PolyG: Adaptive Graph Traversal for Diverse GraphRAG Questions

Renjie Liu[1], **Haitian Jiang**[2], **Xiao Yan**[3], **Bo Tang**[1], **Jinyang Li**[2]
Southern University of Science and Technology[1], New York University[2], Wuhan University[3]
liurj2023@mail.sustech.edu.cn

Abstract

GraphRAG enhances large language models (LLMs) to generate quality answers for user questions by retrieving related facts from external knowledge graphs. However, current GraphRAG methods are primarily evaluated on and overly tailored for knowledge graph question answering (KGQA) benchmarks, which are biased towards a few specific question patterns and do not reflect the diversity of real-world questions. To better evaluate GraphRAG methods, we propose a *complete four-class taxonomy* to categorize the basic patterns of knowledge graph questions and use it to create PolyBench, a new GraphRAG benchmark encompassing a *comprehensive* set of graph questions. With the new benchmark, we find that existing GraphRAG methods fall short in *effectiveness* (i.e., quality of the generated answers) and/or *efficiency* (i.e., response time or token usage) because they adopt either a *fixed* graph traversal strategy or *free-form* exploration by LLMs for fact retrieval. However, different question patterns require distinct graph traversal strategies and context formation. To facilitate better retrieval, we propose PolyG, an *adaptive* GraphRAG approach by decomposing and categorizing the questions according to our proposed question taxonomy. Built on top of a unified interface and execution engine, PolyG *dynamically* prompts an LLM to generate a graph database query to retrieve the context for each decomposed basic question. Compared with SOTA GraphRAG methods, PolyG achieves a higher win rate in generation quality and has a low response latency and token cost. Our code and benchmark are open-source at `https://github.com/Liu-rj/PolyG`.

1 Introduction

Large Language Models (LLMs) [1–3] have achieved remarkable success for various natural language processing (NLP) tasks [4–6]. However, they are susceptible to hallucinations when generating answers for questions that require information beyond their knowledge [7, 8]. To tackle this problem, graph-based retrieval-augmented generation (GraphRAG) [9] has emerged as an effective approach, which grounds LLM's responses using external knowledge graphs [10–12]. When answering a user question, GraphRAG first retrieves the relevant entities and relations from the knowledge graph and then utilizes this information along with the question to prompt the LLM to generate the answer.

Limitation of existing GraphRAG evaluation. While many GraphRAG solutions are proposed, they are primarily evaluated on Knowledge-Graph Question-Answering (KGQA) benchmarks. The questions in these benchmarks mainly ask about the concrete attributes of target entities with specific relation constraints on the related entities. As a result, these benchmarks cannot capture the diversity of real-world graph questions as shown in Table 1, based on our question taxonomy in section 3. The lack of question diversity leads new GraphRAG methods to overfit specific question patterns in their development and thus hinders their ability to excel in real-world applications.

Table 1: Question coverage and data source of our PolyBench and existing KGQA benchmarks. In the question pattern, s means subject, p is predicate chain, o refers to object, and $*$ is the missing component. WIKIQA inspires our benchmark and targets text-understanding task rather than KGQA.

Benchmark	Basic Question Pattern				Data Source
	$\langle s, *, * \rangle$	$\langle s, p, * \rangle$	$\langle s, *, o \rangle$	$\langle s, p, o \rangle$	
SimpleQ [23] (2015)		✓			Freebase Fact
WebQSP [24] (2016)	✓	✓			Google Suggest API
GraphQ [25] (2016)		✓			Random Construct.
CWQ [26] (2018)		✓			WebSQP
GrailQA [27] (2021)		✓			Random Construct.
RGBench [19] (2024)		✓			Question Template
CypherBench [28] (2025)		✓			Question Template
WIKIQA [29] (2015)	✓	✓	✓		Bing Query Log
PolyBench (Ours)	✓	✓	✓	✓	Question Template

To enable more comprehensive evaluation, we introduce a new GraphRAG benchmark called *PolyBench* (shown in Table 1) that includes a *comprehensive* and *complete* set of question patterns. To do so, we create a taxonomy of graph question patterns by abstracting graph question using a fact triple $\langle$*subject*, *predicate chain*, *object*$\rangle$ and classifying questions *depending on the missing component* in the triple. Under this taxonomy, a question can inquire about specific or unspecified attributes of an entity, explore potential relations between two entities, or verify the existence of a relation. Moreover, our benchmark also includes nested questions consisting of multiple basic questions from the taxonomy to assess the ability of GraphRAG methods on handling complex questions.

Limitations of existing GraphRAG methods. Most existing GraphRAG methods either use a *fixed* graph traversal strategy or rely on LLMs for *free-form* exploration. For example, Microsoft GraphRAG [9] (hereafter referred to as MS_GraphRAG) and LightRAG [13] adopt one-hop Breadth-First Search (BFS) to retrieve information from the neighborhood of the extracted entities; RoG [14] deduces concrete and faithful reasoning paths starting from the question's entities and use them for traversal; Fast-GraphRAG [15] and HippoRAG [16] use the personalized PageRank (PPR) score [17] to select the top paths; while ToG [18] and Graph-CoT [19] invoke the LLM at each traversal step to select the next entity to visit in the knowledge graph. Recently, methods such as G-retriever [20], GNN-RAG [21] and SubgraphRAG [22] propose trainable retriever that flexibly adapts to each query. However, they need to train the retriever on the KGQA benchmark and thus are not applicable to general GraphRAG scenarios where questions typically do not have a golden ground truth.

Since the traversal strategy of most existing GraphRAG methods is either *fixed* or *free-form* while the question patterns in GraphRAG workloads can be diverse, they suffer from two limitations.

- *Limited effectiveness*: Using a fixed graph traversal strategy works well for a specific question pattern but falls short for other patterns. For instance, using one-hop neighbor expansion, MS_GraphRAG and LightRAG suit questions that ask about general aspects of entities but cannot handle questions involving entities that are more than one-hop apart. Relying on the question to deduce the reasoning paths, RoG works well when the question provides explicit relation constraints but fails when these constraints are missing or vague. Fast-GraphRAG and HippoRAG are good for questions without constraints by ranking possible paths using their PPR scores but the selected top-ranking paths may be irrelevant when the question has explicit constraints.
- *High cost*: By using an LLM to determine the next entity to explore at every traversal step, ToG and Graph-CoT can handle all question patterns. However, they incur significant overhead due to frequent LLM invocations, increasing the end-to-end response time by approximately 2× and the number of used tokens by 10× according to our profiling. Moreover, they often do not produce the best response quality by focusing on fine-grained bootstrapping with a narrow view of knowledge.

Our solution PolyG. To tackle the limitations of existing solutions, we propose PolyG, which achieves high accuracy for diverse graph questions while maintaining efficiency. As shown in Figure 1, PolyG first categorizes the input user question into basic or nested questions based on our four-class question taxonomy. Then, for nested questions, PolyG further decomposes the overall question into several basic questions and adaptively prompts the LLM to generate graph database queries to retrieve the relevant information. In particular, PolyG generates Cypher queries [30] which

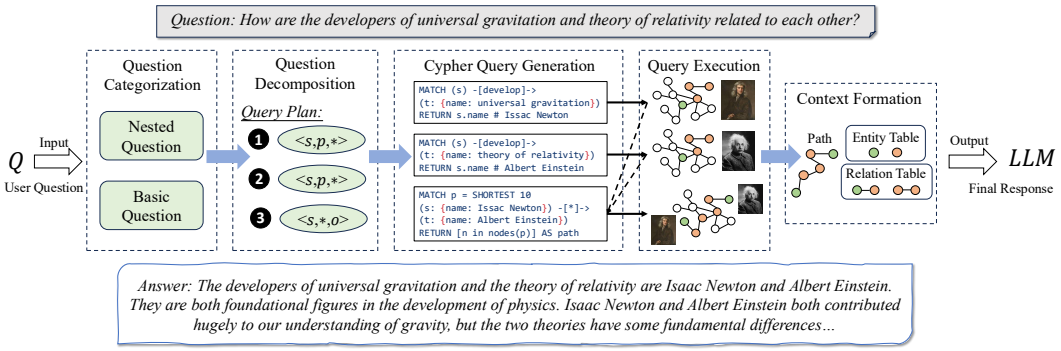


Figure 1: The workflow of our PolyG. PolyG first categorizes the user question according to its complexity (i.e., basic or nested). Then, a query plan is produced where each step is a basic graph question. For each step, PolyG generates a concrete Cypher query to retrieve information from the knowledge graph. Finally, PolyG forms the final context and prompt the LLM to generate response.

are supported by many popular graph databases and the retrieved results are organized into a concise and clear format for the LLM to generate a high-quality response. To our knowledge, PolyG is the first query planner for GraphRAG, and by introducing an explicit query planning stage, PolyG also opens the door for other advanced query processing strategies, such as query rewriting and merging.

Evaluated on PolyBench, we compare the GraphRAG methods in terms of answer quality, end-to-end latency, and token usage. Extensive experimental results show that our PolyG consistently outperforms all baselines, achieving the highest win rates in all evaluation criteria for answer quality, and meanwhile reduces the response latency by up to 2× and the token consumption by up to 90%.

To summarize, we make the following key contributions.

- We systematically study the query patterns in real graph questions and observe the lack of question diversity in current KGQA benchmarks that are used to evaluate GraphRAG methods.
- We introduce a new benchmark, PolyBench, tailored for GraphRAG workloads that encompass a *comprehensive* and *complete* set of question patterns guided by a four-class taxonomy.
- We develop a general GraphRAG solution, PolyG, that automatically categorizes and decomposes questions into basic queries, and adaptively prompts the LLM to generate the executable Cypher query to retrieve necessary context needed to answer the question.

2 GraphRAG Basics

Knowledge graph. A knowledge graph is typically defined as $\mathcal{KG} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of entities and $\mathcal{E}$ is the set of relations. Each entity $v \in \mathcal{V}$ and relation $e \in \mathcal{E}$ is often associated with specific attributes. For example, in an academic knowledge graph, entities may have types such as "*paper*", while relations may represent semantics such as "*cite*". Moreover, knowledge graphs commonly include inverse edges to capture mutual relationships. Specifically, for an edge $e = (u, r, v)$, where u and v are entities and r is a relation type, the inverse edge is defined as $e' = (v, r^{-1}, u)$, where r^{-1} denotes the inverse of relation r.

Reasoning path. A reasoning path in a knowledge graph is a sequence of entities connected by relations. It is represented as $P = v_0 \rightarrow (e_0) \rightarrow v_1 \rightarrow (e_1) \rightarrow \cdots \rightarrow (e_n) \rightarrow v_n$, where $v_i \in \mathcal{V}$ and $e_i \in \mathcal{E}$ denotes the i-th entity and relation. Specifically, each reasoning path corresponds to a fact triple $\langle s, p, o \rangle$, where s is the subject (entity), p is the predicate chain (chain of relations) and o is the object (entity). For example, the reasoning path for the fact "*Universal gravitation was developed by Issac Newton who was born in 1643.*" is: `Universal Gravitation` $\rightarrow$ `(developer)` $\rightarrow$ `Isaac Newton` $\rightarrow$ `(born in)` $\rightarrow$ `1643`, where the subject s is "*Universal Gravitation*", the predicate chain p is "*was developed by Issac Newton who was born in*" and the object o is "*1643*".

GraphRAG. Given a natural language question Q and a knowledge graph KG, the main challenge facing a GraphRAG solution is to develop a retriever ψ that can extract relevant entities, relations,

Table 2: The 4 basic and some common nested question patterns in our taxonomy and their descriptions along with concrete examples. In a knowledge graph fact triple $\langle s, p, o \rangle$, s stands for the subject, p refers to the predicate chain, and o is the object.

	Question Pattern	Description & Example
Basic Pattern	$\langle s, *, * \rangle$	Questions about an entity (the subject) with no specific relation constraints (the predicate) and target entity (the object). The task is to answer a general question about the entity. Example: *"Who is Isaac Newton?"*
	$\langle s, p, * \rangle$	Questions about an entity (the subject) with specific relation constraints (the predicate) but misses the target entity (the object). The task is to answer one specific aspect of the entity. Example: *"What theories and principles has Isaac Newton developed?"*
	$\langle s, *, o \rangle$	Questions about any relations (the predicate) between two entities (the subject and object). The task is to provide the relations between two entities. Example: *"How is Isaac Newton and Albert Einstein related?"*
	$\langle s, p, o \rangle$	Questions about specific relations (the predicate) between two entities (the subject and object). The task is to check the existence of a specific relationship between the two entities. Example: *"Have Isaac Newton and Albert Einstein both contributed to the same same field of science?"*
Nested Pattern	$\langle s, *, * \rangle + \langle s, p, * \rangle$	Nested questions about general information of an unknown entity with specific relation constraints. Example: *"Tell me about the scientist who developed univsersal gravitation."*
	$\langle s, p, * \rangle + \langle s, p, * \rangle$	Nested questions about an entity with specific but convoluted relation constraints (non-linear chains of relations). Example: *"Who is the developer of universal gravitation and influenced by Galileo's discoveries?"*
	$\langle s, *, o \rangle + \langle s, p, * \rangle$	Nested questions about any relations between unknown entities with specific constraints. Example: *"How are the developers of universal gravitation and theory of relativity related to each other?"*
	$\langle s, p, o \rangle + \langle s, p, * \rangle$	Nested questions about specific relations between entities with specific constraints. Example: *"Do the developers of universal gravitation and theory of relativity share similar perspectives about gravitation?"*

and reasoning paths from the knowledge graph. The retrieved context is then provided to the LLM to generate the final answer A. This process is formalized as:

$$A = \mathcal{LLM}(Q, \psi(KG, Q)). \tag{1}$$

Existing GraphRAG methods mainly differ in how their retrievers traverse the knowledge graph and extract the relevant facts. Detailed descriptions of their workflow are provided in Appendix D.

3 Graph Question Patterns

As is discussed in section 1, the lack of question diversity in existing KGQA benchmarks motivates us to explore more question patterns that can appear in real-world scenarios.

Questions from real queries. Due to privacy concerns [31], real-world query logs are rarely made publicly available. However, we studied the WIKIQA benchmark [29], which is sampled from raw Bing query logs and serves as a proxy for real user questions. While existing KGQA benchmarks primarily cover structured factual questions, our analysis of WIKIQA reveals a broader range of question patterns. In addition to factual queries about entities or relationships, users also ask more descriptive or explanatory questions. For example, questions such as "*What is feedback mechanism in plants during respiration?*" and "*How are the directions of the velocity and force vectors related in a circular motion?*" highlight this diversity. These observations motivate us to develop a *complete* set of question patterns derived from the knowledge graph reasoning paths.

Basic question patterns. Based on the analysis of real graph queries, we observe that questions in knowledge graphs correspond to queries over different components of reasoning paths, aka fact triples $\langle s, p, o \rangle$, where s is the subject, p is the predicate chain, and o is the object. By masking each element of the triple as either known or unknown ($*$), a total of eight possible question patterns (2^3) can be generated. For example, a question of the form $\langle s, *, * \rangle$ could be "*Tell me about Isaac Newton?*". However, some combinations do not yield meaningful questions, while others are redundant. Specifically, the pattern $\langle *, *, * \rangle$ does not correspond to any valid query. Additionally, the patterns $\langle *, p, o \rangle$ and $\langle *, *, o \rangle$ are equivalent to $\langle s, p, * \rangle$ and $\langle s, *, * \rangle$, respectively, due to the existence of inverse relations in knowledge graphs. The pattern $\langle *, p, * \rangle$ necessitates a full scan of all relations in $\mathcal{KG}$, which typically does not correspond to meaningful questions, and thus is excluded from this work. Consequently, we identify four valid basic question patterns: $\langle s, *, * \rangle$, $\langle s, p, * \rangle$, $\langle s, *, o \rangle$, $\langle s, p, o \rangle$. In Table 2, we define each of these four patterns, specify their associated tasks, and provide illustrative examples. These four patterns lead us to the following proposition:

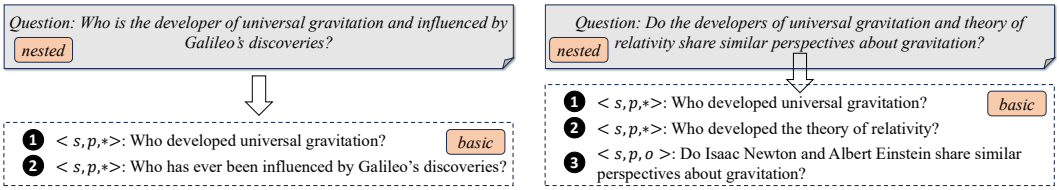


Figure 2: Examples of how nested questions can be decomposed into several basic questions.

Proposition 1. *Any valid graph question either belongs to one of the four basic question patterns* $\langle s, *, * \rangle$, $\langle s, p, * \rangle$, $\langle s, *, o \rangle$, *or* $\langle s, p, o \rangle$, *or can be decomposed into a set of sub-questions, each of which falls into one of these four basic patterns.*

Based on the taxonomy, we conducted a comprehensive analysis of the question patterns covered by existing KGQA benchmarks, as summarized in Table 1. Despite differences in their data curation processes, we observe that these benchmarks exhibit limited diversity in question types, typically covering only 2 out of the 8 patterns. Further details are provided in Appendix E. This lack of diversity limits the ability to evaluate GraphRAG's general capability in addressing real-world questions. It is worth noting that, WIKIQA does not cover all of the proposed patterns, because it samples a limited subset of queries from Bing logs that begin with a WH-word (e.g., "what" or "how"), and therefore does not fully reflect the diversity or distribution of questions in real-world scenarios.

Nested questions. In real-world scenarios, natural language questions over graphs can be complex and involve the nesting of multiple basic questions. For instance, recent KGQA benchmarks such as CWQ [26], GraphQ [26], and GrailQA [27] include questions with intricate fact constraints that go beyond a simple linear chain of relations. Consider the question: "*Who is the developer of both universal gravitation and the law of motion?*" This can be decomposed into two basic $\langle s, p, * \rangle$ sub-questions: "*Who developed universal gravitation?*" and "*Who formulated the law of motion?*". We observe that only questions with the $\langle s, p, * \rangle$ pattern can serve as internal sub-questions in nested queries, as they yield specific answers that can be passed to the outer question. Building on this, we identify three common nested question patterns: $\langle s, *, * \rangle + \langle s, p, * \rangle$, $\langle s, *, o \rangle + \langle s, p, * \rangle$, and $\langle s, p, o \rangle + \langle s, p, * \rangle$, which are composed of multiple basic question patterns. Definitions and examples of these nested question patterns are provided in Table 2. Furthermore, Figure 2 illustrates how nested questions can be decomposed into basic sub-questions.

Note that our taxonomy still may not capture all possible knowledge graph questions, particularly given the high flexibility of natural language. For example, questions can require global summaries [9] or combine more than two basic question patterns. However, such cases are relatively rare and still conform to Proposition 1—they can be decomposed into sub-questions that follow our four basic question patterns.

4 PolyBench: A Benchmark for GraphRAG with Diverse Questions

In this section, we first introduce the knowledge graphs used in PolyBench, then discuss how we generate concrete graph questions and paraphrase them into flexible expressions.

Data collection. We utilize knowledge graphs from the public graph reasoning benchmark GR-Bench [19], which contains 10 knowledge graphs spanning 5 general domains: academia, e-commerce, literature, healthcare, and legal, collected from various public sources. In PolyBench, we select 3 out of the 10 graphs—specifically from the academia [32], literature [33], and e-commerce [34] domains. Detailed statistics of the selected knowledge graphs are provided in Appendix F.

Question generation. Following principles used in prior benchmarks [26, 19, 27, 28], we generate initial fixed-form questions by randomly selecting entities that meet specific constraints and populating predefined question templates. Specifically, we design 73 well-crafted question templates across all three knowledge graphs, covering both basic and nested question patterns with varying levels of complexity. Each template is paired with a manually written Cypher query to retrieve entities that ensure the resulting questions are valid and answerable. In particular, inspired by the diverse complexity of $\langle s, p, * \rangle$ questions seen in recent KGQA benchmarks [26, 25, 27, 19], our templates include questions requiring multi-hop reasoning, with relation paths ranging from 1 to 5 hops.

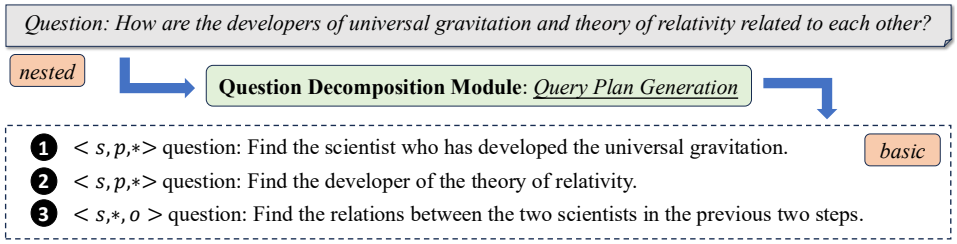


Figure 3: An example of the question decomposition in PolyG.

For higher-order relations, we carefully validate the semantics to avoid generating unrealistic or nonsensical questions. As a result, we generate 400 initial questions per knowledge graph, covering a broad range of patterns and complexities—yielding a total of 1,200 questions across the entire benchmark. The question templates are described in detail in Appendix G.

Partial answer annotation. Most questions in PolyBench do not have gold-standard ground truth answers, and LLM-based judgment is the common practice for evaluation [9, 15]. Nevertheless, we identify that ground truth can be provided for specific question types—namely, $\langle s, p, * \rangle$ and $\langle s, p, * \rangle + \langle s, p, * \rangle$ questions. For these templates, we manually author additional Cypher queries that, when populated with the selected entities during question generation, retrieve the corresponding answers. These ground truth answers enable quantitative evaluation using standard metrics such as *F1-score* and *Hit*, thereby strengthening the rigor of our benchmark assessment.

Question paraphrasing. As noted in existing KGQA benchmarks [26, 25, 27, 19, 28], template-based question generation often results in fixed and repetitive phrasing, which limits linguistic diversity. To address this, we follow the strategy adopted by recent benchmarks [19, 28] and use an LLM (Claude-3.5-Sonnet [35]) to paraphrase each initial question into four distinct variants. We then randomly select one of these paraphrased versions as the final question presented in PolyBench.

5 The PolyG System

In this section, we introduce PolyG, a general and effective GraphRAG solution designed to handle a wide range of graph questions. PolyG automatically classifies and decomposes user questions into a sequence of basic sub-questions. For each sub-question, it generates an appropriate Cypher query with adaptive prompting and self-correction mechanisms to retrieve relevant information. The responses to these sub-questions are then aggregated and fed into the LLM to produce a final answer.

As depicted in Figure 1, PolyG comprises five main stages: *question categorization*, *question decomposition*, *Cypher query generation*, *query execution* and *context formation*. In the following, we describe each stage in detail. The full prompt used in each module is provided in Appendix H.

Question categorization. Given a user question, PolyG prompts the LLM to classify it as either a *basic* or *nested* question by providing clear definitions of the basic question patterns along with few-shot examples. A basic question directly corresponds to one of the four defined patterns, whereas a nested question represents a composition of multiple basic questions. If the input is identified as a basic question, the LLM is also required to specify its exact pattern type. For instance, the question "*Who developed universal gravitation?*" is categorized as a basic question with the pattern $\langle s, p, * \rangle$.

Question decomposition. For nested questions, PolyG prompts the LLM to decompose the complex input into a sequence of basic questions. However, it is often not possible to generate concrete sub-questions without first obtaining intermediate results. For instance, given the question "*How are the developers of universal gravitation and the theory of relativity related to each other?*", we cannot form the sub-question "*How are Isaac Newton and Albert Einstein related?*" until we have identified these individuals as the developers in earlier steps. Therefore, rather than asking the LLM to generate fully instantiated sub-questions, we prompt it to produce a *query plan*, where each step describes a specific sub-question to be resolved. An example is shown in Figure 3, where the third

step in the query plan is: "*Find the relations between the two scientists identified in the previous two steps*," rather than a fully specified question.

Cypher query generation. Once the query plan is constructed with descriptive steps, PolyG instantiates a concrete basic question for each step before execution. At this stage, the LLM is given the full query plan along with all previously generated sub-questions and their responses, and is asked to generate a concrete question for the current step. PolyG then adaptively prompts the LLM to produce a faithful Cypher query by providing the graph schema (i.e., entity and relation types) and task-specific instructions. While the full schemas of large RDF knowledge graphs (e.g., Wikidata and Freebase) may exceed the LLM's context length, prior work [28, 36] has proposed techniques to convert RDF graphs into multiple smaller property graphs with manageable schema sizes. Since each of the four basic question patterns targets different components of a knowledge graph fact triple, as discussed in section 3, we observe that they necessitate *distinct graph traversal strategies*.

In Table 3, we summarize the graph traversal strategies employed for each basic question pattern. Specifically, $\langle s, *, * \rangle$ questions aim to retrieve general information and are best handled via *BFS neighbor expansion*, using queries such as ``MATCH (s)-[p]->(o) RETURN p, o''. This approach efficiently captures the immediate neighborhood of the subject s by gathering its one-hop neighbors and directly connected relations. For $\langle s, p, * \rangle$ questions, where the relation p is explicitly defined, we prompt the LLM to generate *concrete meta-paths* that connect the subject s to the target entity o. A typical Cypher query for this pattern is of the form ``MATCH (s)-[p1]->(o1)-[p2]->...->(on) WHERE... RETURN on'', where `n` denotes the number of hops in the relational chain and $p_1 ... p_n$ constitute p. This guided traversal not only accurately locates the answer but also reduces unnecessary context. For $\langle s, *, o \rangle$ and $\langle s, p, o \rangle$ questions, which inquire about the relations, we generate Cypher queries that perform a *top-k shortest paths* search to identify all relevant paths between s and o, using queries such as ``MATCH P = SHORTEST k (s)-[*]->(o) RETURN P''. In the case of $\langle s, p, o \rangle$ questions, where $p$ is known, we further prompt the LLM to apply filtering clauses (e.g., ``WHERE'') to ensure that only paths matching p are returned. Beyond shortest-paths, PolyG can also be extended to use random walk methods such as DeepWalk [37] and Node2Vec [38] to uncover meaningful reasoning paths.

Table 3: Correspondence from graph question patterns to specific graph traversal strategies.

Question Pattern	Graph Traversal Strategy
$\langle s, *, * \rangle$	BFS neighbor expansion
$\langle s, p, * \rangle$	meta-path guided walk
$\langle s, *, o \rangle$	top-k shortest paths
$\langle s, p, o \rangle$	top-k constrained shortest paths

Query execution. After the LLM generates a concrete, executable Cypher query for each basic question, PolyG executes the query asynchronously over the underlying graph database. However, even with clear specifications and few-shot examples tailored to each basic question pattern, the LLM may still generate erroneous Cypher queries—such as those with incorrect syntax or nonexistent relations—which can lead to execution errors or empty results. To address this, PolyG incorporates a *self-correction* mechanism that enables the LLM to automatically detect and refine faulty queries upon failure. Specifically, PolyG captures the execution error and sends it back to the LLM along with prior responses to prompt correction. Empirically, we find that setting the maximum number of retries to 3 strikes a good balance between quality and efficiency.

Context formation. Based on the results of each Cypher query from the previous stage, PolyG assembles a complete context to enable the LLM to generate a high-quality response. In addition to task-specific prompts, the core components of this context are the entity and relation tables. The entity table lists all attributes of the retrieved entities (e.g., node type and description), while the relation table aggregates all existing relations and their attributes between each pair of entities. Together, these two tables form the essential context that contains all necessary information to answer the question. Furthermore, for $\langle s, *, o \rangle$ and $\langle s, p, o \rangle$-type questions, we also include additional complete reasoning paths between the question entities to better demonstrate their interconnections. Finally, once PolyG completes all steps in the query plan, it compiles all sub-questions, their corresponding responses, and the original user question to prompt the LLM to generate a final summary.

6 Evaluation

In this part, we conduct experiments to evaluate PolyG and compare it with existing baseline solutions.

Table 4: Response generation quality and efficiency of our PolyG and the baselines on PolyBench. The win rates sum over 100% as we allow multiple winners for each question. **Bold faces** denote the best results. F1-score and Hit are only for $\langle s, p, * \rangle$ and $\langle s, p, * \rangle + \langle s, p, * \rangle$ questions.

	Criteria	MS_GraphRAG	RoG	Fast-graphrag	Graph-CoT	Cypher	PolyG
Claude-3.5-sonnet	Comprehensiveness	26.08%	28.75%	17.17%	19.42%	13.25%	**74.67%**
	Diversity	19.00%	26.33%	10.67%	15.83%	8.25%	**66.08%**
	Empowerment	24.42%	26.25%	15.83%	13.58%	12.67%	**64.83%**
	Directness	29.83%	15.75%	31.83%	42.42%	15.50%	**46.67%**
	Overall Winner	17.17%	19.25%	12.25%	13.83%	7.92%	**54.75%**
	F1-score	0.2094	0.6934	0.3304	0.3885	0.1750	**0.7084**
	Hit	0.4400	0.8867	0.6500	0.6500	0.3300	**0.8900**
	Latency (s)	**11.79**	16.75	45.84	53.81	14.84	25.38
	Token Usage	17,464	**1,417**	48,038	104,594	4,273	6,950
Deepseek-R1	Comprehensiveness	29.17%	30.00%	33.08%	9.17%	22.83%	**62.83%**
	Diversity	24.00%	24.00%	29.67%	4.17%	14.92%	**56.75%**
	Empowerment	29.75%	25.50%	31.00%	4.08%	20.42%	**50.58%**
	Directness	33.00%	22.33%	36.58%	25.17%	29.83%	**43.75%**
	Overall Winner	21.92%	21.83%	24.17%	5.08%	17.50%	**42.92%**
	F1-score	0.3246	0.6048	0.4752	0.2952	0.2835	**0.6153**
	Hit	0.4400	0.6767	0.7000	0.3867	0.3533	**0.6867**
	Latency (s)	**13.50**	17.05	46.76	93.02	23.03	22.40
	Token Usage	16,843	**1,534**	48,038	86,128	5,014	5,796

6.1 Experiment Setting

We compare PolyG against several baselines: MS_GraphRAG [9], RoG [14], Fast-graphrag [15], and Graph-CoT [19], on PolyBench. Additionally, we include a baseline named *Cypher*, which directly generates Cypher queries using the LLM without guidance from our proposed question pattern taxonomy to show the necessity of our adaptive prompting. To evaluate performance, we follow the evaluation practices of previous work [9, 13], assessing generation quality based on win rates judged by the LLM across five dimensions: *Comprehensiveness*, *Diversity*, *Empowerment*, *Directness*, and *Overall Winner*. Following previous works, we also report *F1-score* and *Hit* on the questions having golden ground-truth, namely of the type $\langle s, p, * \rangle$ and $\langle s, p, * \rangle + \langle s, p, * \rangle$, along with *response latency* and *token usage* as efficiency metrics. We evaluate the main results using both Claude-3.5-sonnet [35] and Deepseek-R1 [39] for generation, and Deepseek-R1 to give judgments. For ablation studies and micro-benchmarks, we provide the results with Claude-3.5-sonnet. Detailed experimental settings are provided in Appendix B.

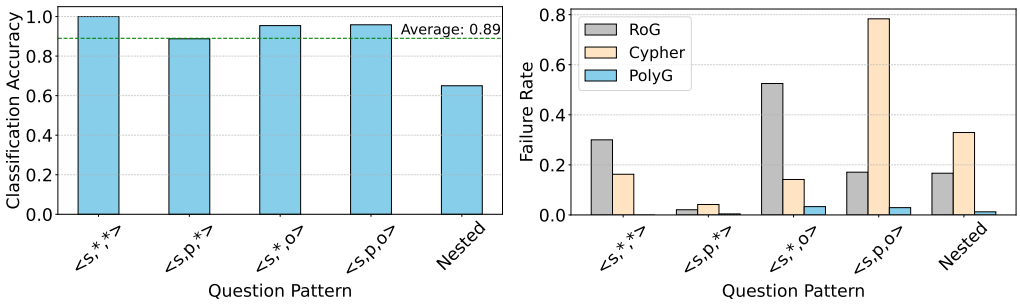
6.2 Experiment Results

Main results. Table 4 presents the generation quality and execution efficiency across all three knowledge graphs in PolyBench. In terms of generation quality, PolyG consistently achieves the highest win rates across all four evaluation criteria, with the most overall wins, as well as the highest F1-score and Hit. These results highlight the limitations of existing GraphRAG methods and underscore the importance of adaptive graph traversal in handling diverse graph questions. The superior performance of PolyG on both Claude-3.5-sonnet and Deepseek-R1 suggests its generalization across different LLMs. RoG achieves comparable quantitative scores to PolyG, as it is specifically tailored for $\langle s, p, * \rangle$ questions. While Graph-CoT is generally applicable to any question type due to the reasoning flexibility of LLMs, its win rates are lower because the LLM often explores only a subset of relevant entities and derives answers from a limited view of the knowledge graph. This results in less comprehensive and diverse responses compared to methods that retrieve all relevant entities. However, this narrow focus allows Graph-CoT to perform comparably to PolyG in terms of directness.

In terms of execution efficiency and cost (see Table 4), PolyG achieves superior generation quality without a significant increase in response latency or token usage. Instead, it maintains relatively low latency and substantially reduces token consumption compared to more resource-intensive baselines. Specifically, PolyG consistently records lower latency and requires fewer tokens than Fast-GraphRAG

Table 5: Win rates for each of the basic question patterns and nested questions.

Question Pattern	MS_GraphRAG	RoG	Fast-graphrag	Graph-CoT	Cypher	PolyG
$\langle s, *, * \rangle$	30.00%	23.75%	2.92%	26.25%	15.00%	**33.33%**
$\langle s, p, * \rangle$	10.83%	**47.08%**	24.58%	18.33%	13.33%	45.00%
$\langle s, *, o \rangle$	25.83%	9.17%	5.42%	10.00%	3.75%	**59.58%**
$\langle s, p, o \rangle$	4.17%	1.25%	20.42%	5.83%	3.75%	**70.42%**
Nested	15.00%	15.00%	7.92%	8.75%	3.75%	**65.42%**



(a) Accuracy of question categorization.

(b) Failure rate comparison.

Figure 4: Question categorization accuracy and failure rates of PolyG on each question pattern.

and Graph-CoT. This efficiency arises because Fast-GraphRAG relies on dynamic computation of Personalized PageRank (PPR) scores for each node during retrieval, and Graph-CoT invokes the LLM at every traversal step—both of which incur high computational complexity and resource usage. MS_GraphRAG also exhibits high token consumption by retrieving large amounts of irrelevant information, even when questions target specific aspects. Although RoG and Cypher offer shorter latency and lower token usage, they do not match PolyG's generation quality, which remains the primary criterion for GraphRAG applications.

Response quality for different question patterns. In Table 5, we break down the win rate by the four basic question patterns and nested questions. We observe that some baselines achieve win rates comparable to PolyG on $\langle s, *, * \rangle$ (e.g., MS_GraphRAG) and $\langle s, p, * \rangle$ (e.g., RoG) questions. For the $\langle s, *, * \rangle$ pattern, questions are typically simple and seek general information about entities; MS_GraphRAG is specifically designed and well-optimized for such tasks. RoG achieves similar win rates to PolyG on $\langle s, p, * \rangle$ questions because this pattern is the primary focus of existing KGQA benchmarks targeted by RoG. Consequently, similar performance trends are also observed on representative KGQA benchmarks, where PolyG matches RoG and outperforms all other baselines. Although specific baselines perform similarly to PolyG in terms of the "*Overall Winner*" metric for these two question patterns, PolyG outperforms them in other criteria (e.g., *Comprehensiveness*), making it a preferable choice for users with specific priorities. For all other patterns, PolyG constantly delivers superior response quality, achieving overall win rates above 60%.

Accuracy of question categorization. Figure 4a presents the accuracy of LLM-based question categorization in PolyG for different question patterns. The average accuracy is 89%, demonstrating that our question pattern taxonomy is clear and effective for the LLM to differentiate between the patterns. Notably, accuracy is lower for $\langle s, p, * \rangle$ and nested questions compared to the other three patterns. This is because PolyG tends to decompose complex $\langle s, p, * \rangle$ questions involving multi-hop relations into multiple steps, while it merges simple nested questions of the form $\langle s, p, * \rangle + \langle s, p, * \rangle$ into a single $\langle s, p, * \rangle$ question. However, such decomposition and merging do not impair the LLM's ability to generate quality responses, and thus we allow this intrinsic flexibility.

Failure analysis. Although the LLM is provided with the graph schema in PolyG, it can still generate incorrect Cypher queries—either semantically incorrect or returning empty results. To evaluate the effectiveness of our self-correction module during Cypher query generation, we report the failure rates broken down by the 4 basic and nested question patterns in Figure 4b. For comparison, we also include the failure rates of RoG and Cypher. The results show that PolyG successfully generates

correct Cypher queries for over 95% of the questions, while RoG and Cypher fail for 24% and 30% of tasks, respectively. Furthermore, without the LLM self-correction module, PolyG would fail for 70% of the nested questions. These findings suggest that our adaptive prompting and self-correction mechanism effectively reduce failures and enable more reliable Cypher query generation.

Additional experiments. In Appendix C, we provide additional experimental results. These include win rate decompositions across the other four criteria, detailed performance and efficiency comparisons on each dataset, and further analysis of the nested question pattern.

7 Conclusions

In this work, we develop a new benchmark, PolyBench, for general GraphRAG workloads, covering a wide range of graph question patterns to enable more reliable evaluation of GraphRAG methods. Furthermore, we propose a novel GraphRAG solution, PolyG, which excels at providing high-quality responses to diverse GraphRAG questions. PolyG is an adaptive framework that automatically identifies the question pattern and dynamically prompts the LLM to generate appropriate Cypher queries to retrieve the relevant context from the knowledge graphs. Experiment results show that PolyG achieves high response quality with a low response latency and token usage.

References

[1] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL `https://arxiv.org/abs/2307.09288`.

[2] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, and Sam Altman et al. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL `https://arxiv.org/abs/2005.14165`.

[4] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. In Jong C. Park, Yuki Arase, Baotian Hu, Wei Lu, Derry Wijaya, Ayu Purwarianti, and Adila Alfa Krisnadhi, editors, *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–718, Nusa Dua, Bali, November 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.ijcnlp-main.45. URL `https://aclanthology.org/2023.ijcnlp-main.45/`.

[5] Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. A complete survey on llm-based ai chatbots, 2024. URL `https://arxiv.org/abs/2406.16937`.

[6] Longyue Wang, Chenyang Lyu, Tianbo Ji, Zhirui Zhang, Dian Yu, Shuming Shi, and Zhaopeng Tu. Document-level machine translation with large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16646–16661, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.1036. URL https://aclanthology.org/2023.emnlp-main.1036/.

[7] Pranab Sahoo, Prabhash Meharia, Akash Ghosh, Sriparna Saha, Vinija Jain, and Aman Chadha. A comprehensive survey of hallucination in large language, image, video and audio foundation models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11709–11724, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.685. URL https://aclanthology.org/2024.findings-emnlp.685/.

[8] 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. Siren's song in the ai ocean: A survey on hallucination in large language models, 2023. URL https://arxiv.org/abs/2309.01219.

[9] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2024. URL https://arxiv.org/abs/2404.16130.

[10] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *KDD*, page 839–848, 2018.

[11] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *CoRR*, 2019.

[12] Laura Garton, Caroline Haythornthwaite, and Barry Wellman. Studying online social networks. *J. Comput. Mediat. Commun.*, 1997.

[13] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation, 2024. URL https://arxiv.org/abs/2410.05779.

[14] Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZGNWW7xZ6Q.

[15] Streamlined and promptable fast graphrag framework designed for interpretable, high-precision, agent-driven retrieval workflows., 2024. URL https://github.com/circlemind-ai/fast-graphrag.

[16] Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=hkujvAPVsg.

[17] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

[18] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=nnVO1PvbTv.

[19] Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Zheng Li, Ruirui Li, Xianfeng Tang, Suhang Wang, Yu Meng, and Jiawei Han. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In Lun-Wei Ku, Andre Martins, and Vivek

Srikumar, editors, *62nd Annual Meeting of the Association for Computational Linguistics, ACL 2024 - Proceedings of the Conference*, Proceedings of the Annual Meeting of the Association for Computational Linguistics, pages 163–184. Association for Computational Linguistics (ACL), 2024.

[20] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=MPJ3oXtTZl.

[21] Costas Mavromatis and George Karypis. GNN-RAG: Graph neural retrieval for large language model reasoning, 2025. URL https://openreview.net/forum?id=EVuANndPlX.

[22] Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025. URL https://openreview.net/forum?id=2NbxnNI94F.

[23] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015. URL http://arxiv.org/abs/1506.02075.

[24] Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2033. URL https://aclanthology.org/P16-2033/.

[25] Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. On generating characteristic-rich question sets for QA evaluation. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1054. URL https://aclanthology.org/D16-1054/.

[26] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *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, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1059. URL https://aclanthology.org/N18-1059/.

[27] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449992. URL https://doi.org/10.1145/3442381.3449992.

[28] Andy K. Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W. Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Mike Yang, Teddy Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Polycarpos Yiorkadjis, Kenny Osele, Gautham Raghupathi, Dan Boneh, Daniel E. Ho, and Percy Liang. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models, 2025. URL https://arxiv.org/abs/2408.08926.

[29] Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1237. URL https://aclanthology.org/D15-1237/.

[30] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 1433–1445, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450347037. doi: 10.1145/3183713.3190657. URL https://doi.org/10.1145/3183713.3190657.

[31] AOL, 2025. URL https://techcrunch.com/2006/08/07/aol-this-was-a-screw-up/.

[32] Database Group Leipzig. Dblp-scholar, 2022. URL https://dbs.uni-leipzig.de/file/DBLP-Scholar.zip.

[33] UCSD Book Graph. Goodreads, 2022. URL https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home.

[34] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 507–517, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341431. doi: 10.1145/2872427.2883037. URL https://doi.org/10.1145/2872427.2883037.

[35] Anthropic. Claude 3.5 sonnet, 2024. URL https://www.anthropic.com/.

[36] Renzo Angles, Harsh Thakkar, and Dominik Tomaszuk. Mapping rdf databases to property graph databases. *IEEE Access*, 8:86091–86110, 2020. doi: 10.1109/ACCESS.2020.2993117.

[37] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329569. doi: 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732.

[38] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939754. URL https://doi.org/10.1145/2939672.2939754.

[39] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He,

Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[40] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on Freebase from question-answer pairs. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1160/.

[41] Google. Google suggest api, 2025. URL https://developers.google.com/workspace/cloud-search/docs/reference/rest/v1/query/suggest.

A Impact Statement

While this work is focused solely on advancing the field of machine learning and is not expected to introduce any societal risks, we acknowledge that there are certainly cases where the large language model (LLM) applications can be possibly developed to cause unintended social implications, whether deliberately or inadvertently. For instance, the improper use of GraphRAG could inadvertently expose confidential user data from internal company databases to unauthorized parties.

B Detailed Experimental Settings

Hardware configurations. We conduct all the experiments on an AWS EC2 g5.16xlarge instance with 1 NVIDIA A10 GPU (24GB memory), 64-core vCPUs, 256G memory and 2TB NVMe SSD storage. For LLM service, we use AWS bedrock API with Claude-3.5-sonnet [35] and Deepseek-R1 [39] models.

Baselines and models. We compare PolyG against existing GraphRAG methods discussed in section 1. To ensure a fair and clear comparison, we categorize baselines based on their adopted graph traversal methods and select one representative from each category, despite minor differences in other modules. For instance, since both MS_GraphRAG and LightRAG use BFS to retrieve all one-hop neighbors, we select MS_GraphRAG as the representative method. Similarly, for traversal methods that invoke LLMs at each step, we compare with Graph-CoT instead of ToG. Additionally, since RoG requires fine-tuning an LLM to generate faithful reasoning paths, we instead prompt the LLM with the graph schema to avoid instruction tuning. For recent methods that require training of the retriever, such as G-retriever [20] and SubgraphRAG [21], we do not include them in this experiment since they can not be applied in general GraphRAG workloads where questions do not have ground-truth labels. We also include the baseline, named Cypher, which directly generates cypher queries using the LLM without the guidance of our proposed question pattern taxonomy and any further optimizations built upon it to show the necessity of our adaptive prompting. For PolyG, we set the k in top-k shortest paths to 10, which means that we only take the top-10 reasoning paths satisfying the optional constraints in the question. For consistency, we use the Claude-3.5-sonnet [35] as the underlying LLM service for PolyG and all baselines, both for categorization and generation.

Evaluation metric. Since most of the graph question patterns do not have a single definitive answer as discussed in section 3, we assess the generation quality using win rates determined by LLM judgments. We evaluate responses from four perspectives: *Comprehensiveness*, *Diversity*, *Empowerment*, and *Directness*, and then let the LLM determines the *Overall Winner* considering all criteria, following previous approaches [9, 13]. Full prompts for these criteria are provided in Appendix I. As each question is judged across multiple baselines and thus there could be multiple high-qualty responses, we prompt the LLM to allow multiple winners for each criterion, leading to total win rates exceeding 100%. In addition to win rates, we report *F1-score* and *Deepseek-R1* for $\langle s, p, * \rangle$ and $\langle s, p, * \rangle + \langle s, p, * \rangle$ questions, which we have automatically labeled ground-truth answers. We use the strong open-weight reasoning model Deepseek-R1 [39] for judgment. Meanwhile, we measure end-to-end response latency and token usage to evaluate the computational efficiency.

C Additional Experimental Results

C.1 More Details on Generation Quality Results

Decomposition over question patterns. Table 6 compares the decomposed generation quality of PolyG and baseline methods with Claude-3.5-sonnet across all five evaluation criteria (averaged) and two quantitative metrics on all three knowledge graphs. The results show that PolyG achieves the highest win rates in 20 out of 25 cases, and also produces the highest *F1-score* and *Hit* on the basic $\langle s, p, * \rangle$ questions and the nested $\langle s, p, * \rangle + \langle s, p, * \rangle$ questions. Notably, PolyG outperforms all baselines across all criteria on the $\langle s, p, o \rangle$ and nested question types. For the remaining three patterns, PolyG has slightly lower win rates under the "*Directness*" criterion. This is primarily because PolyG retrieves a more comprehensive context than the baselines, leading the LLM to generate longer responses that cover diverse aspects of the information. This observation is supported by the high

Table 6: Response generation quality on each basic question pattern and the nested question pattern.

Question Pattern	Criteria	MS_GraphRAG	RoG	Fast-graphrag	Graph-CoT	Cypher	PolyG
$\langle s, *, * \rangle$	Comprehensiveness	52.50%	32.50%	5.83%	35.83%	33.33%	**69.58%**
	Diversity	**43.33%**	27.08%	5.00%	40.42%	23.75%	42.08%
	Empowerment	41.67%	35.83%	6.25%	31.25%	27.50%	**48.75%**
	Directness	**54.17%**	19.58%	28.75%	32.50%	46.25%	48.75%
	Overall Winner	30.00%	23.75%	2.92%	26.25%	15.00%	**33.33%**
$\langle s, p, * \rangle$	Comprehensiveness	17.08%	67.50%	33.75%	25.83%	17.50%	**67.92%**
	Diversity	5.42%	65.42%	17.08%	5.00%	7.92%	**68.75%**
	Empowerment	13.75%	**59.58%**	27.50%	5.42%	18.33%	**59.58%**
	Directness	7.08%	29.58%	25.42%	**56.67%**	9.58%	25.00%
	Overall Winner	10.83%	**47.08%**	24.58%	18.33%	13.33%	45.00%
	F1-score	0.2224	0.7433	0.3659	0.4077	0.2182	**0.7522**
	Hit	0.4625	0.9167	0.7208	0.7083	0.4083	**0.9292**
$\langle s, *, o \rangle$	Comprehensiveness	32.08%	15.00%	7.50%	13.75%	6.25%	**81.67%**
	Diversity	23.75%	11.25%	4.58%	17.08%	2.08%	**80.00%**
	Empowerment	34.58%	13.33%	8.75%	13.75%	6.67%	**71.67%**
	Directness	**51.25%**	8.33%	37.92%	41.25%	10.83%	42.08%
	Overall Winner	25.83%	9.17%	5.42%	10.00%	3.75%	**59.58%**
$\langle s, p, o \rangle$	Comprehensiveness	7.92%	6.25%	26.25%	9.17%	5.00%	**79.58%**
	Diversity	6.25%	7.08%	19.58%	8.33%	4.58%	**69.17%**
	Empowerment	8.75%	4.58%	25.00%	6.67%	5.00%	**74.58%**
	Directness	4.58%	5.42%	35.42%	43.75%	4.58%	**72.92%**
	Overall Winner	4.17%	1.25%	20.42%	5.83%	3.75%	**70.42%**
Nested	Comprehensiveness	20.83%	22.50%	12.50%	12.50%	4.17%	**74.58%**
	Diversity	16.25%	20.83%	7.08%	8.33%	2.92%	**70.42%**
	Empowerment	23.33%	17.92%	11.67%	10.83%	5.83%	**69.58%**
	Directness	32.08%	15.83%	31.67%	37.92%	6.25%	**44.58%**
	Overall Winner	15.00%	15.00%	7.92%	8.75%	3.75%	**65.42%**
	F1-score	0.1573	0.4938	0.1884	0.3115	0.0022	**0.5332**
	Hit	0.3500	**0.7667**	0.3667	0.4167	0.0167	0.7333

Table 7: Response generation quality of PolyG and baseline methods on each knowledge graph.

Dataset	Criteria	MS_GraphRGA	RoG	Fast-graphrag	Graph-CoT	Cypher	PolyG
Academia	Comprehensiveness	29.00%	26.75%	12.00%	21.50%	10.75%	**79.75%**
	Diversity	22.50%	23.50%	7.75%	12.25%	5.00%	**74.00%**
	Empowerment	27.50%	22.50%	12.00%	12.50%	9.50%	**71.00%**
	Directness	31.00%	13.25%	29.75%	**51.25%**	14.25%	48.25%
	Overall	19.00%	17.00%	9.00%	14.75%	6.50%	**61.50%**
	F1-score	0.2132	0.7551	0.371	0.5445	0.1332	**0.7577**
	Hit	0.6000	**0.9600**	0.8400	0.8700	0.3600	0.9300
Literature	Comprehensiveness	26.25%	27.50%	21.00%	19.25%	13.75%	**73.50%**
	Diversity	16.75%	26.50%	10.25%	15.50%	8.00%	**64.75%**
	Empowerment	23.25%	25.50%	18.25%	15.00%	12.00%	**63.50%**
	Directness	30.25%	12.50%	34.50%	38.00%	17.00%	**48.75%**
	Overall	16.75%	18.25%	13.25%	14.75%	8.50%	**54.00%**
	F1-score	0.199	0.7558	0.3575	0.3422	0.2041	**0.7788**
	Hit	0.3800	0.7900	0.6500	0.5500	0.3000	**0.8700**
E-commerce	Comprehensiveness	23.00%	32.00%	18.50%	17.50%	15.25%	**70.75%**
	Diversity	17.75%	29.00%	14.00%	19.75%	11.75%	**59.50%**
	Empowerment	22.50%	30.75%	17.25%	13.25%	16.50%	**60.00%**
	Directness	28.25%	21.50%	31.25%	38.00%	15.25%	**43.00%**
	Overall	15.75%	22.50%	14.50%	12.00%	8.75%	**48.75%**
	F1-score	0.2159	0.5694	0.2627	0.2787	0.1877	**0.5887**
	Hit	0.3400	**0.9100**	0.4600	0.5300	0.3300	0.8700

win rates of PolyG in terms of "*Comprehensiveness*," suggesting a trade-off between "*Directness*" and "*Comprehensiveness*." Therefore, if users prioritize more comprehensive responses—or seek a method that performs robustly across various question patterns—PolyG is the preferable choice.

Decomposition over datasets. In Table 7, we report the detailed win rates of the methods with Claude-3.5-sonnet, averaged across all question patterns for each knowledge graph. The results

show that PolyG consistently outperforms the baselines on all three knowledge graphs, achieving the highest *Win Rate*, *F1-score*, and *Hit* in 18 out of 21 cases. Notably, the win rates of PolyG on the Academia graph are generally higher, as this graph features more clearly defined entity types (e.g., *author*, *paper*, and *venue*) and well-structured relations (e.g., *reference* and *cited by*), making it easier for PolyG to generate appropriate Cypher queries and retrieve the desired information.

C.2 Response Latency and Token Usage Comparison

Table 8: Response latency and token usage of the methods on each question pattern averaged across datasets. The results are in the form *time/tokens.* **Bold faces** denote the lowest response latency and token usage.

Question Pattern	MS_GraphRAG	RoG	Fast-graphrag	Graph-CoT	Cypher	PolyG
$\langle s, *, * \rangle$	**9.78**/6,712	14.38/**1,142**	39.17/53,263	46.30/39,381	15.89/4,975	11.19/8,210
$\langle s, p, * \rangle$	**4.85**/5,756	13.21/**1,556**	36.44/44,926	44.49/55,782	9.86/2,929	20.26/4,004
$\langle s, *, o \rangle$	**8.01**/14,557	16.84/**1,019**	35.99/48,928	51.93/64,534	15.16/2,700	15.18/3,154
$\langle s, p, o \rangle$	27.51/38,424	17.90/**1,819**	34.85/44,900	69.93/254,260	**12.96**/5,045	19.82/4,069
Nested	**9.93**/21,875	19.00/**1,550**	35.97/48,175	70.85/109,016	13.82/5,720	57.79/15,314

Table 9: Response latency and token usage of the methods on each question pattern and each knowledge graph. The results are in the form *time/tokens.* **Bold faces** denote the lowest response latency and token usage.

Dataset	Question Pattern	MS_GraphRAG	RoG	Fast-graphrag	Graph-CoT	Cypher	PolyG
Academia	$\langle s, *, * \rangle$	**11.22**/8,664	12.2/**1,028**	32.23/73,089	50.7/41,054	13.88/3,111	12.56/10,163
	$\langle s, p, * \rangle$	**4.87**/8,043	13.06/**1,328**	28.08/64,863	36.21/40,710	8.89/3,387	27.9/4,977
	$\langle s, *, o \rangle$	**7.10**/13,613	16.98/**935**	29.61/72,270	50.61/55,243	16.97/2,317	30.61/3,464
	$\langle s, p, o \rangle$	33.48/23,391	22.03/**1,759**	26.07/55,755	58.3/179,926	**10.31**/5,758	28.58/5,771
	Nested	**8.21**/13,903	19.27/**1,825**	30.44/73,834	60.42/74,518	15.41/11,684	56.47/19,911
Literature	$\langle s, *, * \rangle$	**9.47**/4,457	15.11/**1,244**	18.35/25,479	48.46/43,434	13.03/2,502	10.68/5,951
	$\langle s, p, * \rangle$	**4.26**/1,931	13.07/**1,781**	17.69/28,755	50.9/77,353	9.95/3,078	17.2/3,791
	$\langle s, *, o \rangle$	8.9/12,261	14.73/**1,099**	17.05/28,905	53.35/83,293	11.78/1,357	**6.55**/2,781
	$\langle s, p, o \rangle$	31.54/53,826	13.72/**1,547**	18.11/38,993	87.83/413,733	**8.78**/1,602	14.65/3,276
	Nested	**8.20**/17,395	19.93/**1,396**	17.65/30,094	71.9/104,617	11.14/1,575	69.82/14,714
E-commerce	$\langle s, *, * \rangle$	**8.66**/7,016	15.83/**1,153**	66.93/61,220	39.73/33,655	20.76/9,313	10.32/8,516
	$\langle s, p, * \rangle$	**5.42**/7,294	13.51/**1,558**	63.54/41,161	46.35/49,283	10.74/2,322	15.67/3,244
	$\langle s, *, o \rangle$	**8.04**/17,797	18.81/**1,024**	61.32/45,610	51.84/55,067	16.74/4,426	8.39/3,218
	$\langle s, p, o \rangle$	17.5/38,054	17.95/**2,150**	60.36/39,952	63.67/169,121	19.8/7,775	**16.24**/3,160
	Nested	**13.38**/34,328	17.81/**1,429**	59.82/40,598	80.24/147,912	14.91/3,902	47.08/11,316

Decomposition over question patterns. Table 8 presents the detailed response latency and token usage of PolyG and the baseline methods across each question pattern using Claude-3.5-sonnet, averaged over the three knowledge graphs. The results show that, in most cases, MS_GraphRAG achieves the lowest response latency, while RoG consumes the fewest tokens. However, this low execution complexity comes at the cost of generation quality. For instance, although MS_GraphRAG is fast, it only performs best in 3 out of 31 criteria for generation quality, as shown in Table 6. Similarly, RoG uses the least tokens but only wins in 2 out of 31 criteria. Since response quality is the primary factor that determines the usefulness of an answer, their low latency or token usage does not indicate stronger capability for general GraphRAG workloads.

Meanwhile, although PolyG does not always offer the best execution efficiency or lowest resource consumption, its response latency and token usage are comparable to the best-performing baselines in most cases (e.g., 11.19s vs. 9.78s in latency and 3,154 vs. 1,019 tokens). Moreover, across all question patterns, PolyG achieves up to 4× speedup in response latency (e.g., for $\langle s, *, * \rangle$) and reduces token usage by more than 90% (e.g., for $\langle s, p, o \rangle$) compared to the worst-performing baselines.

Decomposition over datasets. Table 9 further decomposes the results from Table 8 by breaking them down by each knowledge graph, showing the response latency and token usage of each method across different question patterns and datasets. In addition to the similar observations made in Table 8, we find that PolyG achieves the lowest response latency for $\langle s, *, o \rangle$ questions on the Literature graph and for $\langle s, p, o \rangle$ questions on the E-commerce graph. Furthermore, response latency and token usage are generally higher on the Academia graph, which can be attributed to its dense graph topology, especially in the *author-paper* and *paper-paper* relations.

D Related Work

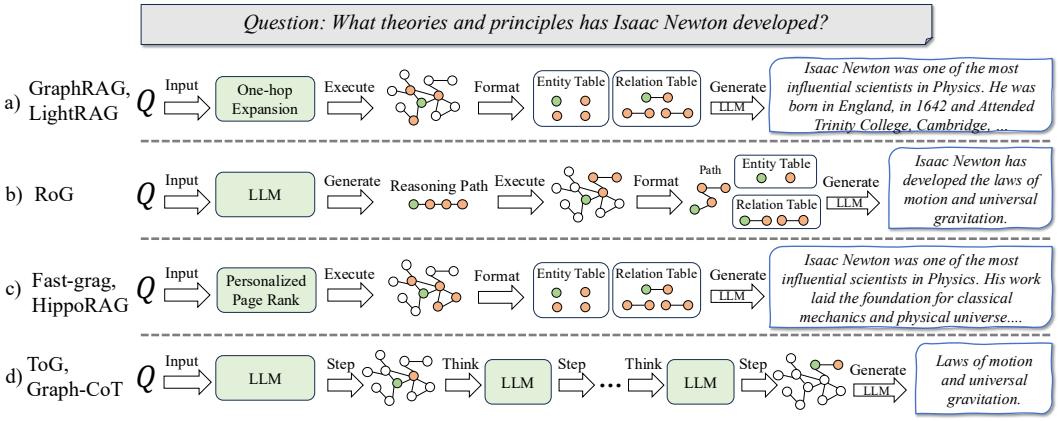


Figure 5: Workflow demonstration and comparison of existing GraphRAG methods.

Many GraphRAG approaches are proposed to tackle KGQA tasks or answer general user questions about the knowledge graph, and they differ in the adopted graph traversal methods. In the following, we introduce representative GraphRAG methods and Figure 5 illustrates the workflow of them.

As shown in Figure 5a, GraphRAG [9] and LightRAG [13] use one-hop neighbor expansion to retrieve neighboring entities and relations of the anchor entities (entities that appear in the question), and return two tables as the retrieved context to the LLM as external grounding knowledge. While one-hop neighbor expansion is efficient in execution, this method fails to capture long-distance knowledge and thus cannot well answer questions that require reasoning paths beyond one-hop.

RoG [14], as depicted in Figure 5b, prompts the LLM to generate faithful reasoning paths that can be followed on the knowledge graph and returns the retrieved paths for generation. Faithful reasoning paths directly point from the anchor entity to the answer entities without involving useless noise in the context. However, it is only applicable when the question itself explicitly reveals concrete relation constraints. If the question is abstract in its predicate chain, e.g., asking about general information of an entity or relationships with another entity, no faithful reasoning paths could be given.

The workflow of Fast-graphrag [15] and HippoRAG [16] is shown in Figure 5c. It retrieves entities and relations by running the well-studied random walk method, Personalized PageRank, and returns those with a relevance score greater than a threshold. Due to the diversity and complexity of graph questions, this fuzzing relevance matching that purely depends on the structure of knowledge graph might not accurately find what the question exactly requires.

The workflow of ToG [18] and Graph-CoT [19] is displayed in Figure 5d. They hand over the decision on every step of graph traversal to the LLM, prompting it with the current neighbors and all previous contexts to determine which neighbors to explore next. Traversal stops when the LLM thinks it reaches the answer, and then the answer entities are returned as results. This chain-of-thought manner is adaptable to various cases by leveraging the reasoning adaptability in LLMs. However, they end up at high cost in both response latency and token usage, which is expensive to use in real applications.

E Question Pattern Identification in Existing KGQA Benchmarks

E.1 Introduction of Existing KGQA Benchmarks

As listed in Table 1, there are seven major KGQA benchmarks developed over the past decade. Specifically, SimpleQ [23] and WebQSP [24] are two early KGQA benchmarks. SimpleQ constructs queries from Freebase facts by using the subjects and relationships as inputs and treating the objects as answers, which means it only contains questions of the $\langle s, p, * \rangle$ type in our taxonomy (discussed below). WebQSP is derived by selecting only those questions from WebQuestions [40] that can be translated into simple SPARQL queries; the original questions in WebQuestions were generated using the Google Suggest API [41]. To verify the question patterns in WebQSP, we prompt the

LLM to analyze the queries and validate that most of the questions in WebQSP also belong to the $\langle s, p, * \rangle$ pattern. CWQ [26], GraphQ [25], and GrailQA [27] are more recent benchmarks that feature higher query complexity. CWQ builds upon WebQSP by applying additional fact constraints (e.g., conjunctions and compositions) to the original SPARQL queries. GraphQ and GrailQA employ a graph-expansion algorithm to induce a subgraph from the question entity and then generate a question based on each subgraph. Although these datasets contain questions with multiple fact constraints and include features like counting, superlatives (e.g., argmax, argmin), and comparatives, the questions still ask about a concrete entity and can largely be categorized as combinations of $\langle s, p, * \rangle$ patterns.

E.2 Question Patterns in Existing KGQA Benchmarks

Table 10: Number of questions of each question pattern in WebQSP, CWQ and WIKIQA.

Benchmark	basic Question Pattern				Nested Question Pattern			
	$\langle s, *, * \rangle$	$\langle s, p, * \rangle$	$\langle s, *, o \rangle$	$\langle s, p, o \rangle$	$\langle s, *, * \rangle$ $\langle s, p, * \rangle$	$\langle s, p, * \rangle$ $\langle s, p, * \rangle$	$\langle s, *, o \rangle$ $\langle s, p, * \rangle$	$\langle s, p, o \rangle$ $\langle s, p, * \rangle$
WebQSP [24] (2016)	179	4,558						
CWQ [26] (2018)		19,631				15,058		
WIKIQA [29] (2015)	761	1,753	276		24	224		

For most benchmarks, the question pattern can be inferred from their question generation procedures—for example, from the templates used in RGBench [19] and CypherBench [28], or the subgraph construction methods in GraphQ [25] and GrailQA [27]. However, for WebQSP [24] and CWQ [26], we need to examine the questions directly to determine their corresponding patterns, as these questions are generated using the Google Suggest API. To this end, we classify each question in WebQSP and CWQ using an LLM (Claude-3.5-sonnet). The prompt used to guide the LLM in identifying the question pattern is largely the same as the one used in PolyG for question categorization, as shown in Figure 7. This prompt provides clear definitions of each question pattern, includes concrete few-shot examples, and allows for exceptions where questions do not fit cleanly into one of the four basic patterns and should instead be labeled as nested questions. Additionally, we ask the LLM to provide a question decomposition for each nested question. In the following, we discuss the question categorization results of WebQSP and CWQ, respectively.

Table 10 reports the results on WebQSP and CWQ. We observe that the majority of questions in the WebQSP benchmark fall into the $\langle s, p, * \rangle$ pattern, with only 179 out of 4,737 questions belonging to the $\langle s, *, * \rangle$ category. An example of a $\langle s, *, * \rangle$ question is "What is Mount St. Helens?". Although such questions typically lack definitive ground-truth answers, WebQSP annotates responses like "Stratovolcano" and "Volcano", which reduces the evaluation to checking for the presence of certain keywords rather than assessing the overall quality or helpfulness of the information. There are no $\langle s, *, o \rangle$ or $\langle s, p, o \rangle$ questions in WebQSP, as it is difficult to define a unique set of ground-truth answers for these types. In contrast, CWQ [26], which is sampled and augmented from WebQSP, introduces additional $\langle s, p, * \rangle + \langle s, p, * \rangle$ questions by incorporating extra fact constraints using conjunctions, superlatives, comparatives, and compositions. This results in 15,058 nested $\langle s, p, * \rangle + \langle s, p, * \rangle$ questions out of the total 34,689 questions.

To study the diversity and possible graph question patterns that may arise in real-world scenarios, we analyze WIKIQA [29], a text-understanding benchmark whose questions are extracted from real Bing query logs. We apply the same classification method used for WebQSP and CWQ to determine the pattern of each question in WIKIQA. As reported in Table 10, among the 3,047 questions, 761 fall under the $\langle s, *, * \rangle$ pattern, 1,753 under $\langle s, p, * \rangle$, and 276 under $\langle s, *, o \rangle$. Among the nested questions, 24 follow the $\langle s, *, * \rangle + \langle s, p, * \rangle$ pattern and 224 follow $\langle s, p, * \rangle + \langle s, p, * \rangle$. No $\langle s, p, o \rangle$ questions are found in WIKIQA, likely because the dataset only includes question-like queries that begin with WH-words (e.g., "what" or "how") and end with a question mark. In contrast, $\langle s, p, o \rangle$ questions are more likely to start with expressions such as "Is there," "Do," or "Have." Furthermore, since WIKIQA extracts only a subset of Bing query log data using basic word filtering, its question pattern distribution may not fully represent real-world scenarios. Nonetheless, the presence of multiple question patterns in WIKIQA supports the validity of our proposed taxonomy.

As we have established, most existing KGQA benchmarks primarily contain questions following the $\langle s, p, * \rangle$ pattern and its nested extension $\langle s, p, * \rangle + \langle s, p, * \rangle$. We now illustrate how representative

Cypher Query Template for <s,p,*> Question

```
MATCH (src1:[node_type] {attr: [attr1]})...-[relations]->(dst:[node_type])
MATCH (src2:[node_type] {attr: [attr2]})...-[relations]->(dst:[node_type])
...
WHERE dst.[attr1] [</>/=] [constraint1] AND dst.[attr2] [</>/=] [constraint2]...
WITH [DISTINCT] dst, operation(dst) AS addattr
WHERE addattr [</>/=] [addconstraint]
ORDER BY dst.[attr] AND/OR addattr
RETURN dst.[attr] or COUNT(dst)
```

Figure 6: Cypher template for $\langle s, p, * \rangle$ questions.

questions from these benchmarks exhibit strong similarities and limit the diversity of question coverage. In particular, we observe that they share similar Cypher query structures for retrieving the desired answers, which can be abstracted into a general template, as shown in Figure 6. Empirically, all $\langle s, p, * \rangle$ questions can be instantiated using this Cypher template, with the specific graph schema guiding how the relations in the graph correspond to the fact constraints expressed in the user question. Below, we present several examples to demonstrate how this template can be instantiated for various $\langle s, p, * \rangle$ questions.

❶ *"Which city held the summer Olympics twice?"*

Cypher query:

```
MATCH (src:[Event] {name: [Summer Olympics]})-[held in]->(dst:[City])
WITH DISTINCT dst, COUNT(dst) AS count
WHERE count == 2
RETURN dst.name
```

❷ *"What year did the Florida Marlins win their 2nd world series title?"*

Cypher query:

```
MATCH (src:[Title] {name: [World Series Chaimpion]})-[rel:won by]->(
    dst:[Team])
WHERE dst.name == Florida Marlins
WITH dst, src
ORDER BY src.date
RETURN dst.name
LIMIT 1
```

❸ *"Which states does the river that flows under the DeSoto Bridge pass through?"*

Cypher query:

```
MATCH (src:[Bridge] {name: [DeSoto Bridge]})-[on top of]->(n:[River])
    -[pass through]->(n:[City])-[belong to]->(dst:[States])
WITH DISTINCT dst
RETURN dst.name
```

❹ *"How many languages are used where L̈upang Hinirangïs the national anthem?"*

Cypher query:

```
MATCH (src:[Song] {name: [Lupang Hinirang]})-[national anthem]->(n:[
    Country])-[people]->(n:[People])-[speak]->(dst:[Language])
WITH DISTINCT dst
RETURN COUNT(dst)
```

Table 11: Statistics of the knowledge graphs used for evaluation

Attributes	Academia	Literature	E-commerce
# of Entities	2.7M	3.7M	8.1M
# of Relations	67M	28M	193M
# Templates	24	28	21
# of Questions	400	400	400

F Dataset Used in PolyBench

The statistics of the datasets used in PolyBench are presented in Table 11. The three knowledge graphs are sourced from RGBench [19]. Specifically, in the academic domain, there are three types of entities: *papers*, *authors*, and *venues*. These are naturally interconnected by citation-related relationships such as "*written-by*" and "*publish-in*". The selected academic graph is from the Physics domain of DBLP [32].

In the literature domain, the knowledge graph contains four types of entities—*books*, *authors*, *publishers*, and *series*. Its structure captures the publication relationships among these entities, extracted from the Goodreads dataset [33], which provides a rich collection of books along with detailed metadata. The relations include "*written-by*", "*publish-in*", "*book-series*", and others.

For the e-commerce domain, we adopt the graph from the Amazon product dataset [34], which offers metadata for items across a wide range of product categories. In this domain, the nodes include *items* and *brands*, and the interconnecting relations include "*also-viewed*", "*also-bought*", "*buy-after-viewing*", "*bought-together*" and "*item-brand*".

In the following, we list the concrete graph schema (i.e., entity and relation types) in the above three knowledge graphs.

F.1 Academia Domain

Node properties:

- type: author, properties: ["id", "name", "node_type"]
- type: paper, properties: ["id", "name", "label", "year", "node_type", "abstract"]
- type: venue, properties: ["id", "name", "node_type"]

Edge properties:

Author nodes are linked to their paper nodes by authorship.

- author $\rightarrow$ "paper" $\rightarrow$ paper

Paper nodes are linked to their author nodes, venue nodes, other paper nodes.

- paper $\rightarrow$ "author" $\rightarrow$ author
- paper $\rightarrow$ "reference" $\rightarrow$ paper
- paper $\rightarrow$ "venue" $\rightarrow$ venue
- paper $\rightarrow$ "cited_by" $\rightarrow$ paper

Venue nodes are linked to their included paper nodes.

- venue $\rightarrow$ "paper" $\rightarrow$ paper

F.2 Literature Domain

Node properties:

- type: book, properties: ["id", "name", "node_type", "description", "publication_year", "genres"]
- type: author, properties: ["id", "name", "node_type"]

- type: publisher, properties: ["id", "name", "node_type"]
- type: series, properties: ["id", "name", "node_type", "description"]

Edge properties:

Book nodes are linked to book nodes, author nodes, publisher nodes and series nodes.

- book → "author" → author
- book → "publisher" → publisher
- book → "series" → series
- book → "similar_books" → book

Author nodes are linked to their neighboring book nodes.

- author → "book" → book

Publisher nodes are linked to their neighboring book nodes.

- publisher → "book" → book

Series nodes are linked to their neighboring book nodes. 1. series → "book" → book

F.3 E-commerce Domain

Node properties:

- type: item, properties: ["id", "name", "node_type"]
- type: brand, properties: ["id", "name", "node_type"]

Edge properties: Item nodes are linked to neighboring item nodes and brand nodes.

- item → "also_viewed_item" → item
- item → "buy_after_viewing_item" → item
- item → "also_bought_item" → item
- item → "bought_together_item" → item
- item → "brand" → brand

Brand nodes are linked to their neighboring item nodes. Specific relations are:

- brand → "item" → item

G Question Templates for Question Generation

In the following, we list the question templates used in question generation for each knowledge graph.

G.1 Academia Domain

$\langle s, *, * \rangle$ **Question Template.**

- *Give me a broad introduction about "[author name]".*
- *Tell me some information about "[paper name]".*
- *Give me a comprehensive description about "[venue name]".*

$\langle s, p, * \rangle$ **Question Template.**

- *What paper have the author "[author name]" published?*
- *What are the academic collaborators of "[author name]"?*
- *What venues have the author "[author name]" published in?*
- *Who are the authors of the paper "[paper name]"?*

- *Where is the paper "[paper name]" published?*
- *Who are the academic collaborators of the author who writes the paper "[paper name]"?*
- *What venues have the author of the paper "[paper name]" published in?*
- *What venues have the academic collaborators of the author who writes the paper "[paper name]" published in?*

$\langle s, *, o \rangle$ **Question Template.**

- *How does "[author name]" and "[author name]" influence each other?*
- *How are "[paper name]" and "[paper name]" related to each other?*
- *What is the connection between "[author name]" and "[paper name]"?*
- *What is the relationship between "[venue name]" and "[paper name]"?*
- *how are "[venue name]" and "[author name]" connected?*

$\langle s, p, o \rangle$ **Question Template.**

- *Have the author "[author name]" cited or been cited by the work of the author "[author name]" and what are those works?*
- *Have authors "[author name]" and "[author name]" both collaborated with some other authors and who are they?*
- *Have the authors "[author name]" and "[author name]" ever published papers in the same venues? If so, tell me some examples.*
- *Do the venues "[venue name]" and "[venue name]" have the same authors publishing work in both of them and who are they?*

Nested Question Template.

- *Tell me about the academic contributions of the academic collaborators of the scholar "[author name]".*
- *Who are the academic collaborators of the author who writes both the paper "[paper name]" and paper "[paper name]"?*
- *What is the relationship between the scholar '' and the authors of the paper "[paper name]"?*
- *Have the scholars "[author name]" and "[author name]" both published work at the venue having the paper "[paper name]"?*

G.2 Literature Domain

$\langle s, *, * \rangle$ **Question Template.**

- *Tell me some information about "[author name]".*
- *Give me a broad introduction about "[book name]".*
- *Briefly describe "[publisher name]".*
- *Give me a comprehensive description about "[series name]".*

$\langle s, p, * \rangle$ **Question Template.**

- *Who are the authors of the book "[book name]"?*
- *What series have the author of the book "[book name]" published?*
- *What books has the author "[author name]" published?*
- *What books have the collaborators of the author "[author name]" published?*
- *What are the series published by the publishers that have published books of the author "[author name]"?*
- *What are the authors of the books published by the publisher "[publisher name]"?*
- *Where does the books of the series "[series name]" published in?*

- *What are the authors of the books that are published by the publishers that have published books of the series "[series name]"?*

$\langle s, *, o \rangle$ **Question Template.**

- *How do "[author name]" and "[author name]" influence each other?*
- *How is "[author name]" related to "[book name]"?*
- *How are "[book name]" and "[book name]" connected?*
- *What is the connection between "[book name]" and "[publisher name]"?*
- *What is the relationship between "[book name]" and "[series name]"?*
- *How are "[publisher name]" and "[publisher name]" related to each other?*
- *What is the relationship between "[series name]" and "[series name]"?*
- *What are "[series name]" and "[publisher name]" related?*

$\langle s, p, o \rangle$ **Question Template.**

- *Have the authors "[author name]" and "[author name]" ever published books in the same publishers? If so, tell me some examples.*
- *Do the publishers "[publisher name]" and "[publisher name]" have any authors publishing books in both of them and what are the publications and authors?*
- *Do the publishers "[publisher name]" and "[publisher name]" have books that belong to the same series, and if so, what are those books?*
- *Do the series "[series name]" and "[series name]" contain books that are published by the same publisher? If so, tell me about them.*

Nested Question Template.

- *Provide a comprehensive overview about the series whose books are similar to the publications of author "[author name]"?*
- *What books are published by the collaborators of both the author "[author name]" and "[author name]"?*
- *What is the relationship between the author "[author name]" and the author who has the book "[book name]"?*
- *Have the authors "[author name]" and "[author name]" ever published books in the same publisher that has the series "[series name]"?*

G.3 E-commerce Domain

$\langle s, *, * \rangle$ **Question Template.**

- *Tell me about "[brand name]".*
- *Tell me about "[item name]".*

$\langle s, p, * \rangle$ **Question Template.**

- *What is the brand of the item "[item name]"?*
- *What are the brands of the items that are also bought after viewing the item "[item name]"?*
- *What are the items that are also viewed when viewing items of the brand owning the item "[item name]"?*
- *What are the brands of the items that are also bought with items of the brand owning the item "[item name]"?*
- *What are the items of the brand "[brand name]"?*
- *What are the items that are bought together with items of the brand "[brand name]"?*
- *What are the brands of the items that are also bought with items of the brand "[brand name]"?*

- *What items does the brands of the items that are also viewed together with items of the brand "[brand name]" have?*

$\langle s, *, o \rangle$ **Question Template.**

- *What is the relationship between "[brand name]" and "[brand name]"?*
- *What are "[item name]" and "[item name]" related to each other?*
- *What are "[brand name]" and "[item name]" connected?*

$\langle s, p, o \rangle$ **Question Template.**

- *Have the items of the brands "[brand name]" and "[brand name]" ever both been also bought with some other items, and if so, what are those items?*
- *Have the items of the brands "[brand name]" and "[brand name]" ever both been bought after viewing some other items, and if so, what are those items?*
- *Have the items of the brands "[brand name]" and "[brand name]" ever been viewed together with some other items, and if so, what are those items?*
- *Have the items of the brands "[brand name]" and "[brand name]" ever been bought together with some other items, and if so, what are those items?*

Nested Question Template.

- *Give a broad introduction about the brands that are bought together with items of the brand "[brand name]"?*
- *What are the brands that are commonly viewed with the brands "[brand name]" and also bought together with the brand "[brand name]"?*
- *What is the relationship between the brand "[brand name]" and the brand which has the item "[item name]"?*
- *Have the items of the brands "[brand name]" and "[brand name]" ever been also viewed with the items that are bought together with the item "[item name]"?*

H Prompts Used in PolyG.

H.1 Question Categorization Prompt

In Figure 7, we provide the prompt we use for the LLM to judge which pattern the input question belongs to, including clear definitions of each question pattern and the corresponding few-shot examples. For instance, the definition for the $\langle s, p, * \rangle$ type is "In this type, the object is missing in the facts and question focuses on finding the object, with specific and concrete relations and attributes to the subject are provided by a concrete predicate chain p." and we give three examples "Who are the authors of the book 'Sunshine for the Latter-Day Sa'?", "What series have the author of the book 'Cookies for the Dragon (Saint Lakes, #2.1)' published?" and "What are the 5 biggest cities in the usa?" to the LLM to better understand how the questions in this pattern looks like.

H.2 Question Decomposition Prompt

In Figure 7, we provide the prompt we use for the LLM to decompose nested questions that can not directly fall into one of the four basic question patterns. We first provide a clear definition to each basic question pattern as in Figure 7 with few-shot examples, and then provide the instructions for decomposition. We also provide concrete example on how to decompose nested questions in the prompt. For instance, the decomposition of the question "Tell me about the academic contributions of the academic collaborators of the scholar 'L. Foldy'." is: 1) <s,p,*> question, find the academic collaborators of the scholar 'L. Foldy', 2) <s,*,*>: retrieve the academic contributions of the scholars found in the previous step. Finally, we ask the LLM to generate a detailed query plan based on given user question. We also provide the LLM with the concrete graph schema for it to generate accurate plans.

Question Categorization Prompt

Prompt:
You are an intelligent assistant tasked with classifying questions into different types based on the missing parts of a fact and the nature of the aspect being asked.

We have four types of fundamental questions, where s is the subject, p is the predicate, o is the object, and * means the missing part. Below, we define the four types of questions with some concrete examples for each type.

The four types of questions are:
- **<s,*,*> (0):**
In this type, the object and concrete predicate are both missing in the facts and question asks about the general or broad information of the subject (e.g., themes, concepts, or a general description).
Examples:
  - "Tell me about 'The Woman in Black: A Ghost Play'."
  - "Give a broad description about 'Frankenstein, or The Modern Prometheus'."
  - "Who is 'Barack Obama'?"
- **<s,p,*> (1):**
In this type, the object is missing in the facts and question focuses on finding the object, with specific and concrete relations and attributes to the subject are provided by a concrete predicate p.
Note that this type of question can have multiple predicates and subjects, and containing time or counting constraints in the predicates.
Examples:
  - "Who are the authors of the book 'Sunshine for the Latter-Day Sa'?"
  - "What series have the author of the book 'Cookies for the Dragon (Saint Lakes, #2.1)' published?"
  - "What are the 5 biggest cities in the usa?"
- **<s,*,o> (2):**
In this type, both the subject and object are given but the predicate is missing. Question asks about their relationships or interactions (e.g., conceptual connections, influences).
Examples:
  - "What is the relationship between 'Kizuna' and 'Kazuma Kodaka'?"
  - "What are the impacts of information communication technology to public relation practices?"
  - "How are the directions of the velocity and force vectors related in a circular motion?"
- **<s,p,o> (3):**
In this question, the subject, predicates object are all given. Questions either inquire about particular aspects of the relationship between the entities or verifies whether a specific relationship exists.
Examples:
  - "Have the authors 'Rubem Fonseca' and 'Lygia Fagundes Telles' ever published books in the same publishers?"
  - "Do the publishers 'Scholastic Inc.' and 'Klutz' have any authors publishing books in both of them?"

- **Nested Question (-1):**
The question may not belong to any of the above 4 types and you should return **-1**. These questions are nested and can be decomposed into several basic types. For example, when the question asks about the relationship between two entities or inquiry about general information about some entities, but the entities need to be determined by another query embeded in the overall question, it is a nested question and should be classified as -1.

Examples:
  - "Provide some concrete information about the academic collaborators of the scholar 'L. Foldy'.": 1. the first step is a <s,p,*> question, finding the collaborators of 'L. Foldy'. 2. the second step is a <s,*,*> question: gather general and broad information for the collaborators.
  - "Have the scholars 'S. Chiku' and 'A. I. Sanda' both published work at the venue having the paper 'weyl groups in ads3 cft2'?": 1. the first step is a <s,p,o> question, finding the venue having the paper 'weyl groups in ads3 cft2'. 2. the second step is a <s,p,o> question, finding the path validating whether the authors have both published work in the venue found in the previous step.
  - "In what way is the scholar 'Liang Wu' linked to the writers of 'bound states of breathing airy gaussian beams in nonlocal nonlinear medium'?" 1. the first step is a <s,p,*> question: find the authors of the paper 'bound states of breathing airy gaussian beams in nonlocal nonlinear medium'. 2. then a <s,*,o> question: find the path validating whether 'Liang Wu' is linked to the authors found in the previous step.

Instruction:
When given a question, analyze it based on the definitions above. Then, return **only a single number** corresponding to the type of the question:

- ****0**** for <s,*,*>
- ****1**** for <s,p,*>
- ****2**** for <s,*,o>
- ****3**** for <s,p,o>
- ****-1**** for nested question

Question: {}

Your Answer:

Figure 7: The prompt for the LLM to categorize questions into one of the four basic question patterns or nested questions.

H.3 Cypher Query Generation Prompt

In Figure 9 and Figure 10, we provide the complete prompt for the LLM to generate reliable cypher queries that can directly execute in the graph database to retrieve relevant information. In particular, we first give clear specifications on the generation task and setting, with a concrete example showing the complete input and desired output. Then, we introduce the goal, which is to generate a trustful cypher query, and corresponding graph schema for reference. Lastly, we instruct the LLM with common tips to write correct and efficient cypher queries, and also specify the output format. Especially, for $\langle s, p, o \rangle$ queries, we prompt the LLM to start with short paths (i.e., lower-order relations) if there are ambiguity for matching between fact constraints in the question and the graph schema. Moreover, for $\langle s, *, * \rangle$ and $\langle s, *, o \rangle$, we directly populate the cypher query templates as mentioned in section 5 using the question entities and avoid consuming LLM tokens.

Question Decomposition Prompt

Prompt:
You are an intelligent assistant tasked with decomposing nested questions into a plan of several sub-questions using the following question taxonomy.

We have four types of unit questions, where s is the subject, p is the predicate, o is the object, and * means the missing part. Below, we define the four types of questions with some concrete examples for each type.

The four types of questions are:
- **<s,*,*> (0):**
In this type, the object and concrete predicate are both missing in the facts and question asks about the general or broad information of the subject (e.g., themes, concepts, or a general description).
Examples:
  - "Tell me about 'The Woman in Black: A Ghost Play'."
  - "Give a broad description about 'Frankenstein, or The Modern Prometheus'."
  - "Who is 'Barack Obama'?"
- **<s,p,*> (1):**
In this type, the object is missing in the facts and question focuses on finding the object, with specific and concrete relations and attributes to the subject are provided by a concrete predicate p.
Note that this type of question can have multiple predicates and subjects, and containing time or counting constraints in the predicates.
Examples:
  - "Who are the authors of the book 'Sunshine for the Latter-Day Sa'?"
  - "What series have the author of the book 'Cookies for the Dragon (Saint Lakes, #2.1)' published?"
  - "What are the 5 biggest cities in the usa?"
- **<s,*,o> (2):**
In this type, both the subject and object are given but the predicate is missing. Question asks about their relationships or interactions (e.g., conceptual connections, influences).
Examples:
  - "What is the relationship between 'Kizuna' and 'Kazuma Kodaka'?"
  - "What are the impacts of information communication technology to public relation practices?"
  - "How are the directions of the velocity and force vectors related in a circular motion?"
- <s,p,o> (3):
In this question, the subject, predicates object are all given. Questions either inquires about particular aspects of the relationship between the entities or verifies whether a specific relationship exists.
Examples:
- "Have the authors 'Rubem Fonseca' and 'Lygia Fagundes Telles' ever published books in the same publishers?"
- "Do the publishers 'Scholastic Inc.' and 'Klutz' have any authors publishing books in both of them?"

Instruction:
When given a nested question that can not be directly classified into one of the four types, decompose it into a plan of several sub-questions with each being a unit question based on the definitions above.

We will also provide you the graph schema, which indicates the types of nodes and edges, and by what relations nodes are connected. This will help you to identify the sub-questions and generate a description for each step of the decomposition plan.

For each step of the decomposition plan, you should first identify the type of the sub-question and then generate a description which will further guide the instantiation of the concrete sub-question.

For example:
Question: "Tell me about the academic contributions of the academic collaborators of the scholar 'L. Foldy'."
Decomposition:
1. <s,p,*>: find the academic collaborators of the scholar 'L. Foldy'.
2. <s,*,*>: retrieve the academic contributions of the scholars found in the previous step.

Important Note: Always avoid consecutive <s,p,*> steps, they can be merged into one single <s,p,*> question with a chain of relations (multi-hop), and you should merge them into one step, just like the examples shown in demonstration of the <s,p,*> question ("Who are the authors of the books that are published by the publishers that have published books of the series 'Shifter Justice'?").

Graph Schema:
{graph_schema}

Question: {query}

Return your answer in the following format:
```plan
1. question-type: desciption 1
2. question-type: desciption 2
```

Figure 8: The prompt for the LLM to decompose nested questions into serveral of the four basic question patterns and output the detailed query plan.

I Criteria Used in Evaluation

The detailed explanation (prompt) of the evaluation criteria used in the experiments are as follows.

Comprehensiveness. How much detail does the answer provide to cover all aspects and details of the question? A comprehensive answer should be thorough and complete, without being redundant or irrelevant. For example, if the question is 'What are the benefits and drawbacks of nuclear energy?', a comprehensive answer would provide both the positive and negative aspects of nuclear energy, such as its efficiency, environmental impact, safety, cost, etc. A comprehensive answer should not leave out any important points or provide irrelevant information. For example, an incomplete answer would only provide the benefits of nuclear energy without describing the drawbacks, or a redundant answer would repeat the same information multiple times.

Diversity. How varied and rich is the answer in providing different perspectives and insights on the question? A diverse answer should be multi-faceted and multi-dimensional, offering different viewpoints and angles on the question. For example, if the question is 'What are the causes and

Cypher Query Generation Prompt for <s,p,*>

---Role---

You are a helpful assistant that can generate trustful reasoning paths with the knowledge of the graph schema to answer the given user question.

---Setting---

You will be provided with the knowledge graph schema, which indicates the types of nodes and edges, and by what relations nodes are connected. User questions ask about nodes which involve multi-hop relation paths.

---Goal---

Generate a trustful cypher query that can be executed on graph to get the answer using the given user question, based on the given schema of the knowledge graph.

If you don't have adequate information to give trustful reasoning paths, just say so. Do not make anything up.

---Graph Schema---

{graph_schema}

---Notes---

1. Please carefully think about the query structure and make sure the query is **correct** and **efficient** to execute. Do not forget to assign a variable name before retrieving the attributes.

2. Add the identification label in the generated cypher query as instructed in the graph schema section to ensure the cypher query inquires about the right graph.

3. Return the unique ids of retrieved entities and set the label of the result column as "id", using `RETURN DISTINCT node.id as id` in the cypher query.

4. Use "LIMIT 20" to limit the number of results returned in the cypher query.

Figure 9: The prompt for the LLM to generate cypher queries for $\langle s, p, * \rangle$ questions.

Cypher Query Generation Prompt for <s,p,o>

---Role---

You are a helpful assistant that can generate trustful reasoning paths with the knowledge of the graph schema to answer the given user question.

---Setting---

You will be provided with the knowledge graph schema, which indicates the types of nodes and edges, and by what relations nodes are connected. User questions are about relationships between nodes which can be indirect and result in multi-hop relation paths. User questions may include a relation constraint that indicates some specific paths.

An example:
User question is "What is the relationship between 'J. Koll' and 'Z. Staykova' in terms of paper reference?". In this case, the user is asking about the paths between two authors that is constraint to paper references and citations.
Suppose the "id" of 'J. Koll' and 'Z. Staykova' is 'id1' and 'id2', and the cypher query for this question is (where only paths that contains reference and citation relations should be considered):
```cypher
MATCH path = (author1:author {{id: 'id1'}})-[:paper]->(paper1:paper)-[:reference|cited_by]->(paper2:paper)-[:author]->(author2:author {{id: 'id2'}})
RETURN path
LIMIT 10
```

---Goal---

Generate a trustful reasoning path that can be executed on graph using the given user question, based on the given schema of the knowledge graph. Also give the cypher query that can be executed on the graph to get the answer.

If you don't have adequate information to give trustful reasoning paths, just say so. Do not make anything up.

---Graph Schema---

{graph_schema}

---Notes---

1. Add the identification label in the generated cypher query as instructed in the graph schema section to ensure the cypher query inquires about the right graph.

2. Please carefully think about the query structure and make sure the query is *correct* and *efficient* to execute. For example, correct cypher query should first "MATCH path" then "RETURN path".

3. Start from short-path cypher queries and do not use `*1..`, `*1..2`, `*1..3`, `*..` or even larger ranges for elation matching if we don't explicitly tell you to do so as it will take a long time. For example, just starting from "(:paper)-[:reference|cited_by]->(:paper)" for matching "paper reference" relations is good. If current simple queries can not find the answer, we will ask you to gradually extend the path with specfic path length.

4. Always use single "MATCH path =" clause for the whole cypher query, starting from one input entity to another and captures the whole path.

5. Return the results in path format:
```cypher
RETURN path
LIMIT 10
```

Figure 10: The prompt for the LLM to generate cypher quries for $\langle s, p, o \rangle$ questions.

effects of climate change?', a diverse answer would provide different causes and effects of climate change, such as greenhouse gas emissions, deforestation, natural disasters, biodiversity loss, etc. A diverse answer should also provide different sources and evidence to support the answer. For example, a single-source answer would only cite one source or evidence, or a biased answer would only provide one perspective or opinion.

Empowerment. How well does the answer help the reader understand and make informed judgements about the topic without being misled or making fallacious assumptions? Evaluate each answer on the quality of answer as it relates to clearly explaining and providing reasoning and sources behind the claims in the answer.

Overall Winner. How specifically and clearly does the answer address the question? A direct answer should provide a clear and concise answer to the question. For example, if the question is 'What is the capital of France?', a direct answer would be 'Paris'. A direct answer should not provide any irrelevant or unnecessary information that does not answer the question. For example, an indirect answer would be 'The capital of France is located on the river Seine'.