Scilight

Transactions on Graph Intelligence and Network Applications
https://www.sciltp.com/journals/tgina

Article

PoP-RAG: Holistic Query Planning for GraphRAG

Qiming Zeng [1], Xintong Hu [1], Yuhao Lin [1], Hao Luo [1], Susie Xi Rao [2], Xiao Yan [1] and Jiawei Jiang [1,*]

[1] School of Computer Science, Wuhan University, Wuhan 430072, China
[2] School of Engineering and Computer Science, Bern University of Applied Sciences, 3012 Bern, Switzerland
* Correspondence: jiawei.jiang@whu.edu.cn

How To Cite: Zeng, Q.; Hu, X.; Lin, Y.; et al. PoP-RAG: Holistic Query Planning for GraphRAG. *Transactions on Graph Intelligence and Network Applications* **2026**. https://doi.org/10.53941/tgina.2026.100006

Received: 18 May 2026
Revised: 6 September 2026
Accepted: 11 September 2026
Published: 22 September 2026

Abstract: Graph-based retrieval-augmented generation (GraphRAG) leverages knowledge graphs to provide context for large language models (LLMs) to generate quality responses. Yet existing GraphRAG methods suffer from two drawbacks: connecting each entity to all passages that mention the entity causes one-to-many entity-passage mapping problem and retrieves redundant passages, and fixed graph traversal patterns fail to locate target information for hard queries. To tackle the two limitations, we propose PoP-RAG, which features a passage-on-edge (PoE) graph that links the passages with graph edges to resolve the one-to-many entity-passage mapping problem. PoP-RAG further introduces a query planning step that decomposes a query into sub-queries and builds a directed acyclic graph (DAG) to model their relationships. This design enables targeted retrieval for each sub-query and ensures transparent response derivation logic, thus enhancing explainability. Experimental results demonstrate that PoP-RAG outperforms existing GraphRAG methods, especially on complex queries.

Keywords: LLMs; RAG; GraphRAG

1. Introduction

Large Language Models (LLMs) [1–3] have become increasingly useful in assisting people with various tasks [4,5] and answering a wide range of questions [6,7]. However, their performance is limited by both outdated training data [5,8,9] and hallucinations [10–12]. To tackle these problems, Retrieval-Augmented Generation (RAG) technology has come to the fore [13–15], enhancing LLMs by grounding their responses in external, authoritative knowledge sources [16,17]. GraphRAG enhances this architecture by introducing knowledge graphs as the underlying knowledge representation, enabling more precise and context-aware responses [18].

A standard GraphRAG framework consists of two key components [19]. The first is offline indexing, where GraphRAG constructs a knowledge graph [20,21] from certain text passages; this structure not only enables deeper semantic understanding by encoding contextual dependencies between concepts [22] but also enhances the capture of latent entity associations that are often obscured in unstructured text. The second is online querying, where the system retrieves information from the knowledge graph in response to user queries and generates accurate answers [23].

Recent developments in GraphRAG explore a variety of graph traversal strategies to support complex question answering scenarios. Many existing methods adopt a fixed graph traversal pattern, where the retrieval path is pre-defined and remains unchanged across different queries. MGRAG [24] first applies community detection algorithms to the knowledge graph, then generates summary representations for each identified community. During question answering, these community summaries are re-ranked and incorporated as supplementary contextual information. LightRAG [25] introduces a dual-level retrieval mechanism that first extracts entity-related and relationship-related keywords from the input query, then uses these keywords for direct knowledge graph retrieval, and finally synthesizes the retrieved nodes and edges and their source text into a coherent contextual representation. HippoRAG 2 [26] employs a knowledge graph that combines phrase-level and passage-level nodes, and enhances

retrieval via a personalized PageRank algorithm after the direct triple retrieval. Although these methods provide structured and efficient retrieval pipelines, their fixed traversal patterns lack adaptability to query-specific reasoning demands, often leading to the omission of essential entities in multi-hop QA—especially when those entities are not explicitly stated and must be inferred through intermediate reasoning steps.

Other methods adopt a more flexible, local graph traversal paradigm, where the retrieval path evolves step-by-step based on intermediate results. ToG [27] enables LLMs to iteratively explore and rank reasoning paths via beam search over knowledge graphs, supporting deeper and more interpretable retrieval. Similarly, Graph-CoT [28] proposes an iterative framework where LLMs dynamically determine which graph information to retrieve through repeated cycles of reasoning, and dynamically adjust the graph mining function accordingly. While these approaches improve adaptability, their step-by-step reasoning process remains inherently myopic, constrained by the limited horizon available at each reasoning stage. An early misstep in retrieval can easily cascade through subsequent iterations, progressively steering the reasoning away from the correct answer and ultimately compromising the final outcome.

In summary, existing GraphRAG methods have made notable progress in leveraging knowledge graphs for retrieval-augmented generation. Fixed traversal approaches offer structured and efficient retrieval pipelines, while local traversal methods introduce greater adaptability by dynamically adjusting reasoning paths based on intermediate results. However, existing methods exhibit two limitations in the indexing and retrieval stages that may impede their accuracy.

- One-to-many entity-passage mapping. Adding passage nodes to knowledge graphs is a way to make GraphRAG more accurate. Current implementations extract entities from source passages and link these entities with all the passages mentioning them. During graph-based retrieval, when an entity node is selected, this "entity-passages" mapping may result in the retrieval of many irrelevant passages, thereby causing the one-to-many entity-passage mapping problem. Take the query "Tell me about the history of artificial intelligence development" as an example, the "artificial intelligence" entity node may be linked to multiple passages covering diverse aspects such as historical evolution, application prospects, and research trends. However, only those passages focusing on historical evolution are relevant to the specific query, while the others introduce irrelevant information that could interfere with the retrieval accuracy.
- Fixed or local graph traversal pattern. Current GraphRAG methods employ diverse graph traversal strategies during retrieval. Many existing GraphRAG methods use a fixed graph traversal approach with unchanged, query-inadaptive strategies. Examples include Microsoft's GraphRAG (MGRAG) [24], which utilizes community summarization for question answering; LightRAG [25], which retrieves information via query-extracted keywords; and HippoRAG 2 [26], which applies personalized PageRank to triple retrieval. Take the question "When was the person who Messi's goals in Copa del Rey compared to get signed by Barcelona?" as an example. Since one of the key entities *"Diego Maradona"* (referring to the person in the query) is not mentioned directly in the query, the retrieval process may lead to other entities such as *Copa del Rey, Messi, or Barcelona*. ToG [27], by contrast, employs a local graph traversal pattern, where each step uses the currently retrieved information to traverse the graph and select the next exploration node. The vision of each step of reasoning is limited by present acquired knowledge. Once a deviation or error occurs in a certain step of reasoning, subsequent retrieval is prone to being misled into an incorrect path, causing the entire reasoning process to deviate from the correct direction. Figure 1 illustrates such a situation. For HippoRAG 2 and ToG, the proportion of triples retrieved during the retrieval process that can directly answer the question is not high, and even less than 30% on the most difficult MuSiQue dataset. The imprecise retrieval strategies used by these methods pose a challenge to the accuracy of GraphRAG in multi-hop QA.

To tackle the two limitations, we propose PoP-RAG, which adopts the passage-on-edge (PoE) graph for index construction and directed acyclic graph (DAG)-based query planning for context retrieval. As shown in Figure 1, the two innovations enable PoP-RAG to retrieve contents that are more related to user queries and thus generate more accurate responses. In particular, during index construction, PoP-RAG first extracts entities as graph nodes and relations as graph edges from the passages like existing GraphRAG methods. The PoE graph associates each passage with only the edges that correspond to the relations described by the passage. Consequently, when an edge is retrieved, only its associated source passage or passages are returned. This resolves the one-to-many entity-passage mapping in existing methods and allows the system to retrieve the related passages more accurately.

Before context retrieval, PoP-RAG introduces a query planning step that decomposes a complex user query into simpler sub-queries and uses a DAG to model their relationships. Figure 2 provides a running example for the query planning. Each sub-query is simpler and more specific than the original query, while the DAG provides a global view of the required reasoning process, thereby addressing the limited horizon of step-by-step local graph

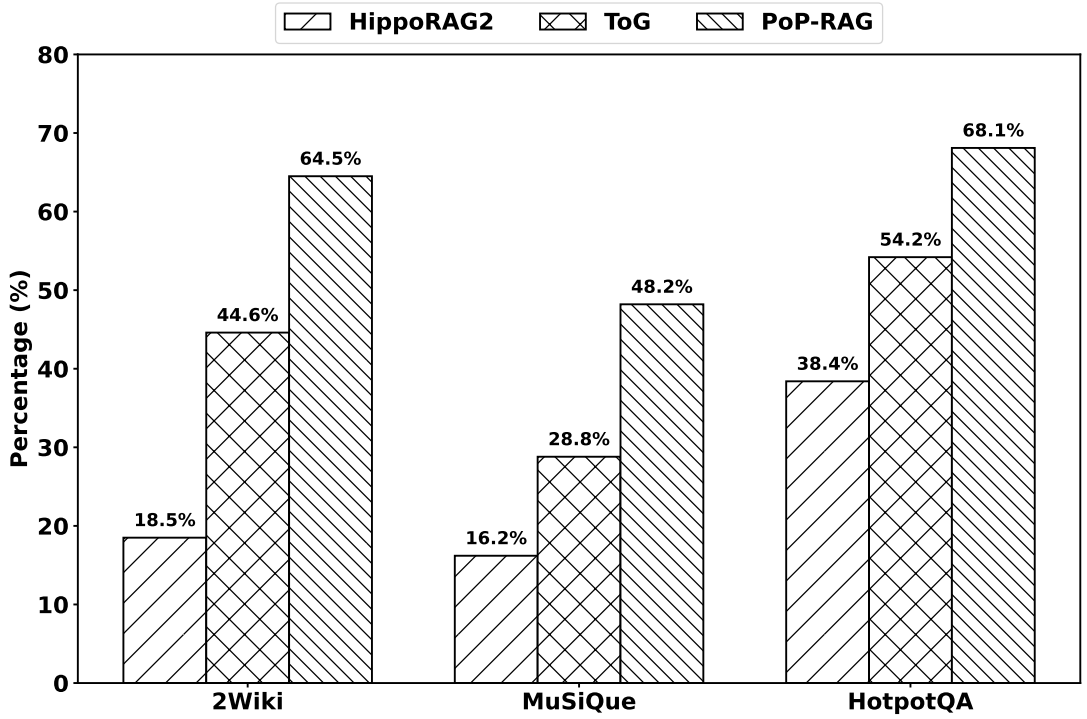

Figure 1. Validity rates of retrieved triples for HippoRAG 2, ToG, and PoP-RAG across 3 datasets. The validity rate is the percentage of questions that can be answered directly using the retrieved triples.

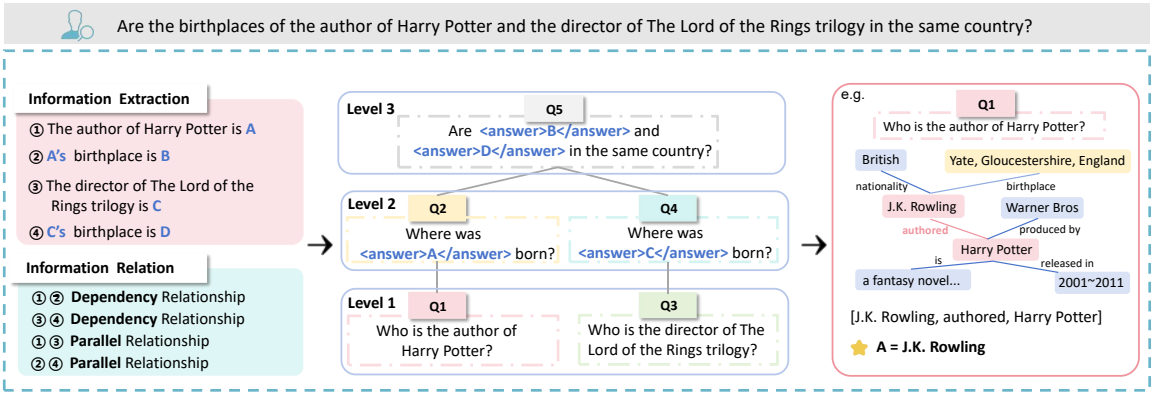

Figure 2. An example of the query planning process of our PoP-RAG.

traversal (e.g., ToG). Moreover, the DAG also shows the retrieved facts and how the final response is derived from these facts. This is particularly useful when users want to explain or verify the response (e.g., when the passages may not be reliable). PoP-RAG executes the DAG-shaped query plan level by level and uses the answers returned by lower-level sub-queries to update dependent higher-level sub-queries. Additional retrieval can be conducted at higher levels with the information provided by the lower-layer sub-queries, which resolves the missing entity problem in our previous example. In particular, after the lower-layer sub-query gets "*Diego Maradona*", retrieval can be conducted to decide "when Diego Maradona was signed by Barcelona".

To evaluate PoP-RAG, we conduct experiments on 3 popular question-answering (QA) datasets and compare PoP-RAG with 5 state-of-the-art (SOTA) baselines. The results show that PoP-RAG consistently achieves higher response accuracy than all baselines across the datasets. Compared with the best-performing baseline, the average improvements in the F1 and exact-match (EM) scores are 4.7% and 4.9%, respectively. The ablation study shows that the two key designs of PoP-RAG, i.e., the PoE graph and DAG-based query planning, are both effective in improving response accuracy. Micro experiments show that PoP-RAG indeed retrieves triples that are more related to user queries (cf. Figure 1). We also observe that PoP-RAG provides larger gains over the baselines for more complex queries while keeping online token consumption at a reasonable level.

To summarize, we make the following key contributions.

- We propose the passage-on-edge (PoE) graph, which places passages on graph edges to enable precise passage retrieval and overcome the problem of one-to-many entity-passage mapping.

- We propose a query planning step for GraphRAG, which decomposes a complex user query into sub-queries and uses a DAG to model the relation between sub-queries. Such planning makes it easy to accurately retrieve the information for each specific sub-query and understand how the response is generated.
- We implement the two innovations in PoP-RAG and comprehensively evaluate it by comparing with SOTA baselines.

2. Materials and Methods

Our proposed PoP-RAG enhances GraphRAG through two components: the passage-on-edge (PoE) graph for indexing and DAG-Based Query Planning (DAG-BQP) for retrieval. We will elaborate on each component in subsequent sections.

2.1. Passage-on-Edge Graph

In the indexing phase, existing mainstream GraphRAG methods with passage nodes adopt a passage-on-node paradigm, linking passages directly to entity nodes. This paradigm's scalability limitations become evident as the dataset grows: the expansion of document volume leads to a proliferation of one-to-many mappings between entities and passages. As demonstrated in Table 1, current methods such as LightRAG and HippoRAG 2 exhibit this problem. For instance, on the MuSiQue dataset, LightRAG has 10,593 entity nodes linked to multiple passages, while HippoRAG 2 exhibits 12,586 such nodes, with maximum out-degree values reaching 716 and 972, respectively.

This data structure recalls multiple passage nodes when retrieving target entity nodes; redundant passages raise screening costs or reduce accuracy via noise interference, undermining overall performance. To address this dilemma and enhance the precision of retrieval after identifying certain triples, we propose the *PoE graph* structure.

During the indexing phase, PoP-RAG utilizes the PoE graph framework to construct a knowledge graph index based on edge-associated textual passages. The construction process involves Entity Extraction, *Triple Extraction* and *PoE Graph Construction*.

In the Entity Extraction phase, we use an NER model [29]:

$$\{e\} = f_{\text{NER}}(t) \tag{1}$$

where f_{NER} denotes the aforementioned NER model, and $\{e\}$ represents the entity set extracted from input text t. Specifically, the NER model receives a passage string and returns a JSON object in the form `{"named_entities": [e1, e2, ...]}`.

In the Triple Extraction phase, an Open Information Extraction (OpenIE) model [30] is utilized:

$$\{\text{triples}\} = f_{\text{OpenIE}}(t, \{e\}) \tag{2}$$

where f_{OpenIE} denotes the OpenIE model and $\{\text{triples}\}$ denotes the set of triples extracted from t using the entity set $\{e\}$. The model receives the passage and its extracted entity list, and returns `{"triples": [[h, r, o], ...]}`, where h, r, and o denote the head entity, relation, and tail entity, respectively.

Extracted triples are mapped to graph components (entities as nodes, interrelationships as edges); original text passages are added as passage nodes and anchored to corresponding relation edges for contextual integrity. Each relation edge retains an association with the identifier of its source passage. Once an edge is selected during retrieval, this association is used to obtain its corresponding source passage or passages.

The PoE graph indexing architecture uses directional mappings to link each edge to its source passage node; no semantic duplicates in test datasets yield one-to-one edge-passage correspondences, with multi-passage-to-single-edge associations remaining compatible with the PoE graph design. This semi-injective relationship overcomes conventional constraints. Figure 3 shows that traditional passage-on-node strategies suffer from one-to-many entity-passage mapping (e.g., the red entity node links to four passage nodes), while our PoE graph enforces semi-injective mapping with unique edge-passage node associations. This design enables accurate identification and tracing of triples' original textual sources even with exponential document scale growth, allowing direct retrieval of triples-relevant passages in later phases.

PoP-RAG supports incremental indexing through content-based hash identifiers. When new passages are added, their identifiers are compared with those in the existing index. Existing passages directly reuse their stored OpenIE results and embeddings, while only previously unseen passages undergo NER, OpenIE, and passage embedding. Newly extracted entities and relation facets are also deduplicated by their hash identifiers before being appended to the embedding stores and the PoE graph, together with the associations between the new relation edges and their source passages. Consequently, incremental indexing processes only newly added content rather than the complete corpus.

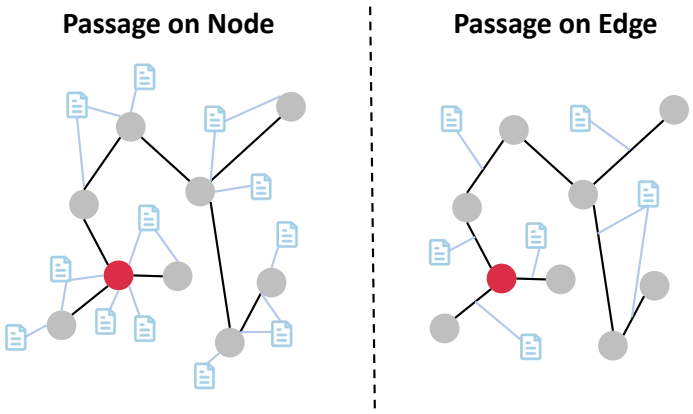

Figure 3. Comparison of entity-passage mapping approaches: (**Left**) Conventional passage-on-node, where a single entity (red) may be connected to multiple passage nodes (blue); (**Right**) Proposed PoE graph resolving the ambiguity by only connecting edges to their source passage.

Table 1. Passage node condition of current GraphRAG methods with passage nodes, PN refers to passage node; PN > 1 indicates the count of entity nodes linked to multiple passage nodes; PN MAX represents the maximum number of passage nodes connected to a single entity node.

Dataset	PN Condition	LightRAG	HippoRAG 2
2Wiki	PN > 1	5745	6683
	PN MAX	244	171
MuSiQue	PN > 1	10,593	12,586
	PN MAX	716	972
HotpotQA	PN > 1	9874	11,574
	PN MAX	303	610

2.2. DAG-Based Query Planning

In the query phase, we observe that existing methods adopt either fixed or local graph traversal patterns, both of which have drawbacks. Fixed graph traversal risks missing implicit key entities in retrieval, while local traversal suffers from error propagation due to limited step-wise reasoning scope. To address the drawbacks of existing works and enhance the accuracy of question answering, we propose the DAG-based query planning (DAG-BQP) method. As illustrated in Figure 4, our DAG-BQP method mainly consists of four components: Information Extraction, Sub-query DAG Construction, Hierarchical Sub-query DAG Execution, and Dual-Augmented Answer Generation. The following sections detail each component.

2.2.1. Information Extraction

Humans typically solve problems via a three-step cognitive process of information retrieval and integration [31,32]: retrieving relevant information, structurally integrating it, and generating an answer. Inspired by this cognitive mechanism, our DAG-BQP introduces an information-extraction step, which is defined as:

$$\langle \mathcal{I}(q), \mathcal{R}(q) \rangle = \Psi_{\text{LLM}}(q, p) \tag{3}$$

where q denotes the user's input query, p represents the task-specific prompt for guiding the extraction process, Ψ_{LLM} is the LLM-driven joint reasoning operator, $\mathcal{I}(q)$ stands for the set of core information units that can be used to answer q, and $\mathcal{R}(q)$ denotes the set of semantic relationships between these information units. It is worth emphasizing that these information units do not need to directly form a complete answer; for information entities that are not yet clear, referential annotation can be used for abstract representation. Each information unit is expressed as a concise natural-language statement that identifies the known entity or contextual constraint, the relation or attribute to be retrieved, and, when necessary, a referential placeholder such as A for the unknown entity. The extracted units and their pairwise relationships are returned separately as information points and information relations.

To construct the question architecture, we further explored the relationships between information units. By summarizing the possible relationships between information units, we found that these relationships can be mainly divided into two categories, namely serial relationships and parallel relationships.

- Serial Relationships. A serial relationship indicates sequential dependency between information units X and Y, where Y's validity and interpretation depend entirely on X. Without X, Y becomes unclear or logically

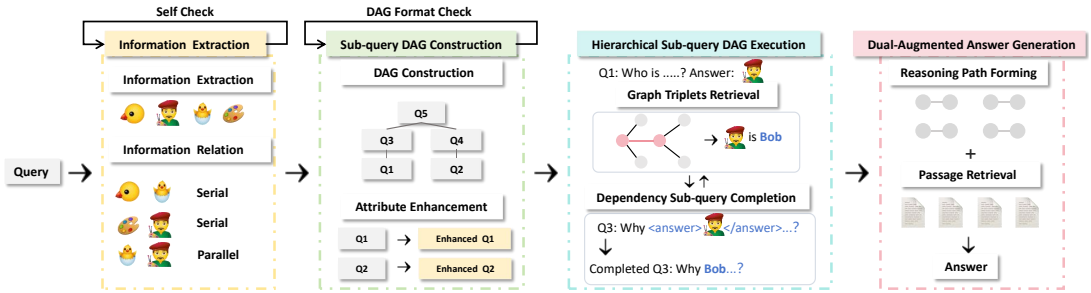

Figure 4. An overview of our DAG-Based Query Planning (DAG-BQP).

incomplete. For instance, in the facts "The author of Harry Potter is A" and "A's birthplace is B", "A" in the latter fact is defined by the former; without this temporal anchor, B loses its chronological reference. Serial relationships show how one piece of information leads to another, like steps in a process or events in a timeline.

- Parallel Relationships. A parallel relationship indicates no sequential dependency between two information units (X and Y). Here, Y's validity remains independent of X—its logic, expression, and integrity require no preconditions from X. Even without X, Y retains clarity and coherence. For instance, "The author of Harry Potter is A" and "The director of the Lord of the Rings trilogy is C" exemplify a parallel relationship. The authorship of Harry Potter remains independent of the directorship of The Lord of the Rings trilogy. Parallel relationships demonstrate how multiple pieces of information exist independently, like unrelated products in a store or simultaneous events in different locations.

The results generated through the extraction of information units and the exploration of relationships between information units can provide guidance for the subsequent construction of the sub-query DAG.

It should be noted that LLMs may produce errors. Therefore, we have further introduced a self-checking step. For the extracted set of information units, there are three possible types of errors: missing information, redundant information and ambiguous information.

- Missing Information. This error refers to the absence of key information in the extracted set of information units that is necessary to answer the user's question.
- Redundant Information. This error occurs when the extracted set of information units contains irrelevant information that does not contribute to answering the user's question.
- Ambiguous Information. This error arises when the extracted information units contain vague or contextually unclear elements that could lead to multiple interpretations.

We input the results generated in the previous steps along with the user's query into LLMs for self-checking to identify whether the above-mentioned errors exist; if errors are detected, LLMs will generate corresponding modification suggestions and regenerate the set of information units based on the suggestions.

2.2.2. Sub-Query DAG Construction

After completing information extraction, DAG-BQP enters the sub-query DAG construction phase. We first formally define a directed acyclic graph (DAG) as a fundamental structure:

$$\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle \tag{4}$$

where $\mathcal{G}$ denotes a DAG; $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ represents the set of nodes in the DAG; $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of directed edges between nodes, and no cyclic edges exist in $\mathcal{E}$.

To anchor the DAG to our query planning task, we define a bijective mapping between DAG nodes and sub-queries:

$$Q : \mathcal{V} \to \mathcal{S} \tag{5}$$

where Q is a bijective mapping function that maps each node $v_i \in \mathcal{V}$ to a unique sub-query $s_i \in \mathcal{S}$; $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ denotes the set of sub-queries. Each answer-bearing information unit normally produces one sub-query node. An information unit used as a contextual constraint may be merged into a related sub-query, and a final integration node may be added to combine the intermediate answers.

Based on the above, we now construct the sub-query DAG as:

$$\mathcal{G}_{\text{sub-query}} = \langle \mathcal{V}, \mathcal{E}, Q \rangle \tag{6}$$

where $\mathcal{G}_{\text{sub-query}}$ is the sub-query DAG our DAG-BQP generated; the edges $\mathcal{E}$ are derived from the relationships between information units: an edge $e_{ij} = (v_i, v_j) \in \mathcal{E}$ exists only if the corresponding information units u_i and u_j (targeted by $s_i = Q(v_i)$ and $s_j = Q(v_j)$) have a serial relationship. The direction from v_i to v_j indicates that v_i is a prerequisite of v_j, and the answer obtained from v_i is bound to the referential placeholder in v_j. Parallel nodes have no dependency edge between them and can be executed at the same DAG level.

Figure 2 illustrates sub-query DAG construction via a simple case: the DAG has three execution levels, where higher-level queries rely on preceding query answers and same-level queries are non-interfering; the tag "<answer></answer>" denotes referential answers waiting to be determined.

To enhance sub-query DAG quality, we perform a parsability-focused format check (verifying execution level consistency and answer tag formatting) with reconstruction for failed cases. Moreover, directly generated sub-queries for questions with lengthy attributive clauses tend to exhibit semantic ambiguity. For example, the query "Sparking the Marian civil war, who helped the recently abdicated queen to escape her imprisonment?" may generate the ambiguous sub-query "who is the recently abdicated queen?" due to the missing temporal anchor "Marian civil war". Thus, an LLM-based attributive enhancement step is added to the end of the pipeline to resolve sub-query vagueness in the sub-query DAG.

2.2.3. Hierarchical Sub-Query DAG Execution

With the sub-query DAG obtained, we execute graph retrieval on the constructed knowledge graph to collect question-relevant information. Leveraging the sub-query DAG's hierarchical structure, execution operates hierarchically: queries in the same hierarchy are parallel and non-interfering, while cross-level queries have a serial relationship, with subsequent ones depending on prior answers. Thus, after executing all questions in one execution level we move on to the next execution level, the answers in lower execution levels are organized to update dependent queries in the upper execution levels (e.g., in Figure 2, Level 1 sub-query execution updates *A* to "*J.K. Rowling*" and *C* to "*Peter Jackson*").

We encode each sub-query s_i into a dense vector representation:

$$s_{\text{emb}} = \text{Encoder}(s_i, \Theta_E) \tag{7}$$

and perform similarity retrieval against the knowledge graph to obtain the top-N most relevant triples using:

$$\text{Sim}(s_{\text{emb}}, t_k) = \cos(s_{\text{emb}}, t_k) \tag{8}$$

where t_k denotes the embedding of a single triple in our PoE graph. To accurately identify query-relevant triples, LLMs are employed to filter the N candidate triples against the user query; the retained triples are then used to answer the corresponding sub-queries. For each sub-query, all top-N candidate facets are provided together to the LLM in a single call. The LLM jointly assesses evidence sufficiency, selects the useful facets, and generates the sub-query answer when the selected evidence is sufficient. If no triples remain after filtering, a fallback mechanism is introduced: all source passages corresponding to the pre-filtered triples are used for answering, and the resulting answers are used to update subsequent sub-queries.

2.2.4. Dual-Augmented Answer Generation

During the execution of the hierarchical sub-query DAG, the triples corresponding to each sub-query are recorded as $Path(\mathcal{G}_{\text{sub-query}})$. Leveraging our PoE graph, DAG-BQP directly retrieves and separately stores source passages corresponding to the triples as *passages*. We formalize the answer generation process as:

$$FinalAns(q) = \Psi_{\text{LLM}}\left(q, Path(\mathcal{G}_{\text{sub-query}}), passages\right) \tag{9}$$

where q denotes the original user query, and Ψ_{LLM} denotes the few-shot LLM employed for answer generation.

The final LLM input jointly contains the original query, the selected relation facets along the DAG reasoning path, and their associated source passages. The answer-generation prompt starts with the following template:

```
Question: {original_query}
Reasoning-path facets: {selected_facets}
Evidence passages: {retrieved_passages}
Using the reasoning-path facets and evidence passages, answer the ques-
tion concisely.
```

Reasoning path triples serve as knowledge network links to clarify query-to-answer logical deduction; source passages supply contextual details (including detailed scenarios, contextual information, professional terms, and so on) to support the reasoning path and improve answer reasonableness. Unlike other verbose GraphRAG methods that only output answers with supporting materials, PoP-RAG can return the reasoning path and sub-query DAG to the user simultaneously, which enhances the interpretability of PoP-RAG.

2.2.5. Comparison to Prior Planning Methods

In GraphRAG, existing query strategies generally rely on fixed graph retrieval or local step-wise planning. Fixed strategies apply predefined graph traversal or propagation procedures to different questions [24–26]. Although efficient, they cannot adapt their retrieval structures to query-specific multi-hop dependencies. Local planning strategies dynamically select the next entities, relations, or subgraphs according to currently retrieved information [27,28], but their limited local view may allow early retrieval errors to propagate. Prior question-decomposition approaches also divide complex questions into sub-questions, but usually organize them as linear sequences with implicit dependencies. These designs generally do not jointly provide global dependency modeling and relation-level grounding in source passages.

The PoE graph addresses the evidence-grounding limitation by associating source passages with the relation edges they express. Unlike passage-on-node, which may return every passage mentioning a retrieved entity, the PoE graph retrieves passages that directly support a selected relation facet. It therefore refines passage grounding from the entity level to the relation level while preserving both structured relations and their original textual context.

DAG-BQP addresses the planning limitation by extracting the required information units before retrieval, explicitly modeling their serial and parallel dependencies, and organizing them into a global sub-query DAG. Independent sub-queries can be executed in parallel, while answers from lower-level nodes are bound to dependent higher-level sub-queries. DAG-BQP and the PoE graph are further coupled during retrieval: each DAG node retrieves the required relation facets, the PoE graph maps them to their source passages, and the final answer-generation stage jointly uses the original query, structured facets, and textual evidence. PoP-RAG thus aligns information requirements, graph retrieval, and source-passage grounding within a unified workflow.

2.2.6. Limitations

The PoE graph supports both multiple passages per edge and many-to-many passage–edge associations. During retrieval, passages associated with the selected edges are collected and deduplicated as relation-level evidence. Although an edge linked to many passages may introduce contextual redundancy, these passages remain directly related to the required relation. This is preferable to passage-on-node, where passages retrieved through a target entity may not express the relation required by the current sub-query. Thus, complex passage–edge associations may increase the evidence volume without weakening the PoE graph's advantage in evidence relevance.

A more prominent limitation of PoP-RAG is the additional query-planning overhead introduced by DAG-BQP. Before retrieval, the system must extract information units, analyze their dependencies, and construct a global sub-query DAG. Its query latency is therefore necessarily higher than that of retrieval approaches that do not employ any planning process. This reflects a trade-off between accuracy and efficiency: the additional planning steps increase processing time, but explicitly modeling multi-hop dependencies improves the completeness of key-facet retrieval and the accuracy of the final answer. The ablation results similarly show that removing DAG-BQP can reduce the planning overhead but leads to clear decreases in both EM and F1. PoP-RAG therefore incurs additional query time in exchange for better multi-hop retrieval and question-answering performance.

3. Results

3.1. Experimental Setup

3.1.1. Datasets

Since the objective of PoP-RAG is to address and optimize the limitations of traditional GraphRAG in multi-hop QA tasks, we have chosen three highly representative multi-hop QA datasets for experimental purposes: *2WikiMultihopQA* [33], *MuSiQue* [34], and *HotpotQA* [35]. In line with previous work [22,26,36,37], we randomly sampled 1000 queries from each validation set.

- 2WikiMultihopQA. This is a multi-hop QA dataset from Wikipedia, testing models' ability to reason across documents and assess deep reasoning through diverse questions.
- MuSiQue. This is a multi-hop QA dataset by NUS and AI2, evaluating complex multi-step reasoning with questions requiring at least two steps, preventing single-hop answers.

- HotpotQA. This is a multi-hop QA dataset by Stanford, CMU, and Montreal University, challenging models to use multiple documents for reasoning beyond single-hop matching.

3.1.2. Baselines

In terms of baseline selection, we select baselines from representative RAG methodologies. For intuitive RAG method, we include *NaiveRAG* [14]. For graph-based RAG methods, we choose representative GraphRAG methods covering *MGRAG* [24], *LightRAG* [25], *HippoRAG 2* [26] and *ToG* [27].

3.1.3. Evaluation Metrics

To evaluate the performance of the proposed method in multi-hop question answering scenarios, we adopt the *exact match (EM)* score and *F1* score as standard evaluation metrics, which are widely used for assessing open-domain QA tasks. The EM score is a binary metric that takes a value of 1 only when the generated answer is completely consistent with the reference ground-truth answer in semantics and key factual content, and 0 otherwise, offering a strict measurement of answer accuracy. The F1 score is the harmonic mean of precision and recall, effectively balancing the precision of retrieved information and the completeness of answer content. Benefiting from such a trade-off property, it is well-suited for evaluating the overall quality of open-ended question answering generation tasks.

3.1.4. Implementation Details

In our PoP-RAG, nvidia/NVEmbed-v2 is employed as the retriever in the vector retrieval phase; while GPT-4o-mini is utilized in such phases as knowledge graph construction, LLM-assisted support during the retrieval process (GPT-4o is used in sub-query DAG construction step), and answer generation. The number of triples retrieved by similarity search, N, is set to 10.

3.2. Main Results

To systematically evaluate the performance of PoP-RAG, we conducted experiments and presented the results in Table 2, including exact match (EM) and F1 scores, their averages across three multi-hop QA datasets, and improvements over the best-performing baseline. To mitigate fluctuations caused by the randomness of LLM-generated answers, each QA task was repeated five times, with the mean and standard deviation reported.

The experimental data demonstrate that the proposed PoP-RAG method achieves SOTA accuracy on all three multi-hop QA datasets: 2WikiMultihopQA, MuSiQue, and HotpotQA. Our method surpasses HippoRAG 2 (the best-performing baseline) with EM score improvements of 6.3%, 5.3%, and 1.3%, and F1 score gains of 6.9%, 5.9%, and 1.2% across the three datasets, respectively. This consistent accuracy advantage across datasets confirms the effectiveness of PoP-RAG in multi-hop question answering tasks.

3.3. Ablation Study

We conducted an ablation study to verify the effectiveness of PoP-RAG's two core innovations (i.e., the PoE graph and DAG-BQP). Four variants were tested: *w/o PoE graph*, using conventional indexing (direct passage on node); *w/o DAG-BQP*, using direct triple retrieval without any query planning; *w/o Info.*, directly constructing and executing the sub-query DAG from the original query without Information Extraction; and *w/o Attr.*, retaining Information Extraction and Sub-query DAG Construction while removing Attributive Enhancement. Results are shown in Table 3.

Experimental results show that removing either component leads to consistent performance degradation across all datasets. Removing the PoE graph (*w/o PoE graph*) yields clear performance declines: for instance, EM drops by 11.2%, 1.7% and 3.7%, and F1 decreases by 11.7%, 1.5% and 2.8% on the 2WikiMultihopQA, MuSiQue and HotpotQA datasets, respectively. Meanwhile, the *w/o DAG-BQP* variant exhibits the largest performance drop, with EM reduced by 22.0%, 13.4% and 10.0% and F1 decreased by 25.8%, 15.4% and 10.2% compared with the full PoP-RAG model on all three datasets, verifying the critical role of the DAG-BQP module. The two additional variants provide a finer-grained comparison within DAG-BQP. On 2WikiMultihopQA, *w/o Info.* achieves 59.6 in EM and 68.5 in F1, substantially outperforming *w/o DAG-BQP* in EM by 15.5% and F1 by 20.5%. *w/o Attr.* further reaches 64.4 in EM and 72.9 in F1. A similar performance trend is observed on MuSiQue and HotpotQA.

Further analysis of retrieval behavior (Figure 5) reveals that the *w/o PoE graph* variant retrieves significantly more passages than the full PoP-RAG system across all three datasets. Specifically, on the 2WikiMultihopQA, MuSiQue, and HotpotQA datasets, the average numbers of passages retrieved by the *w/o PoE graph* variant are

Table 2. QA performance of all methods across the datasets, measured by exact match (EM) and F1 (mean ± SD over 5 runs). The Improvement row reports the gain of PoP-RAG over the best-performing baseline.

RAG Methods	2Wiki		MuSiQue		HotpotQA		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
NaiveRAG	43.5 ± 0.19	51.5 ± 0.18	30.0 ± 0.30	40.2 ± 0.19	52.6 ± 0.30	66.6 ± 0.21	42.0	52.8
LightRAG	52.2 ±0.19	60.9 ± 0.27	25.7 ±0.18	36.1 ± 0.17	52.0 ±0.30	65.2 ± 0.22	43.3	54.1
MGRAG	45.8 ± 0.23	61.1 ± 0.32	27.2 ± 0.18	42.3 ± 0.22	51.2 ± 0.23	67.4 ± 0.32	41.4	56.9
ToG	51.9 ± 0.32	58.9 ± 0.27	27.7 ± 0.32	35.5 ± 0.34	48.0 ± 0.42	58.6 ± 0.25	42.5	51.0
HippoRAG 2	59.8 ± 0.40	66.9 ± 0.28	35.3 ± 0.31	45.4 ± 0.28	59.4 ± 0.22	71.7 ± 0.34	50.9	61.3
PoP-RAG (ours)	66.1 ± 0.20	73.8 ± 0.30	40.6 ± 0.31	51.3 ± 0.31	60.7 ± 0.23	72.9 ± 0.23	55.8	66.0
Improvement	+6.3	+6.9	+5.3	+5.9	+1.3	+1.2	+4.9	+4.7

Table 3. Ablation results on three datasets. The variants *w/o PoE graph*, *w/o DAG-BQP*, *w/o Info.*, and *w/o Attr.* denote the removal of the PoE graph, DAG-BQP, Information Extraction, and Attributive Enhancement, respectively.

Method	2Wiki		MuSiQue		HotpotQA	
	EM	F1	EM	F1	EM	F1
w/o PoE graph	54.9	62.1	38.9	49.8	57.0	70.1
w/o DAG-BQP	44.1	48.0	27.2	35.9	50.7	62.7
w/o Info.	59.6	68.5	37.3	48.5	56.4	68.2
w/o Attr.	64.4	72.9	39.8	50.2	59.9	72.2
PoP-RAG	66.1	73.8	40.6	51.3	60.7	72.9

23.56, 71.43, and 40.04, respectively, which are approximately 5.8, 10.7, and 9.4 times the corresponding PoP-RAG values (4.03, 6.68, and 4.28). This indicates that the PoE graph effectively filters redundant retrieved passages, enabling more concise and targeted information acquisition while maintaining superior performance.

3.4. Re-Evaluation on Filtered HotpotQA

To investigate PoP-RAG's relatively weaker accuracy gain on HotpotQA compared to other datasets, we conducted additional experiments. Existing research [34] has indicated that the HotpotQA dataset has certain quality flaws. Our analysis reveals that some presumed multi-hop questions are actually single-hop. For example, the question "What are the low sea stacks in the Shetland Islands called?" can be answered through direct retrieval.

Since the method we proposed is designed to solve multi-hop QA problems, we filtered the HotpotQA dataset used in the experiment to extract truly multi-hop questions. This filtering step was accomplished by guiding the LLMs with prompts. After filtering, we found that among 1000 test questions, only 835 are truly multi-hop questions. We then retested the accuracy of each baseline model on these 835 questions, and the specific results are shown in Table 4.

Experimental results indicate that PoP-RAG exhibits an accuracy improvement when handling actual multi-hop questions. The model's EM score has increased from 60.7% to 63.1%, and the F1 score has risen from 72.9% to 75.1%. Compared with the existing SOTA method, the accuracy advantage of PoP-RAG has expanded from the original 1.0% to nearly 3.0%.

3.5. Validity Rate of Retrieved Facts

This experiment evaluates whether the triples directly retrieved by PoP-RAG, HippoRAG 2, and ToG provide sufficient information to answer each query, thereby comparing retrieval effectiveness across the methods.

The test process is designed systematically: First, we collect and organize all retrieved triples to ensure data integrity. Then, for each question, we input its retrieved triples along with the query into LLMs. Using predefined criteria, the model evaluates whether the triples can answer the question.

As shown in Figure 1, PoP-RAG achieves the highest validity rates across all datasets, with 64.5% on 2WikiMultihopQA, 48.2% on MuSiQue, and 68.1% on HotpotQA. These validity rates are higher than those achieved by HippoRAG 2 (18.5%, 16.2%, 38.4%) and ToG (44.6%, 28.8%, 54.2%).

3.6. Failure Case Analysis

To further understand why the retrieved facets remain insufficient for answering some questions, we conducted a small-scale qualitative analysis of representative failure cases. For each case, we examined the complete execution trace, including the generated sub-query DAG, candidate and selected facets, intermediate sub-query answers, associated passages, and the final answer. The analysis provides representative examples corresponding to the three potential causes identified in our discussion.

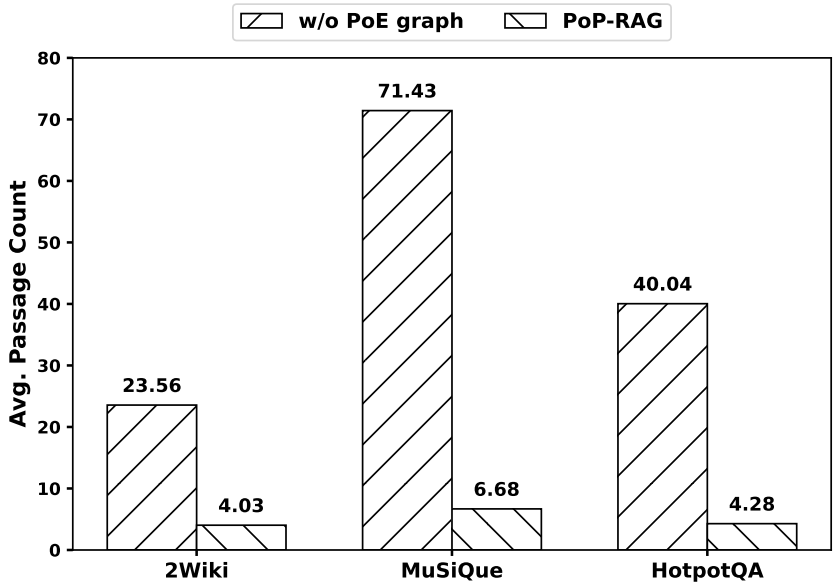

Figure 5. Average numbers of passages retrieved by PoP-RAG and its w/o PoE graph variant on 3 datasets.

Table 4. QA performance (mean ± SD over 5 runs) for all methods on the filtered HotpotQA subset.

RAG Methods	Filtered HotpotQA	
	EM	F1
NaiveRAG	45.2 ± 0.29	67.8 ± 0.18
LightRAG	52.9 ± 0.29	65.8 ± 0.22
MGRAG	52.1 ± 0.26	64.6 ± 0.31
ToG	48.3 ± 0.39	58.3 ± 0.42
HippoRAG 2	59.8 ± 0.38	72.5 ± 0.33
PoP-RAG (ours)	63.1 ± 0.33	75.1 ± 0.36

- Ambiguity in triple construction. In a question asking for the paternal grandmother of Archduke Ferdinand Karl Joseph of Austria-Este, the relevant fact was represented in the graph as "Ferdinand Karl – is a son of – Maria Theresa of Austria", whereas the subsequent sub-query referred to the same person as "Archduke Ferdinand of Austria-Este". The inconsistent entity representations prevented the correct triple from being matched. Consequently, the fallback process incorrectly returned Princess Maria Beatrice Ricciarda d'Este, who was the subject's mother, rather than Maria Theresa of Austria, who was his paternal grandmother. This case shows that entity-name variation introduced during triple construction may cause ambiguity between otherwise equivalent entities.
- LLM error during DAG construction. In a question asking for the wife of the actor playing Grace's boyfriend, the correct reasoning path was to identify Sam Elliott as the actor and then retrieve his wife, Katharine Ross. However, the generated DAG treated Sam Elliott as a fictional character and introduced an unnecessary sub-query asking which actor played Sam Elliott. The system then reversed the relation "Basil L. Plumley was portrayed by Sam Elliott" and propagated Basil L. Plumley into the subsequent spouse query. The final query therefore searched for the wife of Basil L. Plumley and failed to obtain the correct answer. This case demonstrates how an LLM may confuse entity types and relation directions during DAG construction, causing an early planning error to propagate through dependent sub-queries.
- Test-data flaw or evidence gap. In a question asking where the spouse of Frances Tupper was born, the annotated answer was Amherst. PoP-RAG correctly identified Frances Tupper's spouse as Charles Tupper. However, the supporting passage stated that Arthur Rupert Dickey, rather than Charles Tupper, was born in Amherst. Since the annotated answer could not be derived from the provided evidence, the system returned that the requested information was unavailable. This case reflects an inconsistency between the answer annotation and its supporting documents.

Together, these representative cases provide concrete evidence for the three potential failure causes identified in the Discussion: ambiguity introduced during triple construction, cognitive or logical errors made by the LLM, and inconsistencies or evidence gaps in the test data. The first and third categories are not intrinsic to the core designs of the PoE graph and DAG-BQP: the first originates from upstream NER and OpenIE processing, while the third results

from the benchmark itself. The second category is directly related to LLM-based DAG planning, and its robustness and cascading effects are further discussed in the Discussion section. Overall, these cases represent localized rather than systematic failure patterns and do not alter the general performance trend, as PoP-RAG consistently outperforms the strongest baseline across all three datasets in both EM and F1.

3.7. Efficiency and Cost

We evaluate the efficiency of PoP-RAG in terms of online query latency, LLM calls, token consumption, and offline indexing cost, as shown in Table 5. All methods are evaluated across 2WikiMultihopQA, HotpotQA, and MuSiQue for a fair comparison.

We observe several distinct cost patterns across the evaluated methods. HippoRAG 2 achieves the lowest average online latency of 5.91 s, whereas MGRAG incurs the highest online token consumption at 349,552 tokens per query, together with 91.8 M input tokens, 28.9 M output tokens, and 194.2 MB of graph storage during offline indexing. NaiveRAG only performs text chunking without LLM-based indexing or graph construction; consequently, its reported offline LLM token consumption and graph storage are both zero. PoP-RAG and HippoRAG 2 have the same offline LLM token cost of 7.2 M input and 2.4 M output tokens because they adopt the same text chunking and NER/OpenIE extraction pipeline. Nevertheless, PoP-RAG requires only 49.1 MB of graph storage, compared with 164.5 MB for HippoRAG 2, owing to their distinct graph organizations. ToG directly reuses the graph constructed by PoP-RAG and therefore shares its offline indexing cost. Overall, although holistic query planning increases PoP-RAG's average online latency to 34.05 s, its total online token consumption is only 9642 tokens per query, approximately 97% lower than MGRAG, demonstrating a favorable balance between effectiveness and practical cost.

To further clarify where PoP-RAG's per-query latency is spent, we decompose its average end-to-end query time of 34.05s across 2WikiMultihopQA, HotpotQA, and MuSiQue, as reported in Table 5, into the four stages of the online query pipeline described in the Materials and Methods section. The average stage-wise breakdown across the three datasets is summarized in Table 6.

As shown in Table 6, Hierarchical Sub-query DAG Execution is the largest contributor to PoP-RAG's average latency, accounting for 11.07s (32.52%), followed by Sub-query DAG Construction at 10.06s (29.54%); together, these two stages account for 62.06% of the total latency. Information Extraction contributes 7.81s (22.93%), while Dual-Augmented Answer Generation incurs the lowest cost at 5.11s (15.01%). The execution overhead is primarily caused by LLM-based verification of whether the retrieved triples are sufficient to answer each sub-query, whereas dense retrieval, graph traversal, and PoE-based passage mapping involve comparatively lightweight similarity computation and lookup operations. Overall, PoP-RAG's latency mainly arises from global DAG planning and evidence verification rather than graph access itself.

3.8. Case Study

To further analyze the execution mechanism of PoP-RAG in processing user questions, we selected an example from the 2WikiMultihopQA dataset for a case study, detailing the specific process of PoP-RAG in response generation, with the relevant process shown in Figure 6.

When PoP-RAG receives a question (e.g., "Which film whose director was born first, Meri Jung or La Fille Du Torrent?"), it first executes the information extraction step. The extracted information units include: "The director of Meri Jung is A", "A's birth date is C", "The director of La Fille Du Torrent is B", and "B's birth date is D". It can be seen that when the anaphoric pronouns in the information units are all parsed, the final answer to the question can be naturally derived by directly comparing C and D and identifying the earlier one. Meanwhile, to construct the sub-query DAG, it is necessary to extract the relationships between these information units: information units with dependencies are defined as serial relationships, and those without associations are defined as parallel relationships.

Next, in the sub-query DAG construction step, the sub-query DAG is built from information extraction results. A three-level sub-query DAG is constructed: Level 1 identifies the two films' directors; Level 2 queries their birth dates; Level 3 integrates this to determine who was born earlier. It is worth noting that in levels 2 and 3, the parts that depend on the results of the previous levels are identified by enclosing an anaphoric pronoun with "<answer> < /answer>".

The sub-query DAG is then executed. Each sub-query is answered using retrieved triples and their associated passages, and the resulting answers are used to update subsequent sub-queries until all have been answered. During final answer generation, the retrieved triples form a reasoning path, and their associated source passages are retrieved to support the final answer. In this example, "Meri Jung director Subhash Ghai (born 24 January 1945)" and "La Fille Du Torrent director Hans Herwig (born 10 April 1909)" are identified; the latter's earlier birthdate makes "La Fille Du Torrent" the final answer.

Table 5. Average online query and offline indexing costs across 2WikiMultihopQA, HotpotQA, and MuSiQue. Online metrics are reported as per-query averages, offline input and output tokens are reported in millions, and graph-related storage is reported in MB.

Method	Online Query Cost (per Query)					Offline Indexing Cost		
	Latency (s)	LLM Calls	Input	Output	Total	Input (M)	Output (M)	Graph Storage (MB)
NaiveRAG	10.23	1.0	4128	172	4300	0	0	0
LightRAG	17.70	2.0	25,218	214	25,432	53.3	14.2	52.5
MGRAG	31.36	21.2	348,902	650	349,552	91.8	28.9	194.2
ToG	43.48	15.9	22,584	543	23,127	7.2	2.4	49.1
HippoRAG 2	5.91	2.0	4310	154	4464	7.2	2.4	164.5
PoP-RAG (ours)	34.05	9.6	8308	1334	9642	7.2	2.4	49.1

Table 6. Stage-wise breakdown of PoP-RAG's average latency across 2WikiMultihopQA, HotpotQA, and MuSiQue, corresponding to the averaged result reported in Table 5.

Stage	Time (s)	Proportion (%)
Information Extraction	7.81	22.93
Sub-query DAG Construction	10.06	29.54
Hierarchical Sub-query DAG Execution	11.07	32.52
Dual-Augmented Answer Generation	5.11	15.01
Total	34.05	100.00

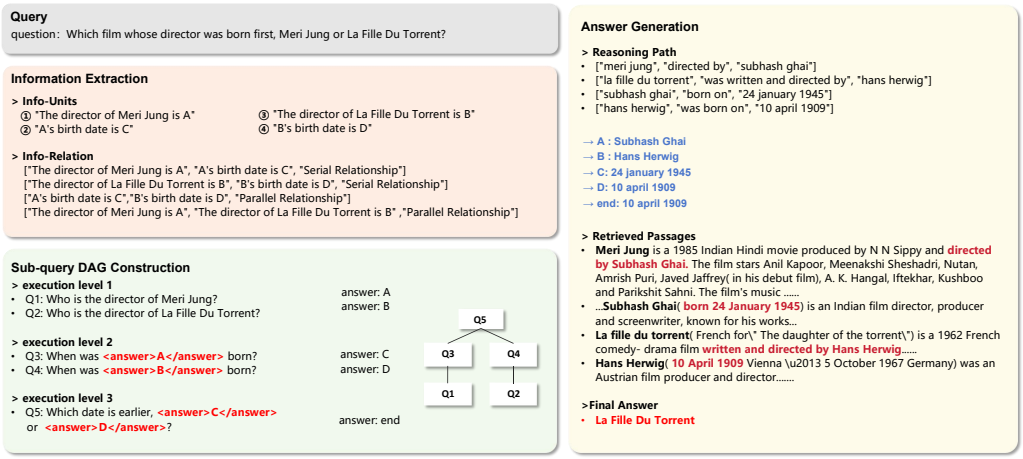

Figure 6. A case study of using PoP-RAG for response generation.

4. Discussion

The experimental results presented above demonstrate PoP-RAG's SOTA performance across all three multi-hop QA benchmarks. In this section, we analyze the underlying factors contributing to these results and discuss their broader implications.

PoP-RAG's accuracy gains systematically result from its core component design: the PoE graph establishes an efficient association between triples and source passages through optimized indexing, while DAG-BQP improves answer precision via hierarchical query decomposition. These components jointly boost multi-hop QA accuracy. The consistent performance advantage across all three datasets confirms that the improvements are not dataset-specific but reflect a fundamental enhancement in the multi-hop QA pipeline. It is worth noting that existing research [34] points out that HotpotQA is not an ideal dataset that can fully reflect the difficulty of multi-hop scenarios. Our experimental results show that PoP-RAG achieves larger accuracy improvements on the more challenging 2WikiMultihopQA and MuSiQue datasets than on HotpotQA, which indirectly confirms its advantages in handling complex multi-hop QA tasks.

The re-evaluation on the filtered HotpotQA dataset further substantiates this observation. Among the 1000 test questions, only 835 are truly multi-hop, meaning approximately 16.5% of the dataset consists of single-hop questions masquerading as multi-hop ones. After filtering, the accuracy advantage of PoP-RAG over the existing SOTA method expanded from 1.0% to nearly 3.0%. This finding has dual implications: firstly, it reveals the inherent limitations of current benchmark datasets in capturing the ability of complex multi-hop question answering; secondly, through systematic empirical analysis, it confirms the unique advantages of the PoP-RAG architecture in handling genuine multi-hop questions. This also suggests that future benchmarking efforts should incorporate more rigorous filtering to ensure dataset quality.

The validity rate analysis provides additional evidence for the effectiveness of the DAG-BQP method. The

substantial gap in retrieval validity rates across all datasets strongly confirms, from an empirical perspective, the prominent advantage of DAG-BQP in achieving precise matching between queries and information triples. By decomposing complex queries into a structured DAG of sub-queries, PoP-RAG is able to systematically explore the implicit entity associations that are often missed by direct retrieval approaches. The qualitative failure-case analysis in Section 3.6 provides concrete evidence for the previously suspected causes of the remaining retrieval failures. The representative cases show that failures may arise from inconsistent entity representations during triple construction, entity-type or relation-direction errors during LLM-based DAG construction, and inconsistencies between answer annotations and their supporting documents. These cases represent localized rather than systematic failure patterns and do not change the overall performance trend, as PoP-RAG still consistently outperforms the strongest baseline in both EM and F1.

Regarding DAG-BQP robustness, errors introduced during information extraction or DAG construction may propagate through serial dependencies. Because an upstream answer is bound to its dependent sub-queries, an incorrect intermediate answer can change downstream retrieval targets and affect the retrieved facets and final response. Such propagation is limited to the descendants of the affected node, while independent parallel branches remain unaffected. Dual-Augmented Answer Generation improves tolerance to these errors by jointly using the original query, valid facets, and associated passages. If a passage containing the key evidence has been retrieved, the final model may recover the correct answer despite an intermediate reasoning deviation. If an upstream error prevents the key evidence from being retrieved, the error may persist and lead to failure.

The ablation study offers key insights into the respective roles of the two core components. The PoE graph addresses the one-to-many entity-passage mapping issue—the w/o PoE graph variant retrieves significantly more passages (as shown in Figure 5), introducing noise in answer generation. PoP-RAG (with the PoE graph) effectively reduces the average retrieved passage count, demonstrating its effectiveness in minimizing redundancy and improving relevance. The *w/o PoE graph* variant performs worst on 2WikiMultihopQA, likely due to high article similarity in this dataset, where redundancy heavily interferes with answer generation. Since the length of a single passage is not very long, the resulting interference is less significant on the other two datasets. Meanwhile, the *w/o DAG-BQP* variant exhibits the most severe performance degradation, confirming that direct triple retrieval fails to explore omitted entities in multi-hop scenarios, leading to incomplete key information. This highlights DAG-BQP's core value in capturing implicit entity associations via DAG-based query planning. In summary, the PoE graph and DAG-BQP serve complementary roles: the former optimizes retrieval precision by reducing passage-level noise, while the latter ensures retrieval completeness through structured query decomposition. Their synergy is the key driver of PoP-RAG's overall performance gains. Compared with *w/o DAG-BQP*, *w/o Info.* retains the direct construction and execution of a sub-query DAG that explicitly represents dependent and parallel relationships among sub-queries and achieves substantially better performance. This comparison shows that organizing sub-queries into a structured DAG constitutes the primary source of the planning gain. The further improvement of the complete PoP-RAG over *w/o Info.* indicates that explicit information-unit extraction improves the quality of DAG construction. Meanwhile, the smaller but consistent gap between *w/o Attr.* and the complete model shows that preserving contextual constraints in the generated sub-queries provides an additional benefit. These results demonstrate that the main advantage of DAG-BQP arises from constructing a globally structured sub-query DAG, while Information Extraction and Attributive Enhancement provide complementary improvements.

In terms of efficiency, PoP-RAG has an average online latency of 34.05 s, which is of the same order as that of MGRAG and lower than that of ToG, although lightweight retrieval methods such as HippoRAG 2 achieve lower latency. This additional latency accompanies an average improvement of approximately 5% in both EM and F1 over HippoRAG 2. PoP-RAG consumes 9642 tokens per query, which is approximately 62%, 58%, and 97% lower than LightRAG, ToG, and MGRAG, respectively, although it remains higher than NaiveRAG and HippoRAG 2 because of its query-planning operations. During offline indexing, PoP-RAG consumes an average of 7.2 M input tokens and 2.4 M output tokens, while its graph-related storage is 49.1 MB, substantially lower than that of HippoRAG 2 and MGRAG. These results indicate that PoP-RAG incurs additional online planning latency in exchange for higher answer quality, while maintaining moderate token consumption and compact graph storage.

The stage-wise latency breakdown in Table 6 provides a more concrete view of this trade-off. Hierarchical Sub-query DAG Execution is the largest component at 11.07s (32.52%), followed by Sub-query DAG Construction at 10.06 s (29.54%); together, they account for 62.06% of the average query latency. Information Extraction contributes 7.81 s (22.93%), whereas Dual-Augmented Answer Generation accounts for 5.11 s (15.01%). The dominant overhead therefore arises from global sub-query planning and repeated LLM-based evidence-sufficiency verification, while dense retrieval, graph traversal, and PoE-based passage mapping remain comparatively lightweight. This interpretation is consistent with the ablation results: removing DAG-BQP causes the largest performance drop across all three benchmarks, reducing EM by 22.0%, 13.4%, and 10.0% and F1 by 25.8%, 15.4%, and 10.2% on

2WikiMultihopQA, MuSiQue, and HotpotQA, respectively. Together with PoP-RAG's higher valid-fact retrieval rates, these results show that its additional latency is concentrated in the planning and verification components that support its accuracy gains rather than in graph access itself.

Beyond accuracy, the case study illustrates PoP-RAG's interpretability advantages. By returning the reasoning path, sub-query DAG, and associated passages together, the system provides transparency that is particularly valuable in practical applications where users need to verify and trust the reasoning process. The structured nature of the sub-query DAG also provides a natural audit trail, enabling users to trace how each piece of evidence contributes to the final answer. Such interpretability advantages make PoP-RAG not only more accurate but also more suitable for deployment in scenarios requiring high accountability and trust.

5. Conclusions

This paper proposes PoP-RAG, a GraphRAG method for better multi-hop QA. It introduces a PoE graph and DAG-BQP in the indexing and querying stages. The PoE graph solves the one-to-many entity-passage mapping problem by directly connecting passage nodes with relation edges, and the DAG-BQP addresses the limitations of fixed and local graph traversal patterns through holistic query planning before execution. Experimental results on standard multi-hop QA benchmarks demonstrate that PoP-RAG outperforms current GraphRAG methods.

Author Contributions

Q.Z.: conceptualization (scheme design), software (code writing), investigation (experimental testing), writing—original draft (manuscript writing), writing—review & editing (manuscript revision); X.H.: visualization (figure drawing), writing—original draft (manuscript writing); Y.L. and H.L.: conceptualization (scheme design), writing—review & editing (manuscript revision); S.X.R., X.Y. and J.J.: supervision, writing—review & editing. All authors have read and agreed to the published version of the manuscript.

Funding

This work was sponsored by National Natural Science Foundation of China (NO. 62472327), Key R&D Program of Hubei Province (No. 2023BAB077), Sichuan Clinical Research Center for Imaging Medicine (YXYX2402).

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The datasets used in this study are publicly available, including 2WikiMultihopQA, MuSiQue and HotpotQA. All source code and experimental results can be provided upon reasonable request.

Acknowledgments

We thank all anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used AI tools to improve the clarity and fluency of the text. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

1. Touvron, H.; Martin, L.; Stone, K.; et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv* **2023**, arXiv:2307.09288.
2. Achiam, J.; Adler, S.; Agarwal, S.; et al. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774.
3. Bai, J.; Bai, S.; Chu, Y.; et al. Qwen technical report. *arXiv* **2023**, arXiv:2309.16609.

4. Chen, J.; Liu, Z.; Huang, X.; et al. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web* **2024**, *27*, 42.
5. Hadi, M.U.; Qureshi, R.; Shah, A.; et al. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Prepr.* **2023**, *1*, 1–26.
6. Tan, Y.; Min, D.; Li, Y.; et al. Can ChatGPT replace traditional KBQA models? An in-depth analysis of the question answering performance of the GPT LLM family. In Proceedings of the International Semantic Web Conference, Athens, Greece, 6–10 November 2023; Springer: Cham, Switzerland, 2023; pp. 348–367.
7. Yang, J.; Jin, H.; Tang, R.; et al. Harnessing the Power of Llms in Practice: A Survey on ChatGPT and Beyond. *arXiv* **2023**, arXiv:2304.13712.
8. Yu, X.; Zhang, Z.; Niu, F.; et al. What Makes a High-Quality Training Dataset for Large Language Models: A Practitioners' Perspective. In Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, Sacramento, CA, USA, 27 October–1 November 2024; pp. 656–668.
9. Villalobos, P.; Ho, A.; Sevilla, J.; et al. Will we run out of data? Limits of LLM scaling based on human-generated data. *arXiv* **2022**, arXiv:2211.04325.
10. Ji, Z.; Yu, T.; Xu, Y.; et al. Towards mitigating LLM hallucination via self reflection. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP, Singapore, 6–10 December 2023; pp. 1827–1843.
11. Friel, R.; Sanyal, A. Chainpoll: A high efficacy method for LLM hallucination detection. *arXiv* **2023**, arXiv:2310.18344.
12. McDonald, D.; Papadopoulos, R.; Benningfield, L. Reducing llm hallucination using knowledge distillation: A case study with mistral large and mmlu benchmark. *Authorea Prepr.* **2024**. https://doi.org/10.36227/techrxiv.171665607.76504195/v1.
13. Lewis, P.; Perez, E.; Piktus, A.; et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
14. Gao, Y.; Xiong, Y.; Gao, X.; et al. Retrieval-augmented generation for large language models: A survey. *arXiv* **2023**, arXiv:2312.10997.
15. Siriwardhana, S.; Weerasekera, R.; Wen, E.; et al. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Trans. Assoc. Comput. Linguist.* **2023**, *11*, 1–17.
16. Arslan, M.; Ghanem, H.; Munawar, S.; et al. A Survey on RAG with LLMs. *Procedia Comput. Sci.* **2024**, *246*, 3781–3790.
17. Fan, W.; Ding, Y.; Ning, L.; et al. A survey on rag meeting llms: Towards retrieval-augmented large language models. In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Barcelona, Spain, 25–29 August 2024; pp. 6491–6501.
18. Zhang, Q.; Chen, S.; Bei, Y.; et al. A survey of graph retrieval-augmented generation for customized large language models. *arXiv* **2025**, arXiv:2501.13958.
19. Procko, T.T.; Ochoa, O. Graph retrieval-augmented generation for large language models: A survey. In Proceedings of the 2024 Conference on AI, Science, Engineering, and Technology (AIxSET), Laguna Hills, CA, USA, 30 September 2024–2 October 2024; pp. 166–169.
20. Boccaletti, S.; Latora, V.; Moreno, Y.; et al. Complex networks: Structure and dynamics. *Phys. Rep.* **2006**, *424*, 175–308.
21. Peng, B.; Zhu, Y.; Liu, Y.; et al. Graph retrieval-augmented generation: A survey. *arXiv* **2024**, arXiv:2408.08921.
22. Jimenez Gutierrez, B.; Shu, Y.; Gu, Y.; et al. Hipporag: Neurobiologically inspired long-term memory for large language models. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 59532–59569.
23. Han, H.; Wang, Y.; Shomer, H.; et al. Retrieval-augmented generation with graphs (graphRAG). *arXiv* **2024**, arXiv:2501.00309.
24. Edge, D.; Trinh, H.; Cheng, N.; et al. From local to global: A graph rag approach to query-focused summarization. *arXiv* **2024**, arXiv:2404.16130.
25. Guo, Z.; Xia, L.; Yu, Y.; et al. LightRAG: Simple and fast retrieval-augmented generation. *arXiv* **2024**, arXiv:2410.05779.
26. Gutiérrez, B.J.; Shu, Y.; Qi, W.; et al. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. In Proceedings of the Forty-Second International Conference on Machine Learning, Vancouver, BC, Canada, 13–19 July 2025.
27. Sun, J.; Xu, C.; Tang, L.; et al. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In Proceedings of the The Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
28. Jin, B.; Xie, C.; Zhang, J.; et al. Graph Chain-of-Thought: Augmenting Large Language Models by Reasoning on Graphs. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024, Bangkok, Thailand, 11–16 August 2024; pp. 163–184.
29. Wang, S.; Sun, X.; Li, X.; et al. GPT-ner: Named entity recognition via large language models. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, NM, USA, 29 April–4 May 2025; pp. 4257–4275.
30. Kolluru, K.; Adlakha, V.; Aggarwal, S.; et al. OpenIE6: Iterative Grid Labeling and Coordination Analysis for Open Information Extraction. *arXiv* **2020**, arXiv:2010.03147.
31. Fischer, A.; Greiff, S.; Funke, J. The Process of Solving Complex Problems. *J. Probl. Solving* **2011**, *4*, 19–42.
32. Tversky, A.; Kahneman, D. Utility, Probability, and Human Decision Making. *Science* **1974**, *185*, 1124–1131.

33. Ho, X.; Nguyen, A.K.D.; Sugawara, S.; et al. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv* **2020**, arXiv:2011.01060.
34. Trivedi, H.; Balasubramanian, N.; Khot, T.; et al. MuSiQue: Multihop Questions via Single-hop Question Composition. *Trans. Assoc. Comput. Linguist.* **2022**, *10*, 539–554.
35. Yang, Z.; Qi, P.; Zhang, S.; et al. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv* **2018**, arXiv:1809.09600.
36. Press, O.; Zhang, M.; Min, S.; et al. Measuring and Narrowing the Compositionality Gap in Language Models. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP, Singapore, 6–10 December 2023; pp. 5687–5711.
37. Trivedi, H.; Balasubramanian, N.; Khot, T.; et al. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Toronto, ON, Canada, 9–14 July 2023; pp. 10014–10037.