Decompositional Reasoning for Graph Retrieval with Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) excel at many NLP tasks, but struggle with multi-hop reasoning and factual consistency, limiting their effectiveness on knowledge-intensive tasks like complex question answering (QA). Linking Knowledge Graphs (KG) and LLMs has shown promising results, but LLMs generally lack the ability to reason efficiently over graph-structured information. To tackle this problem, we propose a novel retrieval approach that integrates textual knowledge graphs into the LLM reasoning process via query decomposition. Our method decomposes complex questions into sub-questions, retrieves relevant textual subgraphs, and composes a question-specific knowledge graph to guide answer generation. For that, we use a weighted similarity function that focuses on both the complex question and the generated subquestions to extract a relevant subgraph, which allows efficient and precise retrieval for complex questions and improves the performance of LLMs on multi-hop QA tasks. This structured reasoning pipeline enhances factual grounding and interpretability while leveraging the generative strengths of LLMs. We evaluate our method on standard multi-hop QA benchmarks and show that it achieves comparable or superior performance to competitive existing methods, using smaller models and fewer LLM calls. *Source code will be available upon acceptance.*

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable success across a wide range of natural language processing (NLP) tasks (Brown et al. (2020), Chowdhery et al. (2023), Touvron et al. (2023a), Ouyang et al. (2022)), including question answering (Kamalloo et al., 2023), summarization (Liu et al., 2024), and machine translation (Zhang et al., 2023). As LLMs have grown in size and have been trained on increasingly diverse and large datasets, their emergent ability

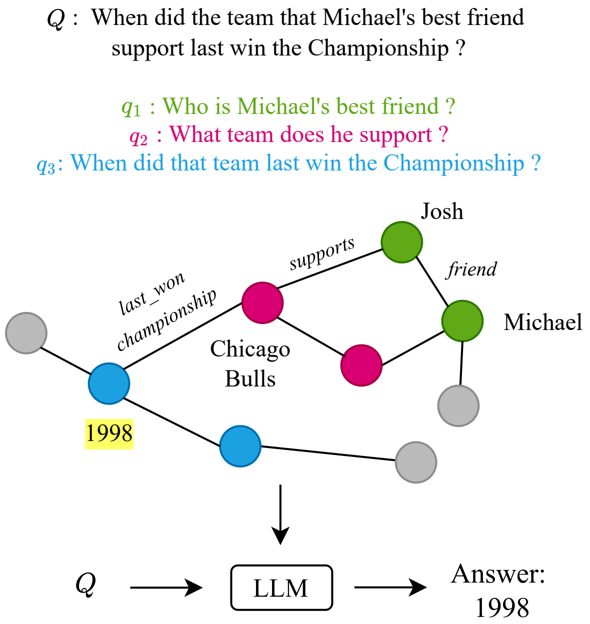


Figure 1: Illustration of our decompositional retrieval-based reasoning method. Our method decomposes the question into sub-questions, performs iterative, context-aware retrieval conditioned on previous answers, and merges the resulting subgraphs for guided reasoning.

to perform different types of reasoning (Wei et al. (2022a), Zhou et al. (2023)), ranging from arithmetic (Imani et al., 2023) and neurosymbolic reasoning (Fang et al., 2024) to commonsense inference (Zhao et al., 2023), has become a central focus of recent research. This has opened new possibilities for solving complex problems that traditionally required structured or symbolic approaches (Pan et al. (2023), He-Yueya et al. (2023)). However, despite their broad capabilities, LLMs still struggle with tasks requiring multi-hop reasoning (Yang et al., 2024), factual grounding, or explicit access to structured knowledge. These models are prone to hallucinations and logical inconsistencies, particularly when operating in knowledge-intensive domains (Ji et al. (2023b), Huang et al. (2025)). This is partially due to the high reliance on implicit knowledge stored in parameters (Hu et al., 2024b), and the lack of explicit mechanisms for integrating or reasoning over structured information. Recent work on retrieval-augmented generation (Lewis et al., 2020), graph-augmented LLMs (Yasunaga et al., 2021), and neurosymbolic reasoning (Fang et al., 2024) has aimed to bridge this gap.

In this work, we address these issues by proposing a method for knowledge-guided decompositional reasoning with LLMs. Our method injects structured knowledge into the reasoning process using textualized knowledge graphs. Specifically, we decompose complex questions into sub-questions, retrieve relevant subgraphs from a textual knowledge graph, and merge them for structured and reasoning-enhanced retrieval (Figure 1). The obtained graph is then used to guide LLMs toward generating more accurate and interpretable answers. This hybrid approach enhances both the factual correctness and the transparency of LLM predictions, particularly in settings requiring multi-step reasoning over domain knowledge. To verify the effectiveness of our approach, we conduct extensive experiments on benchmark datasets for complex question answering, namely CWQ (Talmor and Berant, 2018) and WebQSP (Yih et al., 2016). We compare our method against standard prompting techniques, as well as existing state-of-the-art approaches that combine LLMs with knowledge graphs. Results show that our method achieves consistent improvements in accuracy without increasing the number of parameters for the LLM or the number of LLM calls, which shows the efficiency of our method. Our contributions are as follows:

- We develop a novel knowledge graph retrieval method that uses query decomposition, helping the LLM to reason over structured data for complex questions.
- We introduce a hybrid similarity function that uses the complex question and its decomposition to guide the retrieval process.
- We demonstrate improvements in accuracy and factual consistency on multi-hop QA benchmarks.
- Our method reduces the number of LLM-calls compared to other baselines, achieving a $3\times$ to $5\times$ reduction for both datasets.

2 Background

2.1 Can LLMs reason ?

LLMs such as GPT-3 (Brown et al., 2020), PaLM (Chowdhery et al., 2023), and LLaMA (Touvron et al., 2023a) have demonstrated strong performance across a wide range of language tasks, including reasoning-based benchmarks. Their ability to generalize in zero-shot (Kojima et al., 2022) and few-shot settings has led to the emergence of new prompting techniques, such as Chain-of-Thought (CoT) reasoning (Wei et al., 2022b), which improves multi-step reasoning by encouraging models to generate intermediate reasoning steps. Variants like self-consistency (Wang et al., 2023) further refines this by sampling multiple reasoning paths and aggregating answers for improved robustness. More recently, reinforcement learning has been used to entirely train new models (DeepSeek-AI et al., 2025) or improve model prompting (Pternea et al., 2024), showing great potential for the future.

Despite these advances, LLMs remain prone to hallucinations—generating fluent but factually incorrect or logically inconsistent outputs (Huang et al., 2025), (Srivastava et al., 2023), (Ji et al., 2023b). This is especially problematic in knowledge-intensive tasks requiring factual grounding, multi-hop reasoning, or domain-specific expertise (Ji et al. (2023a), Opsahl (2024)). These issues stem in part from the implicit nature of knowledge storage in model parameters, which limits their ability to verify facts or reason explicitly over external knowledge (Petroni et al. (2019), Bommasani et al. (2021)). Recent work has explored augmenting LLMs with tool use, such as code interpreters (Pi et al., 2022), equation solvers (He-Yueya et al., 2023) or symbolic solvers (Lam et al., 2024) (Pan et al., 2023), to externalize and validate parts of the reasoning process.

2.2 LLMs and graphs

Graphs offer a natural and interpretable way to represent real-world data through entities and their structured relationships. Integrating knowledge graphs with Large Language Models (LLMs) is a promising research direction that enables models to better handle real-life scenarios with structured data (Li et al., 2024) (Hu et al., 2024a). Knowledge graphs can enhance LLMs by providing explicit, grounded context, which

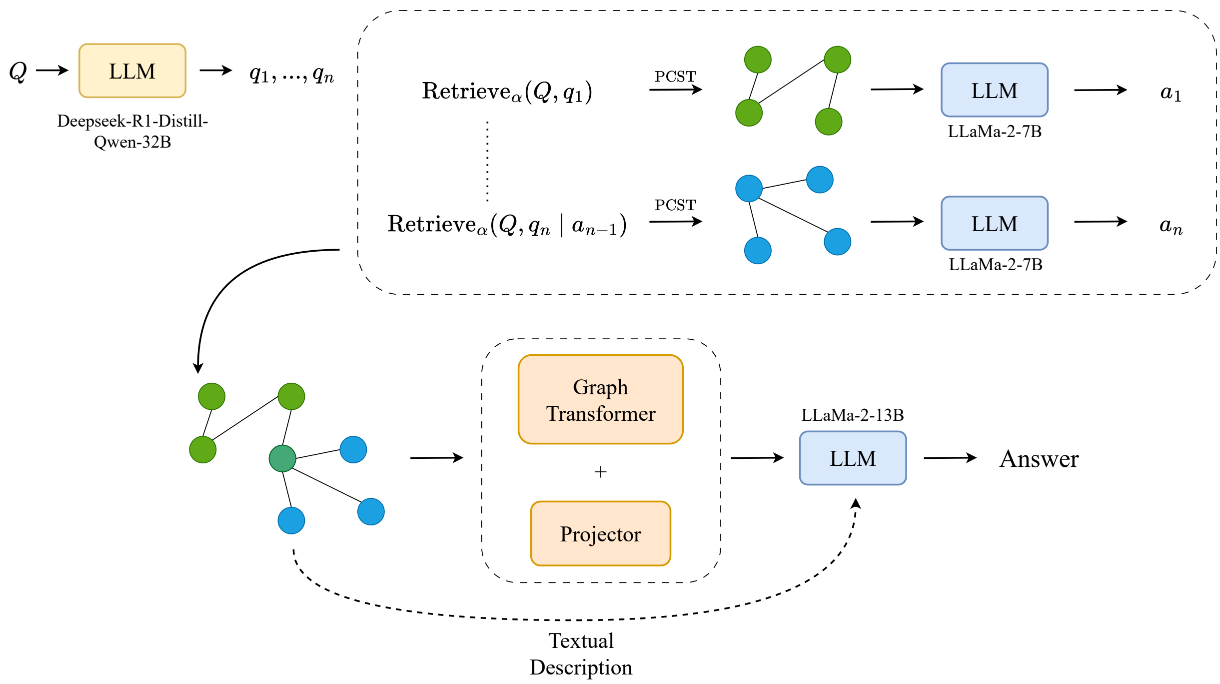


Figure 2: Overview of our retrieval method: the complex question is first decomposed into smaller subquestions, for which we iteratively perform retrieval and answer generation; once the retrieval is done for all subquestions, we merge the subgraphs and give the result as a hard (textualized graph) and a soft prompt (graph encoder output) to the model.

helps mitigate hallucinations (Li et al., 2024) (Agrawal et al., 2024), but also makes the model dependent on the noise or incompleteness of the graph (Dong et al., 2025). By grounding the generation process in a textualized or symbolic knowledge graph, LLMs can produce responses that are more accurate and aligned with real-world facts. This is especially useful in tasks such as question answering (Baek et al., 2023) (Yasunaga et al., 2021), logical reasoning (Choudhary and Reddy, 2024), or dialogue systems (Kang et al., 2023) where factual precision is crucial.

LLMs and graph neural networks (GNNs) can also be used together (Xu et al., 2024) (He et al., 2024a), each complementing the other. Graphs can be used to inject knowledge into LLMs via methods like structured prompting (Baek et al., 2023) (Zhang et al., 2024) or retrieval-based augmentation (Lewis et al., 2020) (Peng et al., 2024). LLMs can support and enhance graph-centred tasks (Pan et al., 2024) by performing entity linking, relation extraction, or even link prediction (Shu et al., 2025), which largely improves the graph's coverage. LLMs have also been explored as generators of graph-structured outputs or as interpretable reasoning agents over graphs using intermediate symbolic steps. In such hybrid frameworks, LLMs benefit from the structure and factual reliability of graphs, while graphs gain from the generalization and language understanding ability of LLMs (Pan et al., 2024). Nonetheless, most existing methods remain heuristic and lack a principled understanding of how best to align symbolic and neural representations (Cheng et al., 2025).

3 Related Work

Different methods have already demonstrated promising results on Knowledge Graph Question Answering (KGQA) tasks. He et al. (2024b) retrieves a subgraph from a textual knowledge graph and feeds it to the LLM without any explicit reasoning step, which can hinder performance on complex questions. Other existing techniques introduce some reasoning mechanisms within their framework: Sun et al. (2024) performs iterative entity and relation explorations, and directly reasons on the obtained paths. Similarly, Chen et al. (2024) uses task decomposition and then performs multiple cycles of exploration, memory updating, and evaluation. Performing iterative calls to the LLM has many advantages, but both mentioned techniques require using a relatively large model (LLaMa-2-70B, GPT-3.5 / GPT-4...) for planning and evaluation. In contrast, our method focuses on retrieving a more pertinent graph rather than answering iteratively, and uses fewer LLM calls—which can be controlled by the number of generated subquestions. Other methods like Luo et al. (2024) first predict reasoning paths as plans and search for those paths within the knowledge graph. Given that the LLM does not have any prior knowledge of the relations within the knowledge graph, the technique requires knowledge distillation into the LLM to generate faithful relation paths. Our method does not require any fine-tuning on the LLM, which reduces the cost of usage and the preprocessing time for new datasets.

4 Method

The overall pipeline for our method is presented in Figure 2. In order to tackle complex questions, we first decompose a complex question into a set of logically ordered subquestions. We then perform an iterative retrieval cycle by performing retrieval on the graph for each subquestion that we obtain. The results for the multiple retrievals are then combined into a single graph, which is then used to generate the final answer to the complex question.

4.1 Subquestions Generation

Given a complex question Q, we want to obtain a set of subquestions $\{q_1, ..., q_n\}$. The subquestions must be logically ordered (answering q_1 is necessary to answer q_2, etc.), atomic (can not be split into smaller subquestions), and cover all aspects of the complex question. Therefore, answering all subquestions in the given order should be equivalent to answering the complex question. In our work, we generate the subquestions using an LLM, leveraging its semantic understanding and its implicit knowledge capabilities. Using an LLM provides a flexible framework for decomposing complex questions, independent of the domain or the question type. To fulfill all the mentioned conditions above, we prompt the model with specific instructions about subquestions; we also provide some manually generated examples of decomposition to guide the model's behavior (see Appendix B for details about prompting).

4.2 Hybrid Entity Retrieval

For the retrieval part of the generation pipeline, we want to obtain a subgraph for each subquestion. Considering each subquestion independently might lead to very distinct subgraphs; moreover, the subquestions can lack sufficient contextual information on their own to retrieve all the relevant nodes and edges from the knowledge graph. To address this, we introduce a hybrid retrieval method that combines both the subquestion and the original complex question, allowing the model to benefit from the specificity of the former and retain the broader context provided by the latter. Our hybrid retrieval mechanism is implemented through a weighted similarity function, controlled by a parameter α, which balances the influence of both components. Figure 3 presents the equations for both node and edge retrieval.

When performing retrieval on the graph for the subquestion q_i, we keep track of the previous answer (answer a_{i-1} to subquestion q_{i-1}). This is crucial, as the answer to q_i might depend on the answer to q_{i-1}. Before retrieval, we embedded our complex question, the subquestions, and the textual attributes of the nodes/edges in the graph using a Sentence Transformer embedding model (see Appendix B for details). After having retrieved all necessary nodes and edges, we build a connected subgraph from these elements, following work done in He et al. (2024b). The connectivity of the graph is enforced by the Prize-Collecting Steiner Tree (PCST) algorithm (Bienstock et al., 1993), which optimizes the selection of a subgraph of maximum value based on node/edge weights and query similarity, under a size constraint.

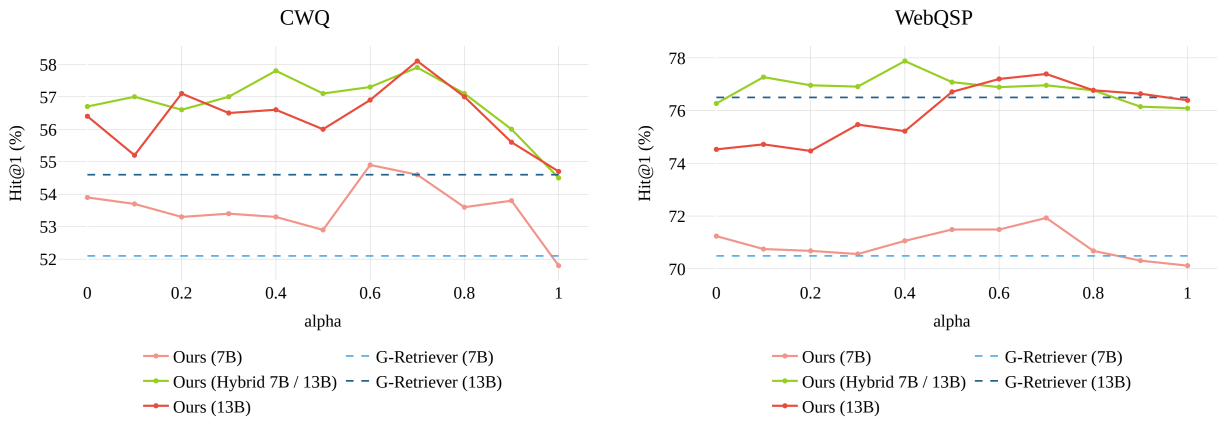
$$V_k{}^i = \underset{n \in V}{\text{argtopk}} \left[\alpha \cos(z_i, z_n) + (1-\alpha) \cos(z_Q, z_n)\right]$$

$$E_k{}^i = \underset{e \in E}{\text{argtopk}} \left[\alpha \cos(z_i, z_e) + (1-\alpha) \cos(z_Q, z_e)\right]$$

Figure 3: Hybrid retrieval: $V_k{}^i$ and $E_k{}^i$ denote the top nodes and edges from $G = (V, E)$, ranked by a weighted similarity to the subquestion q_i and original question Q. z_i, z_Q, z_n, and z_e are the embeddings of q_i, Q, node n, and edge e, respectively.

4.3 Subgraphs Merging

After retrieving subgraphs corresponding to each subquestion, we proceed to merge them in order to link relevant information and remove redundancy. Each subgraph is initially connected, as it is constructed using the PCST algorithm. To form the final graph, we take the union of all distinct nodes and edges across all subgraphs. Importantly, we do not directly enforce full connectivity during this



(a) CWQ benchmark (b) WebQSP benchmark

Figure 4: Model Accuracy (Hit@1) against the value of the α parameter for both CWQ and WebQSP datasets.

merging step, as doing so would require introducing virtual edges, which could potentially compromise the semantic integrity of the graph or resort to computationally expensive graph expansion methods.

4.4 Answer Generation

Once we obtained the merged graph for the complex question, we pass it to our LLM in two different ways, following the generation process described in He et al. (2024b): we provide a textualized version of the graph in the prompt, and also pass the graph through a trained graph encoder (Shi et al., 2021) followed by a linear projection layer. Providing the encoded graph as a soft prompt guides the LLM's response by feeding a trained embedding vector to the self-attention layers of the language model. When answering the complex question, we include the merged graph and its textual description in the prompt; we chose not to include the answers to the subquestions in the final prompt, as a single prior error can force the model to give a wrong answer, even when the graph contains the correct answer.

5 Experiments

5.1 Benchmarks

We evaluate our method on two different Question-Answering (QA) benchmarks to assess the quality of our results: CWQ (ComplexWebQuestions) (Yih et al., 2016) and WebQSP (WebQuestionsSemanticParses) (Talmor and Berant, 2018), which are both based on the Freebase (Bollacker et al., 2008) knowledge base. CWQ is a complex QA benchmark that focuses on multi-hop questions. As it needs the integration of multiple facts, it benefits from compositional reasoning, making it a suitable benchmark for our approach. WebQSP, on the other hand, contains a wide range of simple and factual questions. It also includes SPARQL annotations that we do not use in this work. We use the preprocessed version of the dataset provided in Luo et al. (2024).

5.2 Evaluation Metrics

We use the standard QA evaluation metrics found in related work. We report performance using accuracy and F1 scores. Accuracy measures exact matches, while F1 allows a more nuanced evaluation, especially when predictions are partially correct. In line with previous studies (Chen et al. (2024), Sun et al. (2024), Luo et al. (2024)), we use Hit@1 as our primary accuracy metric. Hit@1 determines whether the top prediction matches the ground truth and is widely used in QA evaluation. We report both Hit@1 and F1, enabling direct comparison with prior work.

5.3 Choice of language models

Our method requires two distinct capabilities, each handled by different classes of models. First, strong decompositional reasoning is needed to break down the complex question into logically ordered, comprehensive, and atomic subquestions. There, we use a Qwen-32B model distilled from Deepseek-R1 (DeepSeek-AI et al., 2025) for its advanced reasoning abilities. Second, we need efficient models to answer the subquestions and generate the final answer. For this, we experiment both with LLaMA-2-7B and LLaMA-2-13B (Touvron et al., 2023b). We also propose a "Hybrid 7B/13B" setting in which the 7B model answers the subquestions, while the

13B model handles the final complex question answer. The rationale is that atomic subquestions are simple and can be handled by a smaller model, while the final answer—requiring the integration of the full merged graph—benefits from the greater capacity of a larger model. This setting leverages model efficiency by allocating larger capacity only where necessary. We evaluate both uniform and hybrid settings in Section 6.

5.4 Balancing the Retrieval Query

Using only an atomic subquestion for retrieval can lead to ineffective results, as it lacks the broader context of the original complex question. To address this, we propose balancing the influence of the complex question and the current subquestion in the retrieval query embedding. We introduce an α parameter (Section 4.2) that controls this trade-off via a weighted average of their respective query embeddings. As shown in Figure 3, α determines the contribution of each: lower values (close to 0) emphasize the subquestion, while higher values shift focus toward the original complex question. When $\alpha = 1$, retrieval is based solely on the complex question, without any decompositional reasoning, as in He et al. (2024b) (see Figure 4).

6 Results

6.1 Influence of α parameter

During retrieval, we use both the complex question and its subquestions, with the α parameter controlling their relative importance in the query (Figure 3). We vary α and report model accuracy in Figure 4. We observe that using a larger model (13B) in the final answer stage (7B/13B and 13B setups) significantly outperforms the 7B-only setup, however using such a model for answering the subquestions offers no clear benefit, as we see, the hybrid 7B/13B and 13B-only setups yield similar results. Across all setups, extreme α values (near 0 or 1) underperform, and intermediate values work best. This supports the need to balance focus between subquestions and the main question during retrieval. In the rest of the paper, we use $\alpha = 0.7$.

Varying α also impacts the structure of the retrieved graph, potentially affecting the connectivity constraint previously ensured for subgraphs by the PCST algorithm. Higher α leads to more connected and denser merged graphs (Figures 9, 10), while lower values produce more distinct subgraphs and a sparser, occasionally disconnected graph. Although we observe connected graphs empirically yield better performance (Figure 5), disconnected ones remain rare in proportion (Figure 9). Despite the drop in performance in these cases, results remain competitive with state-of-the-art methods (Table 1). We discuss the statistical significance of these results in the Appendix A.

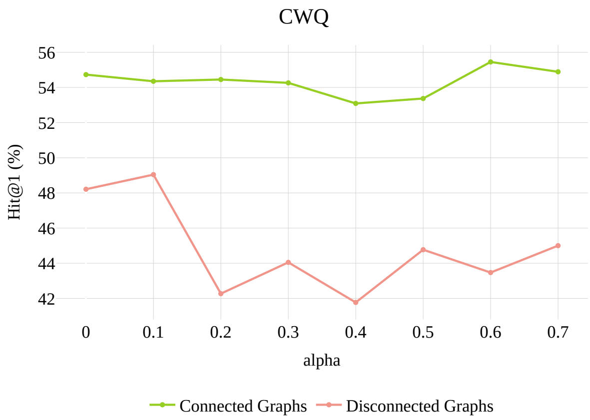


Figure 5: Model Accuracy (Hit@1) for connected and disconnected graphs against the value of the α parameter for the CWQ dataset.

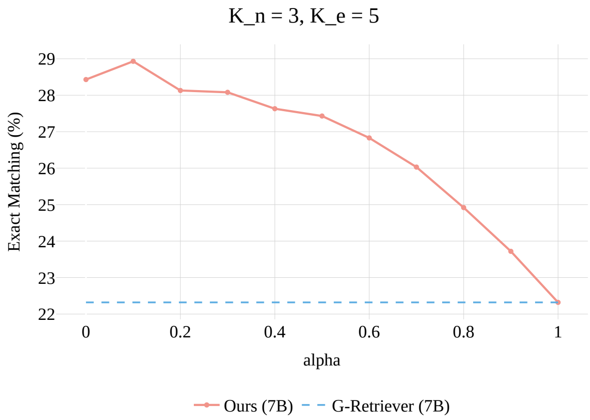


Figure 6: Exact Matching against the value of the α parameter, for the CWQ benchmark.

Since our focus is on improving retrieval, we report the Exact Matching scores for different α values. Exact Matching score is defined as the percentage of graphs containing a node that exactly matches the answer label. We observe in Figure 6 that focusing on the subquestions rather leads to Exact Matching scores. Setting $\alpha = 1$ serves as a sanity check to verify that we obtain similar metrics to He et al. (2024b). Additional results for Exact Matching can be found in Appendix A. We observe similar results for the Matching score: we define this metric as the percentage of retrieved graphs that contain a node very similar to the answer label (based on a cosine similarity between embeddings, using a similarity threshold of 0.9). This more flexible metric allows to check the presence of highly related nodes in the retrieved graph.

6.2 Graph size

K_n and K_e correspond respectively to the number of relevant nodes and edges that we consider to build the connected subgraph with PCST. At the retrieval step, we set the values of K_n and K_e to extract a certain number of relevant entities in the original graph (Figure 3). Choosing higher values of K_n and K_e leads to a higher quantity of retrieved information, which improves the probability of retrieved relevant nodes and edges, but also increases potential noise in the subgraph that we are building. Logically, choosing higher values of K_n produces significantly larger graphs (Figure 12 in Appendix A), which can be harder to handle for the LLM.

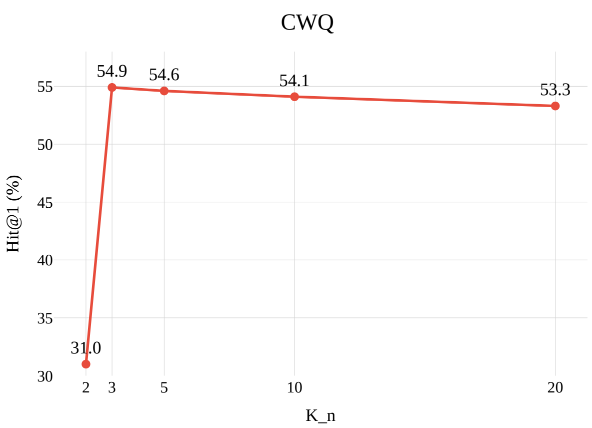


Figure 7: Accuracy (Hit@1) against the value of the K_n parameter (retrieved nodes), for the CWQ benchmark. Those results were obtained for our 7B model.

We also show (see Figure 7) that setting a high value for K_n (or K_e) does not lead to better performances for our 7B model. This observation has also been made in He et al. (2024b) on the WebQSP dataset. Setting K_n too low (e.g. $K_n = 2$) does not allow the model to retrieve enough knowledge in the graph; but setting K_n too high (above 5 empirically for our method) will add noise (non-relevant nodes) to the retrieved subgraphs and can disrupt the correctness of the generated answers. If we use a larger model (see Figure 8), the difference between using $K_n = 3$, $K_e = 5$) and $K_n = 5$, $K_e = 7$ (denoted as "larger graphs") does not lead to significant improvement; although we have a higher change of retrieving important nodes and edges, this observation highlights the presence of noise within the larger retrieved graphs. For the evaluation of our method, we use the default values of $K_n = 3$ and $K_e = 5$.

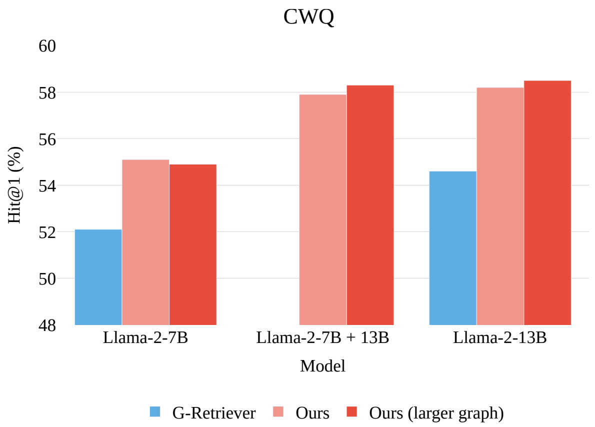


Figure 8: Model Accuracy (Hit@1) for different model sizes, for the CWQ benchmark. In default setting, we use $K_n = 3$, $K_e = 5$; for "larger graphs", we use $K_n = 5$, $K_e = 7$.

6.3 Main Results

For our main evaluation, we consider various baselines and model configurations. In particular, we highlight the "Hybrid 7B/13B" setting, where a 7B model answers each subquestion and a 13B model handles final answer generation (as described in Section 5.3).

Across both CWQ and WebQSP benchmarks (Table 1), our method achieves strong performance compared to approaches using similar model sizes and no fine-tuning. On CWQ, which features multi-hop questions, we observe a significant improvement over prior non-finetuned baselines, including those using larger models like Sun et al. (2024) (70B) and He et al. (2024b) (13B). On WebQSP, a simpler QA dataset, our method still outperforms related methods, though the margin is smaller—likely because decomposition is less helpful for single-hop questions. In both cases, only methods relying on dataset-specific fine-tuning or very large models (e.g., GPT-3.5 in (Chen et al., 2024)) achieve better scores, highlighting the value of simple decompositional

Method	**CWQ**		**WebQSP**	
	Hit@1	F1	Hit@1	F1
IO prompt (ChatGPT)	37.6	-	63.3	-
CoT (ChatGPT)	38.8	-	62.2	-
StructGPT (ChatGPT)	54.3	-	72.6	-
ToG (LLaMa-2-70B)	53.6	-	63.7	-
ToG (ChatGPT)	57.1	-	76.2	-
RoG (LLaMa-2-7B + FT)	62.6	56.2	**85.7**	70.8
PoG (GPT-3.5)	**63.2**	-	82	-
G-R (LLaMa-2-7B)	52.1	44.8	70.5	51.7
Ours (LLaMa-2-7B)	54.9	46	71.9	52.4
G-R (LLaMa-2-13B)	54.6	46.9	76.5	57.2
Ours (Hybrid 7B/13B)	57.9	50.3	**77.9**	**58.2**
Ours (LLaMa-2-13B)	**58.1**	**50.8**	77.4	56.4

Table 1: Performance comparison on the CWQ and WebQSP benchmarks. Bold indicates best results; underlined values indicate second-best. Results are sourced from the original papers: Brown et al. (2020), Wei et al. (2022b), Jiang et al. (2023), Sun et al. (2024), Luo et al. (2024), Chen et al. (2024), He et al. (2024b).

reasoning at the retrieval stage. A key observation is that our "Hybrid 7B/13B" setup performs comparably to a full 13B pipeline, suggesting that most of the benefits come from decompositional retrieval, not simply model scale. Figure 8 highlights this efficiency: we maintain competitive performance while using fewer resources, by relying on a lightweight model for subquestions and a larger one only for the final answer.

Finally, Table 2 compares the average number of LLM-calls for our method and compares it with baselines (Sun et al. (2024), Chen et al. (2024)) that made this data available. These methods use iterative cycles to answer the complex question, which does not give any upper-bound for the number of calls to the model. In our case, the number of calls to the model directly depends on the number of generated subquestions, which can ultimately be controlled via prompting at the decomposition step. We achieve state-of-the-art accuracy while reducing LLM usage for both CWQ and WebQSP, showing the efficiency of our decompositional retrieval method.

Since we use a single LLM call for both decomposition and final answer generation, we can deduce the average number of subquestions generated. Without setting a limit on the number of subquestions, we obtained an average of 2.8 subquestions for CWQ and 2.3 for WebQSP; this demonstrates that more complex questions result in more subquestions.

Method	**CWQ**	**WebQSP**
ToG	22.6	15.9
PoG	13.3	9.0
Ours	**4.8**	**4.3**

Table 2: Average number of LLM calls per question on the CWQ and WebQSP datasets

7 Conclusion

In this work, we introduced a novel graph retrieval method using decompositional reasoning with LLMs. By leveraging textual knowledge graphs and a hybrid retrieval mechanism that balances subquestion-specific and global context, our method enhances both the accuracy and interpretability of multi-hop QA. We demonstrated the effectiveness of our approach on complex QA benchmarks such as CWQ and WebQSP, achieving improved performance over strong baselines without increasing model size. Our results highlight the value of structured knowledge and explicit reasoning steps in addressing the limitations of LLMs in knowledge-intensive tasks. Future work may explore adding reasoning mechanisms at the generation step for improved model capabilities.

Limitations

Although our method demonstrated state-of-the-art results with smaller LLMs, we can mention some limitations of our method. Our method is mostly adapted to complex QA datasets, as the decomposition works best on difficult and multi-hop questions that can be transformed into a set of simple and atomic questions. The decomposition is not systematic, and we prompt the LLM to not decompose a question if it is considered to be simple enough; this approach can work on simple QA datasets (shown in Table 1 for the WebQSP dataset), but there is no guaranty that the model will not force the decomposition of simple questions.

We decompose complex questions using an LLM with a specific prompting technique shown in Appendix B. This method is advantageous for preprocessing an entire dataset, but it requires the use of a large enough LLM (we use Deepseek-R1-Distill-Qwen-32B, which is still relatively small compared to other baselines used for direct reasoning). Also, it is hard to control the quality of the decomposition; manual evaluation has been conducted to control the quality of the decomposition. It has been observed that some generated subquestions were redundant or irrelevant to the final goal, which can act as noise when providing them to the model.

Ethical Considerations

This work improves the reasoning abilities of large language models by using structured knowledge from textual graphs. While this improves the model's ability to make consistent and transparent predictions, it does not eliminate risks such as the propagation of biases present in the training data or the underlying knowledge graphs. We do not train new language models or use user-generated content. Our experiments are conducted using publicly available datasets. No personal or sensitive data is used. Nevertheless, caution should be exercised when deploying such systems in high-stakes or real-world applications, as flawed reasoning over structured data can result in factually inaccurate outputs.

References

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Can Knowledge Graphs Reduce Hallucinations in LLMs? : A Survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3947–3960, Mexico City, Mexico. Association for Computational Linguistics.

Jinheon Baek, Alham Fikri Aji, and Amir Saffari. 2023. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 78–106, Toronto, Canada. Association for Computational Linguistics.

Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, and David Williamson. 1993. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59(1-3):413–420.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, Vancouver Canada. ACM.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, and 1 others. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Liyi Chen, Panrong Tong, Zhongming Jin, Ying Sun, Jieping Ye, and Hui Xiong. 2024. Plan-on-graph: Self-correcting adaptive planning of large language model on knowledge graphs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Kewei Cheng, Nesreen K Ahmed, Ryan A Rossi, Theodore Willke, and Yizhou Sun. 2025. Neural-symbolic methods for knowledge graph reasoning: A survey. *ACM Transactions on Knowledge Discovery from Data*, 18(9):1–44.

Nurendra Choudhary and Chandan K. Reddy. 2024. Complex Logical Reasoning over Knowledge Graphs using Large Language Models. *arXiv preprint*. ArXiv:2305.01157 [cs].

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint*. ArXiv:2501.12948 [cs].

Na Dong, Natthawut Kertkeidkachorn, Xin Liu, and Kiyoaki Shirai. 2025. Refining noisy knowledge graph with large language models. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, pages 78–86.

Meng Fang, Shilong Deng, Yudi Zhang, Zijing Shi, Ling Chen, Mykola Pechenizkiy, and Jun Wang. 2024. Large Language Models Are Neurosymbolic Reasoners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17985–17993. Section: AAAI Technical Track on Natural Language Processing I.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024a. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. *arXiv preprint*. ArXiv:2305.19523 [cs].

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024b. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. In *Advances in Neural Information Processing Systems*, volume 37, pages 132876–132907. Curran Associates, Inc.

Joy He-Yueya, Gabriel Poesia, Rose E. Wang, and Noah D. Goodman. 2023. Solving Math Word Problems by Combining Language Models With Symbolic Solvers. *arXiv preprint*. ArXiv:2304.09102 [cs].

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2024a. A Survey of Knowledge Enhanced Pre-Trained Language Models. *IEEE Transactions on Knowledge and Data Engineering*, 36(4):1413–1430.

Peng Hu, Changjiang Gao, Ruiqi Gao, Jiajun Chen, and Shujian Huang. 2024b. Large language models are limited in out-of-context knowledge reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3144–3155.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, 43(2):1–55. ArXiv:2311.05232 [cs].

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. MathPrompter: Mathematical Reasoning using Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42, Toronto, Canada. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023a. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Delong Chen, Wenliang Dai, Ho Shu Chan, Andrea Madotto, and Pascale Fung. 2023b. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38. ArXiv:2202.03629 [cs].

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A General Framework for Large Language Model to Reason over Structured Data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating Open-Domain Question Answering in the Era of Large Language Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.

Minki Kang, Jin Myung Kwak, Jinheon Baek, and Sung Ju Hwang. 2023. Knowledge Graph-Augmented Language Models for Knowledge-Grounded Dialogue Generation. *arXiv preprint*. ArXiv:2305.18846 [cs].

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Long Hei Matthew Lam, Ramya Keerthy Thatikonda, and Ehsan Shareghi. 2024. A Closer Look at Logical Reasoning with LLMs: The Choice of Tool Matters. *arXiv preprint*. ArXiv:2406.00284 [cs].

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2024. A Survey of Graph Meets Large Language Model: Progress and Future Directions. In *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*, pages 8123–8131, Jeju, South Korea. International Joint Conferences on Artificial Intelligence Organization.

Yixin Liu, Kejian Shi, Katherine He, Longtian Ye, Alexander Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. 2024. On Learning to Summarize with Large Language Models as References. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8647–8664, Mexico City, Mexico. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Tobias Aanderaa Opsahl. 2024. Fact or fiction? improving fact verification with knowledge graphs through simplified subgraph retrievals. In *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pages 307–316.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph Retrieval-Augmented Generation: A Survey. *arXiv preprint*. ArXiv:2408.08921 [cs] version: 1.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Xinyu Pi, Qian Liu, Bei Chen, Morteza Ziyadi, Zeqi Lin, Qiang Fu, Yan Gao, Jian-Guang Lou, and Weizhu Chen. 2022. Reasoning Like Program Executors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 761–779, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Moschoula Pternea, Prerna Singh, Abir Chakraborty, Yagna Oruganti, Mirco Milletari, Sayli Bapat, and Kebei Jiang. 2024. The rl/llm taxonomy tree: Reviewing synergies between reinforcement learning and large language models. *Journal of Artificial Intelligence Research*, 80:1525–1573.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. 2021. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1548–1554. International Joint Conferences on Artificial Intelligence Organization.

Dong Shu, Tianle Chen, Mingyu Jin, Chong Zhang, Mengnan Du, and Yongfeng Zhang. 2025. Knowledge Graph Large Language Model (KG-LLM) for Link Prediction. In *Proceedings of the 16th Asian Conference on Machine Learning*, volume 260 of *Proceedings of Machine Learning Research*, pages 143–158. PMLR.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, and 432 others. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint*. ArXiv:2206.04615 [cs].

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint*. ArXiv:2302.13971 [cs].

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint*. ArXiv:2307.09288 [cs].

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent Abilities of Large Language Models. *arXiv preprint*. ArXiv:2206.07682 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022b. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Junjie Xu, Zongyu Wu, Minhua Lin, Xiang Zhang, and Suhang Wang. 2024. LLM and GNN are Complementary: Distilling LLM for Multimodal Graph Learning. *arXiv preprint*. ArXiv:2406.01032 [cs].

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do large language models latently perform multi-hop reasoning? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10210–10229.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The Value of Semantic Parse Labeling for Knowledge Base Question Answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. Prompting Large Language Model for Machine Translation: A Case Study. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 41092–41110. PMLR.

Qinggang Zhang, Junnan Dong, Hao Chen, Daochen Zha, Zailiang Yu, and Xiao Huang. 2024. KnowGPT: Knowledge Graph based Prompting for Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pages 6052–6080. Curran Associates, Inc.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. In *Advances in Neural Information Processing Systems*, volume 36, pages 31967–31987. Curran Associates, Inc.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and 1 others. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Experimental Results

The value of the α parameter, which controls the hybrid retrieval mechanism, can cause the retrieved graphs to be more or less connected. We see on Figure 9 that with a lower value of α, we sometimes produce disconnected graphs; at a higher value of α, most (if not all) graphs become naturally connected. Figure 5 suggests that the model better handles the connected graphs, as they lead to better results, but the low number of disconnected graphs questions the statistical significance of this hypothesis. We observe that for some alpha values, the p-value is less than 0.05. We also use a Beta law to estimate the posterior distribution of p, the parameter for the Binomial law that represents the accuracy of our predictions. Over the different alpha values, we obtain $P(connected > disconnected) = 0.919$, which indicates the plausibility of our claim.

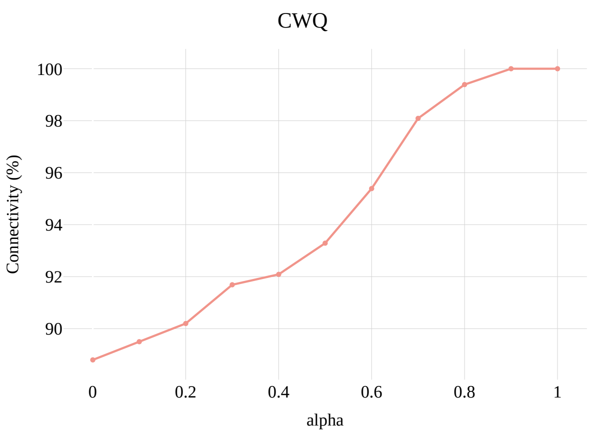


Figure 9: Graph connectivity against the value of the α parameter, for the CWQ benchmark.

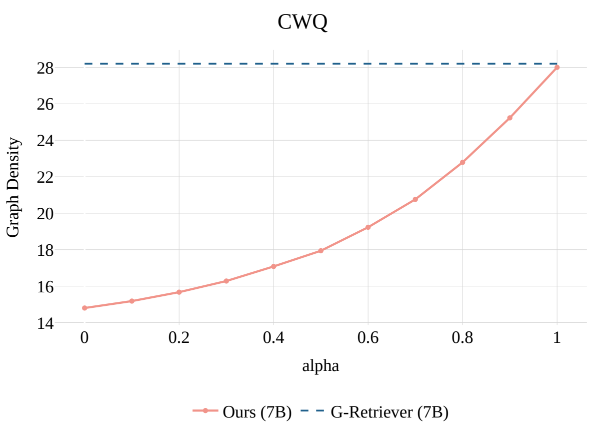


Figure 10: Graph density against the value of the α parameter, for the CWQ benchmark.

We make a similar observation with graph density; as shown in Figure 10, a lower α produces less dense graphs, but the retrieved graphs will be denser as the value of the parameter increases towards 1. Evidently, $\alpha = 1$ produces identical results to He et al. (2024b), as we only use the complex question. The density of a graph $G = (V, E)$ is given by Figure 11 :

$$D(G) = \frac{2 \cdot |E|}{|V| \cdot (|V| - 1)}$$

Figure 11: Graph density formula (undirected graph). $|V|$ and $|E|$ denote the number of nodes and edges in the graph $G = (V, E)$. Density quantifies how many edges exist compared to the maximum possible number of edges in the graph.

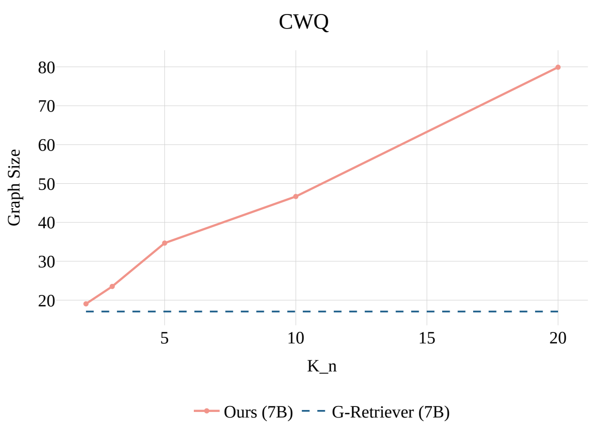


Figure 12: Graph size against the value of the K_n parameter (retrieved nodes), for the CWQ benchmark.

Figure 12 shows how changing K_n (similar effects with K_e) acts on the size of the final merged graph. As we perform retrieval for each subquestion (multiple times for each complex question), a small increase in the value of K_n will result in a much larger merged graph (each subgraph is larger). This effect will naturally have an impact on the performance of our method, as the model needs to process larger graphs.

The Exact Matching score is a metric that describes how often the exact answer to the complex question is found within the retrieved graph. We test the performance of our retrieval method with different models and retrieval settings (K_n and K_e), controlling the size of retrieved graphs. Overall, we observe that the α parameter has a high influence on the metric, which shows that our method improves the presence of target entities in the retrieved graphs. Also, regardless of the value of

α, all experiments show that we obtain higher Exact Matching than by simply using the complex question.

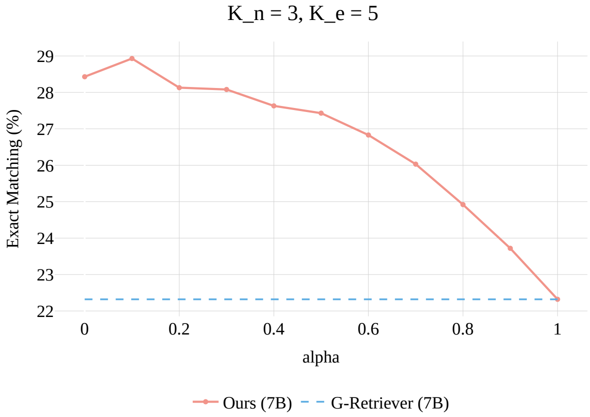


Figure 13: Exact matching (%) as a function of the α parameter on the CWQ benchmark for the 7B model with $K_n = 3$ and $K_e = 5$

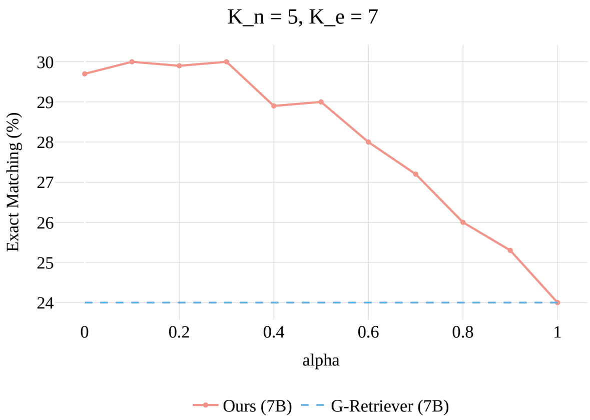


Figure 14: Exact matching (%) as a function of the α parameter on the CWQ benchmark for the 7B model with $K_n = 5$ and $K_e = 7$

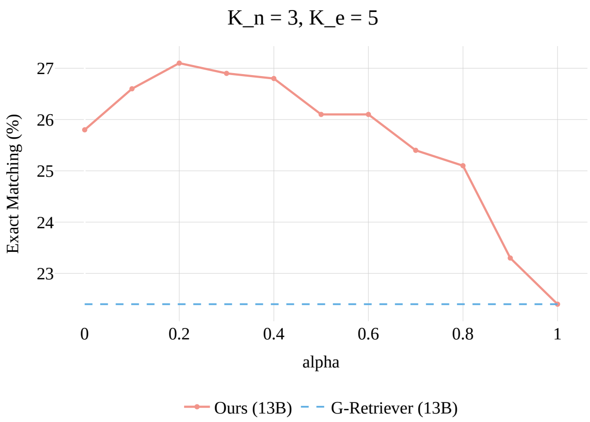


Figure 15: Exact matching (%) as a function of the α parameter on the CWQ benchmark for the 13B model with $K_n = 3$ and $K_e = 5$

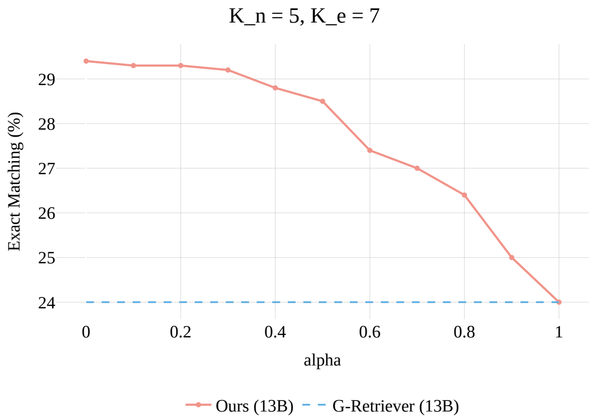


Figure 16: Exact matching (%) as a function of the α parameter on the CWQ benchmark for the 13B model with $K_n = 5$ and $K_e = 7$

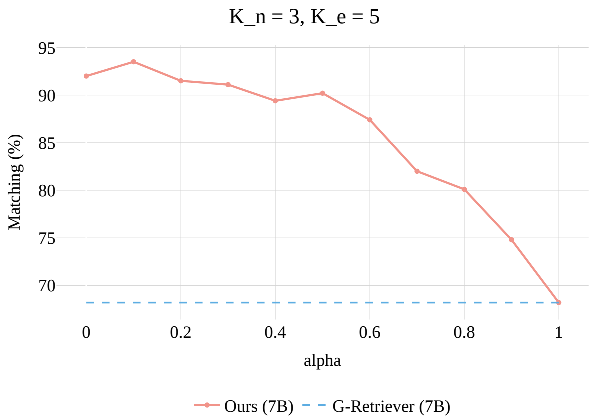


Figure 17: Matching (%) as a function of the α parameter on the CWQ benchmark for the 7B model with $K_n = 3$ and $K_e = 5$

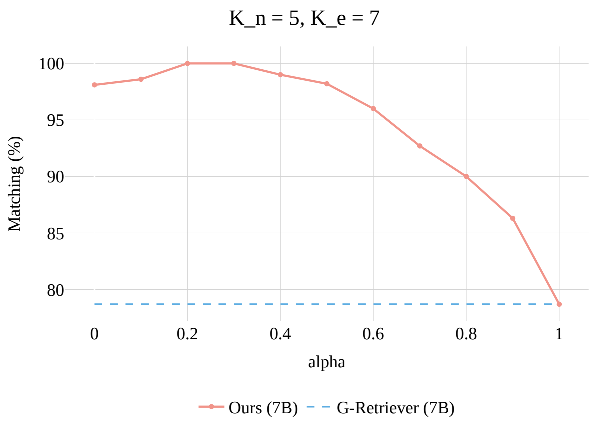


Figure 18: Matching (%) as a function of the α parameter on the CWQ benchmark for the 7B model with $K_n = 5$ and $K_e = 7$

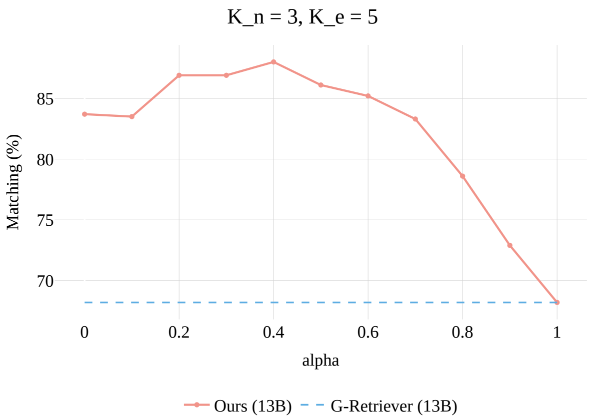


Figure 19: Matching (%) as a function of the α parameter on the CWQ benchmark for the 13B model with $K_n = 3$ and $K_e = 5$

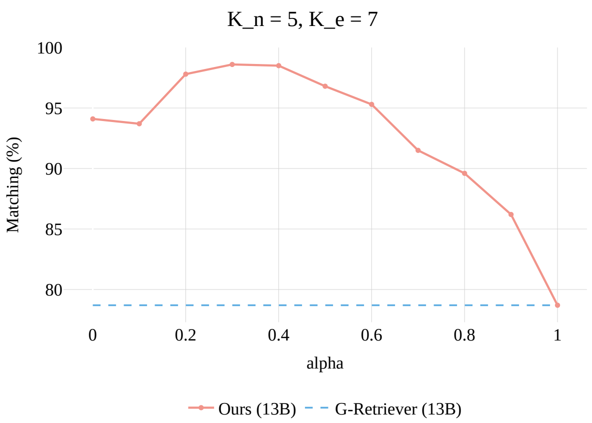


Figure 20: Matching (%) as a function of the α parameter on the CWQ benchmark for the 13B model with $K_n = 5$ and $K_e = 7$

Another useful metric to asses the quality of our retrieval method is the Matching metric. Compared to the Exact Matching, this metric allows for more flexibility and can evaluate the presence of highly similar entities (compared to the ground-truth answer) within the retrieved graph. We run experiments using a similarity threshold of 0.95 with the cosine similarity function. We make similar observations as for the Exact Matching metric, and we empirically show that our method achieves better Matching than previous methods.

We ablate key components of our pipeline on CWQ using LLaMa-2-7B (see Figure 3). Removing the graph encoder or textual representation leads to substantial drops in Hit@1 (-12.1, -9.1 points respectively), confirming the importance of both structured and textual graph information for accurate generation. At the graph retrieval stage, we measure the impact of treating the subquestions dependency or connecting subgraphs. Removing the dependency between subquestions is equivalent to the case where subquestions don't have access to previous subquestions and answers. Again, we observe the importance of both steps at the retrieval stage for the QA pipeline on complex questions.

Configuration	**Hit@1**	**Impact**
Full Pipeline	54.9	-
w/o Graph Encoding	42.8	-12.1
w/o Textual Graph	35.8	-9.1
w/o Subquestions Dependency	36.3	-18.6
w/o Graph Connectivity	43.7	-10.2

Table 3: Ablation study showing the impact of various components from the pipeline. The results provided were obtained on CWQ using LLaMa-2-7B.

B Experimental Setup

At the retrieval step, we encode all nodes and edges using Sentence-BERT model; we use a version based on the roberta-large model [1]. For the language models, we use Deepseek-R1-Distill-Qwen-32B [2] (DeepSeek-AI et al., 2025) for preprocessing (complex questions decomposition); for inference, we use both LLaMa-2-7B and LLaMa-2-13B (Touvron et al., 2023b). At all steps (question decomposition and answer generation), we use the models in a greedy setting (setting the temperature parameter to 0). For the generation pipeline and the choice of hyperparameters, we follow work done in He et al. (2024b) [3]. We set the maximum input text length of the model to 512 tokens and the maximum output size to 32 tokens. For prompt tuning, we set the number of virtual tokens to 10. The setup of the language models, along with the deterministic nature of the hybrid retrieval process, allows for reproducible results for identical runs. All reported results for our method correspond to a single run, and not a mean of different runs. For the graph encoder, we follow Shi et al. (2021) (4 layers, 4 attention heads per layer, hidden dimension of 1024); the following projection layer is a simple feedfoward neural network (2 linear layers, 1 activation layer), where the output size needs to match the hidden representation dimension for the LLM which is being used. For training the graph

[1] https://huggingface.co/sentence-transformers/all-roberta-large-v1

[2] https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B

[3] code used is under MIT license

Prompt for Subquestion Generation

You are an expert at decomposing complex questions into smaller, atomic subquestions. If the question can't be decomposed into smaller questions, leave it as it is. Decompose the following question into a list of simpler subquestions that:

- Are atomic (addressing only one piece of information at a time)
- Are logically ordered
- Have access to answers from previous subquestions
- Cover all necessary aspects of the original question
- Can be answered with a single entity
- Lead to the answer in the last subquestion

You must strictly format your answer as a valid JSON array; do NOT include explanations or reasoning.

Now decompose the following question in JSON format.

Complex Question:
"Which city is the birthplace of the author of the novel "1984"?"
Subquestions:
1. Who is the author of the novel "1984"?
2. Where was this author born?

Figure 21: Example of decomposition prompt for a complex question.

encoder, we use the AdamW optimizer (Loshchilov and Hutter, 2017); we train the graph encoder with a batch size of 4 for 10 epochs (with early stopping). The initial learning rate is set to 10^{-5}, with a weight decay of 0.05. At the retrieval step, when creating a connected graph using PCST, we choose to use the default values of $K_n = 3$ and $K_e = 5$.

For the datasets, we work with the preprocessed versions of CWQ [4] and WebQSP [5] obtained by Luo et al. (2024). For the dataset split, we use the default train and test sets proposed in the indicated versions.

[4]https://huggingface.co/datasets/rmanluo/RoG-cwq
[5]https://huggingface.co/datasets/rmanluo/RoG-webqsp

Few-shot Prompting

Examples:

Input: What is the capital of the country that exports the most honey ?
Output: ["Which country exports the most honey ?", "What is the capital of that country ?"]

Input: What sports team does Michael's best friend support ?
Output: ["Who is Michael's best friend ?", "What sports team does he support ?"]

Input: What fruits grow in the hottest countries from the largest continent in the world ?
Output: ["What is the largest continent in the world ?", "What countries are hottest on this continent ?", "What fruits grow in those countries ?"]

Input: How old is Obama ?
Output: ["How old is Obama ?"]

Now decompose the following question in JSON format.

Figure 22: Example of possible few-shot prompting

We propose an example of a prompt used for decomposing a complex question into multiple atomic and logically ordered subquestions. See Figure 21 for an illustration. Additionally, we provide examples of decomposition to the model to clarify the task and the expected output format. Figure 22 presents some decomposition examples on made-up complex questions; we also choose to add simple questions to show that decomposition is not always necessary.

C Compute Resources and Energy Consumption

We compute the total energy consumption for both CWQ and WebQSP datasets. For each model used, we use a single A100 40GB GPU. The LLaMa-2-13B model consumes more energy and also takes longer to run compared to LLaMa-2-7B. Our hybrid setup is a combination of both models, where we use LLaMa-2-7B for the subquestions, and LLaMa-2-13B only for the final question answering. We showed that we obtain similar accuracy results for the Hybrid 7B/13B model and for LLaMa-2-13B; but Table 4 shows that the hybrid option is much more economical, as we are able to reduce energy consumption by 17%. Compared to the LLaMa-2-7B model, the hybrid option only consumes 6% more energy, all experiments considered.

Model	**GPU**	**Energy (kWh)**
LLaMa-2-7B	A100 40GB	4.62
LLaMa-2-13B	A100 40GB	5.94
Hybrid 7B/13B	A100 40GB	4.95

Table 4: Energy consumption for test-set experiments across model configurations.

Model	**Dataset**	**GPU**	**Energy (kWh)**
R1-Q-32B	CWQ	H100 96GB	3.15
R1-Q-32B	WebQSP	H100 96GB	0.72

Table 5: Energy consumption for question decomposition (entire dataset preprocessing).

Task	**CO_2 Emissions ($kgCO_2e$)**
Preprocessing	2.27
Inference	6.2

Table 6: CO_2 Emissions (kg) for dataset preprocessing and model inference.

We also compute the total energy consumption for dataset preprocessing, which mainly consists of decomposing all questions in the dataset as a set of subquestions. For this task, we use a larger model (DeepSeek-R1-Distill-Qwen-32B), and we report the total energy consumption for each dataset in Table 5. Since the CWQ dataset is much larger than the WebQSP dataset, we observe a large difference in the energy needed in both cases.

Having given the energy consumption for our experiments, we compute the corresponding CO_2 emissions (Mass of CO_2 equivalent, $kgCO_2e$) for the different compute tasks (Table 6).