# SUBQRAG: SUB-QUESTION DRIVEN DYNAMIC GRAPH RAG

*Jiaoyang Li*[1,*], *Junhao Ruan*[1,*], *Shengwei Tang*[1], *Saihan Chen*[1], *Kaiyan Chang*[1], *Yuan Ge*[1], *Tong Xiao*[1,2,†], *Jingbo Zhu*[1,2]

[1] Northeastern University, Shenyang, China
[2] NiuTrans Research, Shenyang, China

## ABSTRACT

Graph Retrieval-Augmented Generation (Graph RAG) is effective at building a knowledge graph (KG) to connect disparate facts across a wide document corpus. However, this broad-view approach often lacks the deep structured reasoning required for complex multi-hop question answering (QA), leading to incomplete evidence and error accumulation. To address these issues, we propose SubQRAG[1], a sub-question driven framework that enhances reasoning depth. SubQRAG decomposes a complex question into an ordered chain of verifiable sub-questions. For each sub-question, it retrieves relevant triples from the graph. A key feature is its ability to adapt: if the original graph is insufficient, the system retrieves information from source documents, extracts new triples, and dynamically updates the graph in real time. Finally, all triples used in the reasoning process are aggregated into a "graph memory", providing a structured and traceable evidence path for the final answer generation. Experiments on three multi-hop QA benchmarks show that SubQRAG consistently achieves significant improvements, particularly in Exact Match scores.

***Index Terms*—** Graph RAG, Multi-hop QA, Sub-question Decomposition, Dynamic Graph Update, Graph Memory

## 1. INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable language understanding and generation capabilities on downstream tasks. However, when solving question answering (QA) tasks that require deeper reasoning and broader coverage, LLMs still face the constraints. Owing to the high cost of re-training, the internal knowledge of LLMs cannot be updated in real time and thus remains limited in coverage, which results in failures to incorporate the latest information and increases the risk of hallucinations in downstream QA tasks [1].

Retrieval-augmented generation (RAG) [2] addresses these challenges by retrieving knowledge from an external knowledge corpus. The standard RAG follows a single-step retrieval and generation framework. For simple QA tasks, this works well. However, this strategy is fragile for multi-hop QA because single-step retrieval can cause the retriever to narrowly focus on one dominant topic. Document-level knowledge often exhausts the limited retrieval window with evidence for a single hop, crowding out the distinct information needed to complete the full reasoning chain [3]. To advance beyond retrieving redundant document spans and to enable a more robust reasoning process, Graph RAG utilizes a knowledge graph (KG) structure. It compresses document-level knowledge into triples as $\langle entity, relation, entity \rangle$ to describe relationships between entities[4]. This allows a guided traversal through the knowledge base, such as in GraphRAG [5], facilitating the integration of multiple sources for complex QA.

Although graph RAG has advanced, it still faces limitations. A primary issue is its continued reliance on single-step retrieval, where the entire complex question is used to perform a single-step search for entry points into the graph. This approach fails to break down the reasoning process, making it difficult to dynamically adjust the retrieval focus for each logical hop [6]. This problem is further exacerbated by the static nature of the pre-constructed KG. Its structure is fixed and often incomplete, with partial coverage and coarse granularity weakening the reasoning support [7]. This inflexibility leads to a third major challenge: error accumulation. Because the initial entry nodes are determined by a single, holistic question, any imprecision can derail the entire reasoning chain. Subsequent pathfinding or node expansion inherits and amplifies these initial errors [8], causing irrelevant exploration of the graph and ultimate failure [9].

Motivated by these challenges, we propose SubQRAG, a framework designed to equip Graph RAG with two crucial capabilities: multi-step reasoning and real-time dynamic updating. To enable multi-step reasoning, SubQRAG first decomposes the original question into an ordered chain of sub-questions. It then addresses each sub-question sequentially, performing a focused retrieval of relevant triples at each step. To facilitate dynamic updating, if the pre-constructed graph lacks the necessary information, SubQRAG falls back to source documents. It retrieves relevant text, extracts new

∗ Equal contribution. † Corresponding author.
[1] https://github.com/ljy1228/SubQRAG

triples, and integrates them into the graph. Finally, all triples used in this adaptive reasoning process are aggregated into a coherent "graph memory". This memory provides a structured and traceable evidence path for the generator, mitigating the error accumulation inherent in undirected graph traversal.

Our contributions are summarized as follows:

- **Decompose sub-questions for dynamic retrieval**. We present SubQRAG, which decomposes the original question into sub-questions and retrieves relevant triples, helping dynamically adjust the retrieval process and provide richer evidence.
- **Dynamic graph updating**. When the pre-constructed graph cannot answer the sub-questions, sub-questions drive KG updates by retrieving source documents.
- **Graph memory as structured guidance**. We integrate the triples supporting each sub-question into a subgraph and provide it to the generator. In this way, SubQRAG produces "graph memory" as reasoning trajectory, which reduces the limitations of the context window and improves QA performance.

## 2. METHOD

As shown in Figure 1, our proposed SubQRAG, a sub-question-driven dynamic Graph RAG framework, operates in four main stages. The process begins with (I) Offline Indexing, where LLMs pre-construct a KG by extracting structured knowledge triples from raw text. Next, during (II) Question Decomposition and Rewriting, a complex original question is broken down into a series of simpler, verifiable sub-questions; to maintain coherence, subsequent sub-questions are then rewritten to incorporate answers from preceding ones. According to simpler sub-questions, SubQRAG is able to perform (III) Retrieval and Dynamic Graph Updating, where for each sub-question, the framework attempts to retrieve supporting triples from the KG. Once the existing graph lacks the necessary information, SubQRAG will dynamically update the original graph by extracting new triples from the source documents. Finally, in the (IV) Answer Generation stage, all supporting triples gathered throughout the intermediate reasoning process are assembled into a graph-based memory, which is then provided to an LLM to synthesize a comprehensive final answer to the original question.

### 2.1. Offline Indexing

SubQRAG leverages an LLM to parse text documents, extract entities and their relations, and organize them into triples $\langle h, r, t\rangle$, which are then used to construct a corpus-level KG $\mathcal{G}$. The KG is formally defined in Eq. 1. In this graph, nodes $h, t \in \mathcal{E}$ represent entities, while the relation edge $r \in \mathcal{R}$ captures the semantic link between the two entities. To avoid redundancy, SubQRAG checks whether a newly generated triple already exists, ensuring the conciseness of $\mathcal{G}$.

$$\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E},\ r \in \mathcal{R}\} \tag{1}$$

### 2.2. Question Decomposition and Rewriting

#### *2.2.1. Question Decomposition*

In this stage, the goal of SubQRAG is to logically decompose the original multi-hop question $q$ into an ordered set of sub-questions $\{q_1, q_2, \ldots, q_n\}$, where each $q_i$ may depend on the results of its predecessors $\{q_1, q_2, \ldots, q_{i-1}\}$. This sequential dependency ensures that sub-questions are resolved in a logically consistent order.

#### *2.2.2. Question Rewriting*

In the question rewriting stage, we explicitly incorporate the answers from previously resolved sub-questions into the current sub-question. As shown in Eq. 2, for the $i$-th sub-question $q_i$, we leverage the answer set $A_{<i} = \{a_1, a_2, \ldots, a_{i-1}\}$ obtained from prior steps to better guide the rewriting process. The function $\text{Rewrite}(\cdot)$ injects key entities or values from $A_{<i}$ into $q_i$, transforming it into a self-contained, unambiguous question $q_i'$ while preserving its original intent. Rewriting eliminates referential ambiguity and contextual gaps, improving retrieval precision and reasoning reliability.

$$q_i' = \text{Rewrite}(q_i, A_{<i}) \tag{2}$$

### 2.3. Retrieval and Dynamic Graph Updating

For each sub-question, SubQRAG employs an embedding model $f(\cdot)$ to compute semantic similarity between question $q_i$ and triples in the pre-constructed KG. The top-k most relevant triples are selected shown in Eq. 3. Based on these triples, generator obtains the answer and the used triples $\mathcal{T}_i^{\text{used}}$ to support subsequent sub-question answering:

$$\mathcal{T}_{\text{sub}} = \underset{r \in \mathcal{G}}{\text{Top-k}} \cos\big(f(q_i),\ f((h, r, t))\big) \tag{3}$$

Once the generator cannot retrieve sufficient information from the selected triples to respond, SubQRAG falls back to corpus-level retrieval. At this stage, we retrieve relevant documents and generate the answer based on their context. From these documents, new triples $\mathcal{T}_{\text{new}}^{\star}$ are extracted, validated, and de-duplicated. The triples are then incrementally written back into the KG.

$$\mathcal{T}_{\text{new}}^{\star} = \text{Extract}(\mathcal{D}) \tag{4}$$

This above process leverages sub-questions' answering to continually update the pre-constructed KG. Rather than the model's parametric knowledge, each question is answered using only the retrieved triples.

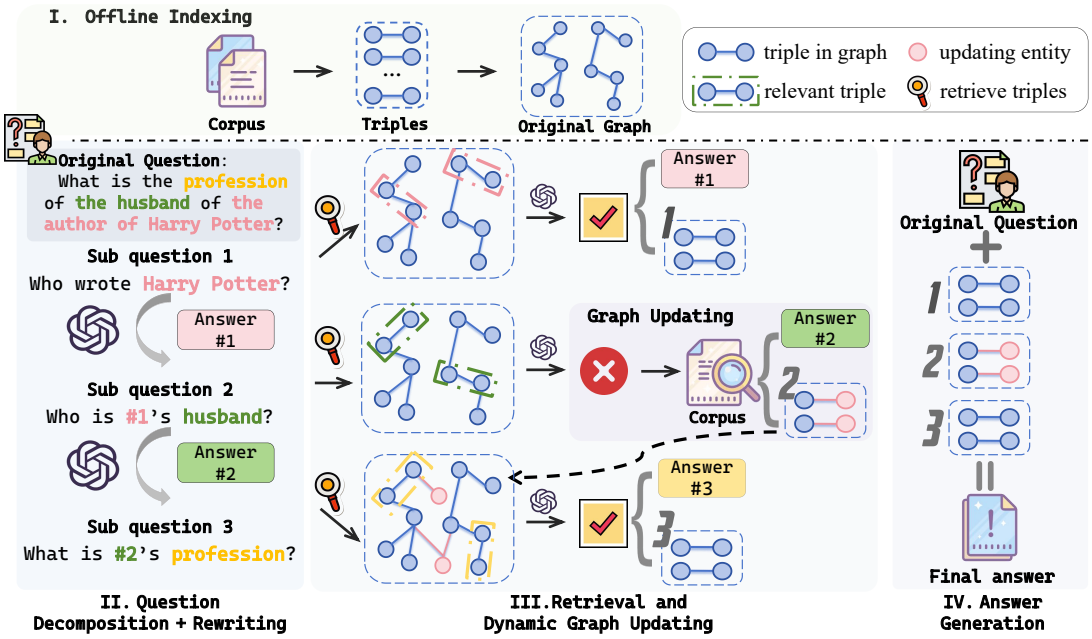


**Fig. 1**. The SubQRAG framework consists of four stages: (I) Offline Indexing: the corpus is processed into triples to construct the initial KG. (II) Question Decomposition + Rewriting: the original question is decomposed into verifiable sub-questions. (III) Retrieval and Dynamic Graph Updating: each sub-question retrieves supporting triples from the graph. If unanswered, documents are consulted to extract new triples and update the graph. (IV) Answer Generation: supporting triples are assembled into graph memory to generate the final answer based on the original question.

### 2.4. Answer Generation

After sub-question reasoning, we collect all triples that are actually used in the intermediate reasoning process to construct the graph memory as shown in Eq. 5.

$$\mathcal{T}^{\text{used}} = \bigcup_{i=1}^{m} \mathcal{T}_i^{\text{used}} \tag{5}$$

The final input to the generator consists of the original question $q$ together with the graph memory $\mathcal{T}^{\text{used}}$. Instead of document-level input, the generator leverages the context window more effectively and provides richer evidence. Meanwhile, this procedure yields graph memory in the form of traceable evidence paths, improving interpretability and reducing error accumulation.

## 3. EXPERIMENTS

### 3.1. Datasets

For evaluation, we adopt three widely used multi-hop question answering benchmarks: HotpotQA [15], MuSiQue [16] and 2WikiMultiHopQA [17]. These datasets cover different reasoning scenarios, including cross-document reasoning, complex contextual understanding and reasoning. This provides a comprehensive evaluation of SubQRAG on multi-hop QA tasks. To ensure both reproducibility and fairness, we strictly follow the experimental protocol of HippoRAG2 [14], using the same retrieval corpus and sampling 1,000 questions from each validation set as test questions. Under this unified setup, the performance of SubQRAG can be fairly compared against existing methods.

### 3.2. Baselines

We compare against three categories of baselines under a unified experimental setting. Zero-shot reports the performance of the backbone language model without RAG. RAG evaluates strong embedding-based retrievers, including GTR [10] and all-MiniLM-L6-v2 [11] , serving as natural baselines. Graph RAG covers structure-enhanced approaches: RAPTOR [12] builds a semantic hierarchy, GraphRAG [5] and LightRAG [13] use KGs for concept-level summaries, and HippoRAG2 [14] further augments a schema-less KG with passage nodes, context edges, and dense–sparse retrieval, providing a strong multi-hop QA baseline.

### 3.3. Metrics

We report two standard QA metrics. **Exact Match (EM)** measures whether the model's answer exactly matches the gold answer after simple normalization, serving as a strict accuracy metric. **F1** score computes the harmonic mean of precision and recall between the predicted and gold answers.

**Table 1**. EM and F1 scores on MuSiQue, 2Wiki, and HotpotQA multi-hop QA benchmarks. Zero-shot means evaluating the parametric knowledge of the backbone LLM without RAG. Retrieval-only baselines include embedding-based retrievers (GTR and all-MiniLM-L6-v2). Graph RAG baselines include RAPTOR, GraphRAG, LightRAG, and HippoRAG2. All graph RAG methods use all-MiniLM-L6-v2 as retriever and gpt-4o-mini as generator. This table highlights the best results in each column.

| Baseline Type | Method | MuSiQue | | 2Wiki | | HotpotQA | |
|---|---|---|---|---|---|---|---|
| | | EM | F1 | EM | F1 | EM | F1 |
| Zero-shot | None | 11.20 | 22.00 | 30.20 | 36.30 | 28.60 | 41.00 |
| Retrieval-only | GTR [10] | 14.20 | 21.78 | 29.80 | 35.01 | 42.10 | 50.66 |
| | all-MiniLM-L6-v2 [11] | 24.10 | 36.38 | 29.80 | 35.01 | 42.10 | 50.67 |
| Graph RAG | RAPTOR [12] | 4.80 | 12.20 | 24.70 | 26.38 | 29.50 | 37.50 |
| | GraphRAG [5] | 4.69 | 13.30 | 35.37 | 40.13 | 28.97 | 39.33 |
| | LightRAG [13] | 9.10 | 17.64 | 33.61 | 43.74 | 21.60 | 23.97 |
| | HippoRAG2 [14] | 28.20 | **40.70** | 50.60 | 59.79 | 51.90 | **66.18** |
| Our proposed | **SubQRAG** | **29.70** | 38.14 | **61.90** | **64.30** | **56.00** | 64.30 |

**Table 2**. Ablation performance on HotpotQA. w/o Decomposition removes the sub-question decomposition module. w/o Rewriting disables the question rewriting step. And w/o Update omits the dynamic updating graph.

| Method | EM | F1 |
|---|---|---|
| SubQRAG | **56.0** | **64.3** |
| w/o Decomposition | 50.5 (-5.5) | 59.6 (-4.7) |
| w/o Rewriting | 49.5 (-6.5) | 50.2 (-14.1) |
| w/o Update | 54.5 (-1.5) | 63.7 (-0.6) |

### 3.4. Implementation Details

For SubQRAG, we employ gpt-4o-mini as the backbone model across all key stages, including entity recognition, knowledge extraction, sub-question decomposition, triple construction, question rewriting and question answering. For the embedding component, we adopt all-MiniLM-L6-v2 as the retriever to encode and match triples and at each retrieval step the top-5 triples are selected to support answer generation. To ensure a fair comparison, all baselines, except zero-shot and GTR, are equipped with the same retriever and generator. All experiments for SubQRAG are conducted on a single NVIDIA RTX 3090 GPU.

### 3.5. Results

As shown in Table 1, we evaluate various retrievers across three benchmarks using gpt-4o-mini as the QA generator, comparing zero-shot, retrieval-only, graph-based RAG methods and our proposed SubQRAG. For zero-shot, although employing gpt-4o-mini, it still fails to achieve strong results. For standard RAG, the results on MuSiQue and HotpotQA show partial improvements owing to retrieving knowledge from an external knowledge base. Graph-based approaches that capture relationships among documents further improve performance, especially HippoRAG2. However, these methods still lack the ability to dynamically update the graph, and perform deeper multi-step reasoning. For our improved SubQRAG, it achieves consistent improvements on EM scores across three benchmarks, with gains of 5.3% on MuSiQue, 22.3% on 2Wiki, and 7.9% on HotpotQA.

### 3.6. Ablation Study

We design ablation experiments for the proposed SubQRAG on HotpotQA. Table 2 shows that removing decomposition, rewriting, and graph updating all lead to performance drops. Decomposition and rewriting are the key factors, while graph updating has a relatively modest effect since the KG is infrequently updated, but it still provides gains, confirming the effectiveness of sub-question driven updates.

## 4. CONCLUSION

In conclusion, we propose SubQRAG, a sub-question driven graph RAG framework which consists of four stages. Based on pre-construct KG, it first decomposes complex questions into verifiable sub-questions, then dynamically updates the KG when necessary for fine-grained retrieval and finally provides structured "graph memory" to the generator. Through leveraging multi-step reasoning and real-time graph updating, SubQRAG addresses incompleteness and error accumulation in the static graph RAG. Experiments on multi-hop QA benchmarks show that SubQRAG consistently surpasses existing graph RAG approaches, demonstrating its effectiveness in multi-hop QA tasks and achieving balance between the reasoning depth and knowledge base breadth.

## 5. REFERENCES


[1] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," *arXiv preprint arXiv:2212.10511*, 2022.

[2] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[3] Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang, "Efficientrag: Efficient retriever for multi-hop question answering," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 3392–3411.

[4] Travis Thompson, Seung-Hwan Lim, Paul Liu, Ruoying He, and Dongkuan Xu, "Inference scaled graphrag: Improving multi hop question answering on knowledge graphs," *arXiv preprint arXiv:2506.19967*, 2025.

[5] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson, "From local to global: A graph rag approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[6] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[7] Zihao Wang, Kwun Ping Lai, Piji Li, Lidong Bing, and Wai Lam, "Tackling long-tailed relations and uncommon entities in knowledge graph completion," *arXiv preprint arXiv:1909.11359*, 2019.

[8] Ziyi Cao, Bingquan Liu, and Shaobo Li, "Rpa: reasoning path augmentation in iterative retrieving for multi-hop qa," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, vol. 37, pp. 12598–12606.

[9] Ningning Zhang, Chi Zhang, Zhizhong Tan, Xingxing Yang, Weiping Deng, and Wenyong Wang, "Credible plan-driven rag method for multi-hop question answering," *arXiv preprint arXiv:2504.16787*, 2025.

[10] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang, "Large dual encoders are generalizable retrievers," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, Eds., Abu Dhabi, United Arab Emirates, Dec. 2022, pp. 9844–9855, Association for Computational Linguistics.

[11] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei, "MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, Eds., Online, Aug. 2021, pp. 2140–2151, Association for Computational Linguistics.

[12] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," in *The Twelfth International Conference on Learning Representations*, 2024.

[13] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang, "Lightrag: Simple and fast retrieval-augmented generation," 2024.

[14] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su, "From rag to memory: Non-parametric continual learning for large language models," 2025.

[15] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, Eds., Brussels, Belgium, Oct.-Nov. 2018, pp. 2369–2380, Association for Computational Linguistics.

[16] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal, "Musique: Multihop questions via single-hop question composition," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 539–554, 2022.

[17] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa, "Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps," in *Proceedings of the 28th International Conference on Computational Linguistics*, Donia Scott, Nuria Bel, and Chengqing Zong, Eds., Barcelona, Spain (Online), Dec. 2020, pp. 6609–6625, International Committee on Computational Linguistics.