry
barbara_of_portugal	spouse	ferdinand_vi_of_spain
empress_myeongseong	cause_of_death	regicide
frederica_of_mecklenburg-strelitz	spouse	ernest_augustus_i_of_hanover
henri_victor_regnault	gender	male
adelaide_of_lowenstein_wertheim_rosenberg	children	maria_josepha_of_portugal
prince_frederick_duke_of_york_and_albany	place_of_death	london
mary_boleyn	spouse	william_carey_1490

Table 11: Example triples in the knowledge graph of PQ dataset.

Dataset	Example questions
PQ 2-hop	1. john_b_kelly_sr's son's job? 2. what is the sex of spouse of mary_stuart_countess_of_bute? 3. where does virginia_heinlein's spouse work for?
PQ 3-hop	1. who is the mom of father of mary_of_teck's heir? 2. what is the name of the gender of son of henry_i_of_england's mother? 3. ferdinand_ii_of_aragon's parent's heir's nation?

Table 12: Example questions in the PQ dataset.

Head	Relation	Tail
Esseid_BELKALEM	plays_in_club	Watford_FC
Frank_LAMPARD	plays_for_country	England
JOAO_MOUTINHO	plays_in_club	AS_Monaco
Agustin_ORION	plays_position	Goalkeeper
Rickie_LAMBERT	is_aged	32
Pedro_RODRIGUEZ	plays_position	Forward
HENRIQUE	is_aged	27
OGC_Nice	is_in_country	France
Andres_GUARDADO	wears_number	18
Michel_VORM	plays_position	Goalkeeper

Table 13: Example triples in the knowledge graph of WC2014 dataset.

Dataset	Example questions
WC2014 1-hop	1. which football club does Alan_PULIDO play for? 2. what position does Alan_PULIDO play? 3. which country is the soccer team Atletico_Madrid based in?
WC2014 2-hop	1. which professional foootball team has a player from Belgium? 2. where is the football club that Rafael_MARQUEZ plays for? 3. which country does Mathieu_DEBUCHY play professional in?
WC2014 Conjunction	1. name a player who plays at Forward position at the club Tigres_UANL? 2. who are the Italy players at club US_Citta_di_Palermo? 3. which Portugal footballer plays at position Goalkeeper?

Table 14: Example questions in the WC2014 dataset.

graph (Lehmann et al., 2015). Its test set contains 9,041 statements to be verified. Table 15 shows some example triples in the DBpedia, while Table 16 are some example statements in the FactKG test set. The FactKG dataset is licensed with CC BY-NC-SA 4.0.

B Prompts

For query-to-pattern alignment, Table 17 shows the prompt for KGQA tasks, including MetaQA, PathQuestions and WC2014 datasets. Table 18 shows the prompt for the fact verification task, i.e., FactKG dataset.

For verbalized subgraph-augmented generation, Table 19 shows the prompt for KGQA tasks, including MetaQA, PathQuestions and WC2014 datasets. Table 20 shows the prompt for the fact verification task, i.e., FactKG dataset.

At each step for processing each dataset, our prompts utilize exactly the same guidances and few-shot examples across different query pattern structures. For example, all 1/2/3-hop queries in the MetaQA dataset share the identical prompt at the step of query-to-pattern alignment.

Head	Relation	Tail
Berlin	country	Germany
United States	governmentType	Republic
Harry Potter	author	J. K. Rowling
Albert Einstein	award	Nobel Prize in Physics
Terrance Shaw	college	Stephen F. Austin State University
Association for the Advancement of Artificial Intelligence	type	Scientific society
Nvidia	industry	Computer hardware

Table 15: Examples triples in the DBpedia knowledge graph used for FactKG dataset.

Example statements
1. It was Romano Prodi who was the prime minister.
2. Are you familiar with Terrance Shaw? He also attended college.
3. Yes, Anastasio J. Ortiz was the Vice President.

Table 16: Example statements from the FactKG dataset.

You need to segment the given query then extract the potential knowledge graph structures.

Notes)
1). Use the original description in the query with enough context, NEVER use unspecific words like 'in', 'appear in', 'for', 'of' etc.
2). For nodes or relations that are unknown, you can use the keyword 'UNKNOWN' with a unique ID, e.g., 'UNKNOWN artist 1', 'UNKNOWN relation 1'.
3). Return the segmented query and extracted graph structures strictly following the format:
{ "divided": ["segment 1", ...], "triples": [("head", "relation", "tail"), ...] }
4). NEVER provide extra descriptions or explanations, such as something like 'Here is the extracted knowledge graph structure'.

Examples)
1. query: "the actor in Flashpoint also appears in which films"
output: {
"divided": [
"the actor in Flashpoint",
"this actor also appears in another films",
],
"triples": [
("UNKNOWN actor 1", "actor of", "Flashpoint"),
("UNKNOWN actor 1", "actor of", "UNKNOWN film 1"),
]
}
2. query: ...
output: ...

Your task)
Please read and follow the above instructions and examples step by step
query: {{QUERY}}

Table 17: The query-to-pattern alignment prompt used for KGQA task.

C Implementations for Approaches

All programs are implemented with Python.

C.1 SimGRAG

Experiments are run with 1 NVIDIA A6000-48G GPU, employing the 4-bit quantized llama3 70B model within the Ollama framework. We use the Nomic embedding model (Nussbaum et al., 2024), which generates 768-dim semantic embeddings for nodes and relations. For retrieving similar nodes (resp. relations), we use HNSW (Malkov and Yashunin, 2018) algorithm implemented by Milvus vector database (Wang et al., 2021), with maximum degree $M = 64$, $efConstruction = 512$ and $efSearch = 8 * k^{(n)}$ (resp. $efSearch = 8 * k^{(r)}$).

By default, we use $k = 3$ and 12-shot in-context learning throughout all experiments, except for the ablation studies in Section 6.5. For MetaQA dataset, we use $k^{(n)} = k^{(r)} = 16$ by default. For the task of fact verification using FactKG dataset, we use $k^{(n)} = 16384$ and $k^{(r)} = 512$ by default, except for the grid search that evaluates the retrieval efficiency in Section 6.6. Moreover, for FactKG dataset, we further utilize the entity type associated

You need to segment the given query then extract the potential knowledge graph structures.

Notes)
1). Use the original description in the query with enough context, NEVER use unspecific words like 'in', 'appear in', 'for', 'of' etc.
2). For nodes or relations that are unknown, you can use the keyword 'UNKNOWN' with a unique ID, e.g., 'UNKNOWN artist 1', 'UNKNOWN relation 1'.
3). Return the segmented query and extracted graph structures strictly following the format:
{ "divided": ["segment 1", ...], "triples": [("head", "relation", "tail"), ...] }
4). NEVER provide extra descriptions or explanations, such as something like 'Here is the extracted knowledge graph structure'.

Examples)
1. query: "The College of William and Mary is the owner of the Alan B. Miller Hall, that is situated in Virginia."
output: {
"divided": [
"The College of William and Mary is the owner of the Alan B. Miller Hall",
"Alan B. Miller Hall is situated Virginia",
],
"triples": [
("The College of William and Mary", "owner", "Alan B. Miller Hall"),
("Alan B. Miller Hall", "situated in", "Virginia"),
]
}
2. query: ...
output: ...

Your task)
Please read and follow the above instructions and examples step by step
query: {{QUERY}}

Table 18: The query-to-pattern alignment prompt used in FactKG dataset.

with the entity nodes in DBpedia. Specifically, we construct a mapping that maps a type like "person" or "organization" to all its entity nodes. Then, for unknown entities in the pattern graph, such as "UNKNONWN person 1", we search for the top-$k^{(t)}$ similar types, then use all nodes with such similar types as the candidate nodes in the retrieval algorithm. By default, we set $k^{(t)} = 16$.

C.2 Pre-trained LLMs

For pre-trained LLMs including ChatGPT (OpenAI, 2024) and Llama 3 70B (Dubey et al., 2024) without training or augmented knowledge, we also use 12 shots in-context learning for fair comparison. For Llama 3 70B, experiments are run with 1 NVIDIA A6000-48G GPU, employing the 4-bit quantized model within the Ollama framework. The license of Llama 3 70B can be found at `https://www.llama.com/llama3/license/`.

C.3 KG-GPT

For evaluation, we use 1 NVIDIA A6000-48G GPU with the 4-bit quantized Llama3 70B model within the Ollama framework. We also use 12-shot in-context learning, and all other parameters are the same as their default setting (Kim et al., 2023a).

C.4 KELP

Experiments were conducted on 1 NVIDIA A6000-48G GPU system. Aligned with their settings (Liu et al., 2024), it involves fine-tuning a 66M-parameter DstilBert model with the AdamW optimizer at a learning rate of $2e-6$ and a batch size of 60. For fairness, we also use 12-shot in-context learning in the prompt. And we also use Llama 3 70B as the LLM, using the 4-bit quantized model within the Ollama framework.

C.5 G-Retriever

Experiments are performed on a system with 6 NVIDIA A6000-48G GPUs. The base LLM is the 4-bit quantized llama3 70B with frozen parameters. The Graph Transformer served as the GNN, configured with 4 layers, 4 attention heads, and a 1024-dimensional hidden layer. During training, we use the AdamW optimizer, a batch size of 4, and 10 epochs, with early stopping after 2 epochs. All the other parameters are the same with their default settings (He et al., 2024).

C.6 KAPING

For evaluation, we use 1 NVIDIA A6000-48G GPU with the 4-bit quantized Llama3 70B model

```
Please answer the question based on the given evidences from a knowledge graph.

Notes)
1). Use the original text in the valid evidences as answer output, NEVER rephrase or reformat them.
2). There may be different answers for different evidences. Return all possible answer for every evidence graph,
except for those that are obviously not aligned with the query.
3). You should provide a brief reason with several words, then tell that the answer.

Examples)
1. query: "who wrote films that share actors with the film Anastasia?"
   evidences: {
            "graph [1]": [
               ("Anastasia", "starred_actors", "Ingrid Bergman"),
               ("Spellbound", "starred_actors", "Ingrid Bergman"),
               ("Spellbound", "written_by", "Ben Hecht"),
            ],
            "graph [2]":[
               ("Anastasia", "starred_actors", "John Cusack"),
               ("Floundering", "starred_actors", "John Cusack"),
               ("Floundering", "written_by", "Peter McCarthy"),
            ]
   }
   answer: According to graphs [1][2], the writter is Ben Hecht or Peter McCarthy.
2. query: ...
   evidences: ...
   output: ...

Your task)
Please read and follow the above instructions and examples step by step
query: {{QUERY}}
evidences: {{RETRIEVED SUBGRAPHS}}
```

Table 19: The verbalized subgraph-augemented generation prompt used for KGQA task.

within the Ollama framework. Aligned with their recommended setting (Baek et al., 2023), we retrieve top-10 similar triples using MPNet as the retrieval model. And their prompt follows a zero-shot approach.

D Discussion about Oracle Entities

As discussed in Section 1, in real applications, users might not always know the precise entity IDs related to their query. Thus, an ideal approach should not require users to specify the oracle entities. However, both KG-GPT (Kim et al., 2023a) and KELP (Liu et al., 2024) expand subgraphs or paths from the user-provided oracle entities, while G-Retriever (He et al., 2024) restricts the KG to a 2-hop oracle entity neighborhood. In other words, they need to know which entities are exactly correct before running, and the search space will be constrained in the ground truth area, thereby reducing the problem hardness.

Though all methods will work better with the oracle entities, experimental evaluation in Table 1 shows that even when we allow certain baselines to benefit from using the oracle entities, their performance still underperforms the SimGRAG method that does not require such entities. In other words, if we do not provide them for such baselines, their performance may degrade further.

Moreover, it is intuitive to use the results of a top-k entity linker as a substitute for the oracle entity in certain baselines. However, it would significantly increase the computational complexity and latency, since these methods might need to run the entire pipeline independently for each candidate entity. In contrast, the SimGRAG method naturally avoids relying oracle entities without such independent redundant computations.

Furthermore, unlike existing approaches, the internal mechanism of SimGRAG method is designed to better handle and filter out those noisy entities in real-world KGs. As discussed in Section 4.2, we use the Graph Semantic Distance (GSD) metric, which can effectively incorporate somewhat distant entities or relations that still contribute valuable evidence to the overall subgraph. For example, Figure 4 shows that a candidate entity whose semantic distance is ranked 112 can still be used in the retrieved subgraph.

Generally, the proposed SimGRAG method is closer to the ideal feature through a carefully designed mechanism, while ensuring better perfor-

Please verify the statement based on the given evidences from a knowledge graph.

Notes)
1). If there is any evidence that completely supports the statement, the answer is 'True', otherwise is 'False'.
2). For questions like 'A has a wife', if there is any evidence that A has a spouse with any name, the answer is 'True'.
3). You should provide a brief reason with several words, then tell that the answer is 'True' or 'False'.

Examples)
1. query: "Mick Walker (footballer, born 1940) is the leader of 1993–94 Notts County F.C. season."
   evidences: {
   "graph [1]": [
   ('Mick Walker (footballer, born 1940)', 'manager', '1993–94 Notts County F.C. season'),
   ('Mick Walker (footballer, born 1940)', 'birthDate', '"1940-11-27"'),
   ],
   "graph [2]":[
   ('Mick Walker (footballer, born 1940)', 'manager', '1994–95 Notts County F.C. season'),
   ('Mick Walker (footballer, born 1940)', 'birthDate', '"1940-11-27"')
   ]
   }
   answer: As graphs [1][2] say that Mick Walker is the manager but not the leader, the answer is False.
2. query: ...
   evidences: ...
   output: ...

Your task)
Please read and follow the above instructions and examples step by step
query: {{QUERY}}
evidences: {{RETRIEVED SUBGRAPHS}}

Table 20: The verbalized subgraph-augemented generation prompt used in FactKG dataset.

mance than the baseline methods.

E Detailed Error Analysis

We manually categorize all the encountered errors of the SimGRAG method in our experiments.

The errors occurring during the query-to-pattern alignment step are defined as: LLM fails to follow the given instructions and examples. For example, for the query "The lady Anne Monson was born in the Darlington location of the ITL?" from FactKG dataset, the LLM gives the pattern graph with only one triple "('Anne Monson', 'born in', 'Darlington')", which is not aligned with the query text.

The error occurred during the subgraph-augmented generation step is defined as that given the correct retrieved subgraph, the LLM fails to provide the final correct response. For example, for the question "what films did Lucky McKee star in" from the MetaQA dataset, correct subgraphs of "[('Lucky McKee', 'starred_actors', 'Roman')]" is successfully retrieved, along with the two subgraphs with lower GSD ("[('Lucky McKee', 'directed_by', 'All Cheerleaders Die')]" and "[('Lucky McKee', 'directed_by', 'May')]"). However, the LLM gives the final response of "According to the evidences, there is no direct connection between Lucky McKee and a film they starred in. The graphs only mention that Lucky McKee directed films ('All Cheerleaders Die' and 'May'), but do not provide information about the films they acted in."

Errors occurring during the pattern-to-subgraph alignment phase are defined as: LLM follows the given instructions and examples to generate a satisfactory pattern graph, but the retrieval algorithm fails to retrieve the ground-truth subgraph for the query. It is because the ground-truth subgraphs have different structures and thus cannot be successfully aligned with the ground-truth subgraphs. For example, for the query "A food is classed as a Dessert and can be served warm (freshly baked) or cold.", the LLM-generated pattern graph is "[('UNKNOWN food 1', 'classed as', 'Dessert'), ('UNKNOWN food 1', 'served', '"warm"'), ('UNKNOWN food 1', 'served', '"cold"')]". However, the ground-truth subgraphs have the structure like "[('The food name', 'classed as', 'Dessert'), ('The food name', 'served', '"warm (freshly baked) or cold"')]".

F Query Pattern Structures

Following the previous study (Zhang et al., 2016), all queries (i.e., the pattern structure) in our experiments can be categorized into the following six types. For simplicity, we focus on topological

structures and ignore the edge directions.

- *1-hop Path:* Find an edge from a known subject s to another entity e.
- *2-hop Path:* Find a 2-hop path from a known subject s to another known or unknown entity e.
- *3-hop Path:* Find a 3-hop path from a known subject s to another known or unknown entity e.
- *2-hop Conjunction:* Find two distinct edges that link two known subjects s_1 and s_2 with an unknown entity e.
- *3-hop Conjunction:* Find an edge that links an known subject s_1 with an unknown entity e, as well as a 2-hop path that connects another known subjects s_2 with the same entity e.
- *3-hop Star:* Find three distinct edges that links to the same known or unknown entities e.

G Parameters for Grid Search

We conduct the grid search for evaluating the top-k retrieval algorithm and its optimized one on the FactKG dataset using DBpedia knowledge graph. Specifically, we randomly sample 100 queries that correctly generate patterns and manually identify the ground truth subgraphs for each query to evaluate retrieval performance using retrieval Hits@1. We fix $k = 1$ and try all combinations of the other parameters $k^{(n)} \in \{128, 256, 512, 1024, 2048, 4096, 8192, 16384\}$, $k^{(r)} \in \{128, 256, 512\}$, $k^{(t)} \in \{1, 2, 4, 8, 16\}$. For 100 queries, any program run out of the time limit of 10,000 seconds will be terminated and not reported. In Figure 5, the point at retrieval Hits@1=1.0 is achieved by using $k^{(n)} = 16384$, $k^{(r)} = 512$ and $k^{(t)} = 16$.