GLM-RAG: Graph Language Models for Graph-Based Retrieval-Augmented Generation

Maya Arseven **Anette Frank** **Beni Egressy*** **Johann Higl*** **Moritz Plenz***

Institute of Computational Linguistics, Heidelberg University
Aleph Alpha Research

{arseven, frank, plenz}@cl.uni-heidelberg.de
{maya.arseven, beni.egressy, johann.higl}@aleph-alpha-research.com

Abstract

Retrieval-augmented generation (RAG) over knowledge graphs requires retrievers that can effectively capture both graph structure and semantic information. Recent approaches have explored graph neural network (GNN)-based retrievers to model graph topology in multi-hop reasoning tasks. In parallel, graph language models (GLMs) have emerged as a promising paradigm that integrates graph reasoning and the semantic capabilities of language models. In this work, we introduce a GLM-based retriever and investigate the comparative strengths of GLM-based, GNN-based, and traditional vector-search-based retrievers in single- and multi-hop RAG settings, and with a particular focus on transferability to unseen domains. Our findings suggest that finetuned GLM retrievers generalize better out of domain, achieving SOTA on two multi-hop benchmarks. On in-domain multi-hop QA datasets they remain comparable to prior work, with promising scaling as parameters and subgraph coverage increase. GNN-based retrievers achieve higher graph coverage with an efficient training setup, whereas the vector-search baseline excels at single-hop datasets.[1]

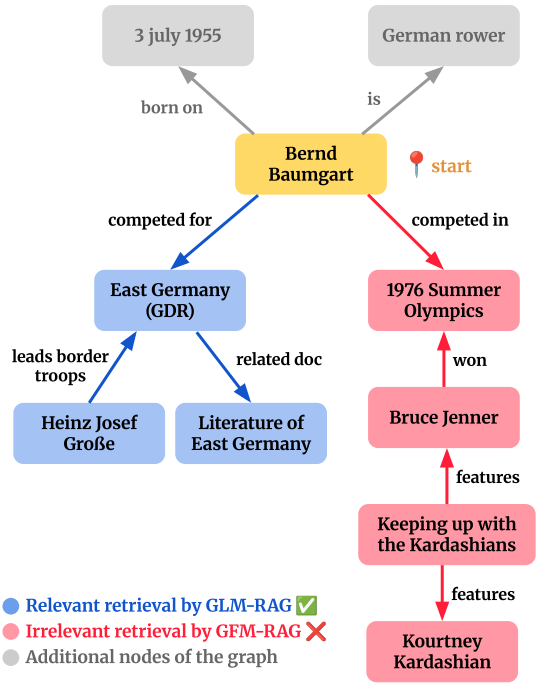


Figure 1: GLM-RAG and GFM-RAG knowledge graph retrieval traces for the query "*What is the three letter abbreviation for the country, which maintains border troops, and claims Bernd Baumgart as a citizen?*" with "*GDR*" being the gold answer. Example chosen from the test set of MuSiQue.

1 Introduction

Language models (LMs) struggle with hallucination in many downstream applications (Zhang et al., 2025b) and often lack knowledge in specialized domains.

Retrieval-augmented generation (RAG) addresses these challenges by retrieving relevant documents and providing them as additional context (Lewis et al., 2020). Thus, domain-specific knowledge can be integrated into an LM's reasoning process, and hallucinations can be reduced by grounding the LM's answer in relevant facts. Although RAG models show strong results in many knowledge-intensive tasks, they lag behind in tasks requiring multi-hop reasoning across multiple documents (Tang and Yang, 2024).

Graphs can naturally capture relationships between documents. Unlike vector search, graph-based retrieval can traverse multiple relationships to incorporate multi-dimensional retrieval chains, enabling stronger cross-document interactions. Hence, graph RAG methods have been introduced to combine the strengths of graph structures and RAG models (Edge et al., 2025).

Current graph RAG methods (Gutiérrez et al., 2024; He et al., 2024; Luo et al., 2025) typically rely on shallow semantic integration, such as sen-

*Shared last authorship

[1] Aleph-Alpha-Research/glm-rag-open

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 4503–4532
November 6-10, 2026 ©2026 Association for Computational Linguistics

tence embeddings, and do not support end-to-end training in a unified graph-text model. Figure 1 shows a knowledge graph (KG) and the retrieval chain of graph RAG models for a complex query: "*What is the three letter abbreviation for the country, which maintains border troops, and claims Bernd Baumgart as a citizen?*".

GFM-RAG (see details in §4.2.1) does not utilize node labels. Consequently, it treats "*East Germany (GDR)*" and "*1976 Summer Olympics*" as equally relevant neighbors of "*Bernd Baumgart*": both are direct neighbors of "*Bernd Baumgart*" and connected via similar relations ("*competed for*" and "*competed in*"). By relying on these structural signals, GFM-RAG retrieves irrelevant entities such as "*Bruce Jenner*" and "*Kourtney Kardashian*".

Considering such shortcomings, we see great potential in unleashing the semantic meaning of text-attributed graphs, by using a model that has a good understanding of the meaning of text and can, at the same time, operate on graphs. Following this motivation, we propose **GLM-RAG**: **G**raph **L**anguage **M**odels for Graph-based **R**etrieval-**A**ugmented **G**eneration. Building on GFM-RAG (Luo et al., 2025), we replace their GNN and sentence-embedding-based retriever with an end-to-end trainable Graph Language Model (GLM; Plenz and Frank, 2024). GLMs adapt pretrained language models into graph transformers while preserving their pretrained parameters, and hence language understanding capabilities (see §3). This allows GLM-RAG to natively process text-attributed graphs on the token level, simultaneously leveraging both structural information and the semantic meaning of entities, relations, and queries. In the above example (cf. also Fig. 1), this enables GLM-RAG to make an informed decision between the two similar neighbors and successfully retrieve the semantically relevant neighborhood.

In addition to introducing GLM-RAG, we systematically assess when graph-enhanced retrieval methods offer benefits. We conduct comprehensive experiments across a range of domains, enabling a rigorous comparison between graph-based methods and a vanilla vector-search approach.

Zero-shot results on out-of-domain (OOD) single-hop datasets reveal that vanilla RAG is sufficient for datasets that require only single-step reasoning. This is in line with recent findings by Xiang et al. (2026). However, in multi-hop settings, we see that the graph-based models outperform vanilla RAG. Furthermore, results on OOD datasets highlight GLM-RAG's strong transferability performance, outperforming GFM-RAG in all settings and achieving new SOTA results on Medical and computer science G-bench (Xiang et al., 2026; Xiao et al., 2025).

Beyond these advantages, our ablations reveal that GLM-RAG's performance scales well with model size, while GFM-RAG plateaus. Combined with the strong OOD performance, this shows GLM-RAG's merit as a general graph RAG foundation model.

Lastly, our analysis of the ranking results reveals that GLM-based retrievers show particular strengths on questions requiring deeper semantic understanding, whereas GNN-based retrievers profit from covering larger parts of the knowledge graph.

Our main contributions are:

1. We propose **GLM-RAG**, a new graph RAG framework powered by a Graph LM-based retriever for stronger integration of text features
2. GLM-RAG shows strong **zero-shot** generalization capabilities, surpassing comparable graph foundation models
3. Our **comparative results** show that vanilla RAG is sufficient for single-hop retrieval tasks, while graph-based methods excel in *multi-hop* settings
4. Ablation studies show that GLM-RAG **scales better** with increasing model size compared to GNN-based retrievers
5. Extensive **analysis** reveals that GNN-based retrievers achieve broader graph coverage but lack the stronger *semantic understanding* of GLM-based retrievers

2 Previous Work

Retrieval-Augmented Generation (RAG). RAG typically relies on dense vector search, where text corpora are split into isolated chunks and indexed via sentence embeddings (Reimers and Gurevych, 2019). Once retrieved, relevant documents are included in an LM's context to help answer questions (Lewis et al., 2020; Karpukhin et al., 2020). While offering a cheaper alternative to finetuning models, RAG models struggle with multi-hop reasoning and complex queries that span across separate texts.

Graph RAG. Graph RAG methods aim to solve more complex multi-document tasks by encoding informational dependencies within a graph. Edge et al. (2025) frame this as a query-focused summarization task, rather than an explicit retrieval task. However, later works use GNN-based retrievers to leverage the graph topology of KGs in multi-hop QA tasks (Mavromatis and Karypis, 2025). Most relevant to our work is GFM-RAG (Luo et al., 2025), a "graph foundation model" powered by a GNN-based retriever, designed to be generalizable to unseen domains without finetuning. GFM-RAG achieved SOTA results on in-domain datasets and showed promising transferability beyond the training data. In recent work, Luo et al. (2026) extended GFM-RAG by including more diverse information in the constructed KG, thus surpassing the previous SOTA.

In this work, we build on GFM-RAG, with the objective of improving its transferability by using a Graph LM (GLM) for retrieval (Plenz and Frank, 2024). Graph LMs are based on pretrained LLMs and therefore combine a strong understanding of (unseen) texts with graph processing capabilities. Combining our approach with Luo et al. (2026) is left for future work.

LMs on graphs. A common approach to encode KGs is to linearize and process them with an LM (Schmitt et al., 2020; Ribeiro et al., 2021; Li et al., 2021; Gao et al., 2024; Yamada and Sugiyama, 2026). While capturing the text features well, such approaches make limited use of the graph structure. Hence, GNNs in combination with often static semantic embeddings are employed (Lin et al., 2019; Malaviya et al., 2020; Yasunaga et al., 2022; Zhao et al., 2023). By slightly adjusting an LM's architecture, while maintaining its pretrained parameters, a deeper interaction between text features and graph structure can be enabled (Plenz and Frank, 2024; Egressy and Stühmer, 2025; Gong et al., 2026; Vajda, 2026). We build on this line of work to improve graph reasoning in graph RAG.

3 Preliminary: Graph Language Models

Graph Language Models (GLMs) bridge language models and graph transformers by converting a pretrained LM into a graph transformer (Plenz and Frank, 2024). This enables the model to leverage pretrained language understanding from the LM while incorporating structural graph reasoning.

This makes GLMs ideal for processing text-attributed graphs, such as knowledge graphs. The method involves two main steps. First, the knowledge (sub)graph undergoes preprocessing: edge and node labels are tokenized individually and connected according to the original graph structure. This transformation allows each triplet to be represented as a token sequence resembling natural text, while still sharing node tokens across multiple triplets, as in a standard graph structure. Second, the LM's self-attention mechanism is modified to use relative positional encodings that capture distances between token pairs within the same triplet, thereby enabling the model to *read triplets as sequential text* – just as a language model would. While these attention patterns, which are typical for LMs, are applied to triplets, the GLM also uses GNN-like attention patterns, which capture the overall graph structure. Together, these adaptations yield a model that is able to read a labeled graph much like contiguous language, thereby achieving strong **(i)** language understanding abilities and simultaneously, **(ii)** graph-based reasoning abilities that are well-suited for graph RAG. We refer to the original publications for more details on the method as well as experimental validation.

4 GLM-RAG

Building on the ability of the GLM to jointly reason over text *and* graph-structured data, we propose **GLM-RAG**, a RAG architecture that relies on a **GLM-based retriever**.

We hypothesize that a GLM-based retriever achieves better retrieval quality and, consequently, better QA performance due to its better understanding of text features in the graph, compared to a GNN-based retriever, which can only utilize the graph structure with shallow text integration.

We build on the GFM-RAG framework of Luo et al. (2025), adopting their KG and QA dataset construction stages, as well as their document ranking and answer generation mechanisms. Our contribution is a **GLM-based retriever** (§4.2.2) that replaces the original KG retrieval component. Crucially, this setup yields a controlled environment that allows us to assess a GLM-based versus a GNN-based retriever. Please refer to Apx. B for more implementation details.

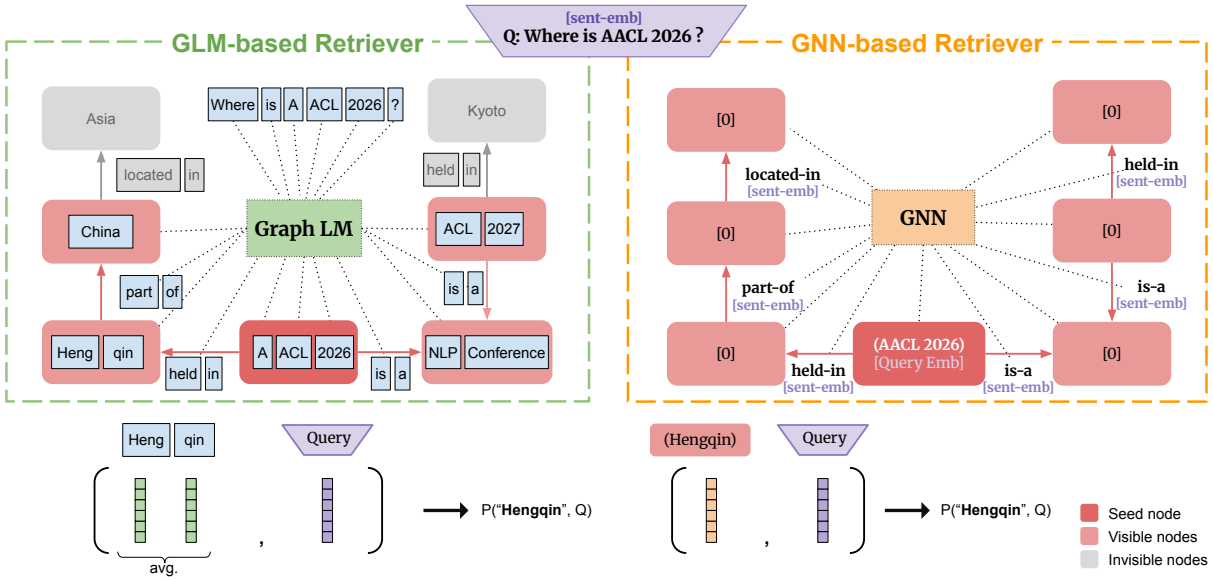


Figure 2: Differences between how GNN- and GLM-based retrievers reason over the KG for the query "*Where is AACL 2026?*". GLM-based retriever (left) tokenizes the nodes, relations, and the query with the Graph LM's encoder. The tokenized input is passed through the graph LM to produce embeddings for each token. The token embeddings corresponding to each node are aggregated to produce the final node representation, which is used for computing the relevance scores of each entity in the extracted subgraph. GNN-based retriever (right) calculates the query and relation embeddings via a sentence embedding model. Only the seed entities are initialized with the query embedding, and all remaining nodes with the zero vector. The GNN iteratively propagates messages along edges using message passing over the full graph, producing a score distribution over all nodes.

4.1 KG & QA Dataset Construction

The pipeline starts from a document corpus, and constructs a corresponding entity-level KG via LLM-based named entity recognition (NER), so that a document such as "*AACL 2026 is held in Hengqin, China.*" is represented through the triplets `[(AACL_2026, held_in, Hengqin), (Hengqin, part_of, China)]` within the KG.

The queries undergo a similar transformation, so that a question "*Where is AACL 2026?*" is represented through a seed entity "`AACL_2026`" as a starting point within the KG. There may be multiple seed entities depending on the complexity of the query.

4.2 KG Retrieval

4.2.1 GNN-based Retriever

The main building block of GFM-RAG is the query-dependent GNN retriever, whose job it is to rank the entities of the constructed KG by relevance to a given input query. The message passing paradigm (Gilmer et al., 2017) is used to propagate information across the graph (see right side of Fig. 2).

While seed entities are initialized with the query embeddings, all other nodes are initialized with zero vectors. By contrast, relations are initialized with their sentence embeddings. Consequently, the resulting node representations from the GNN are driven by their relative graph distance to the seed entities rather than their semantic features. Node texts are used exclusively to identify seed entities and play no further role in the GNN retriever. Given this shortcoming, we also explore a variant of GFM-RAG, GFM-RAG+, where all nodes are initialized with their sentence embeddings (see Sec. 5.1).

At the final layer, the per-node representations are concatenated with the query embedding and scored by an MLP to find the most relevant nodes. Luo et al. (2025) pretrain the GNN retriever on KG completion and finetune it on three Wikipedia-based multi-hop QA datasets.

4.2.2 GLM-based Retriever

Our proposed GLM-based retriever aims for a graph RAG approach that maximizes the potential of text-attributed graphs.

The inner workings of the GLM retriever fundamentally differ from those of the GNN-based one. Most notably, the GNN-based retriever does not make use of the rich textual features in the graph and instead relies on the query and the raw graph structure, whereas our GLM retriever fully utilizes

the semantics in the graph through the GLM's text encoding abilities (§3).

The GLM retriever extracts a local subgraph around the seed entities, converts it into a sequence of textual triplets, and encodes the individually tokenized edge and node representations through a graph encoder with structure-aware relative positions (see left side of Fig. 2). Nodes and relations are tokenized and embedded using the underlying LM's own tokenizer and token embedding layer. The tokens are passed through the LM's layers, using attention masking to encode the graph structure. This stands in contrast to the GNN retriever, where only the seed entities are initialized with non-zero embeddings. As a result, the query-graph interaction happens at a much earlier stage, namely when the question text is fused with the extracted subgraph structure inside the GLM's first attention layer. The resulting per-entity embeddings are element-wise multiplied with a projected question vector and passed through a scoring head.

We finetune the GLM-based retriever on three Wikipedia-based multi-hop QA datasets (§5).

4.3 Ranking & Answer Generation

After the ranking is completed, the most relevant entities are mapped back to their original documents, thereby converting the entity ranking to a document ranking. Finally, an LM generates the answer from the query and the top-k documents.

5 Experiments

In this section, we introduce our experimental setup and report the results of the following experiments:

1. We finetune GLM- and GNN-based retrievers on three Wikipedia-based multi-hop question-answering (QA) datasets and evaluate retrieval (§5.2) and QA performance (§5.3).

2. We test the finetuned GLM-based retrievers transferability on 11 OOD datasets (§5.4).

Further ablation studies (§C) and a breakdown of the retrieval performance (§D.1) per question complexity can be found in the Appendix.

5.1 Experimental Setup

Models

We choose `all-mpnet-base-v2` as the sentence embedding model for GFM-RAG. Following Plenz and Frank (2024), we initialize our GLM retriever with `T5`-large model weights, which we test in different sizes (as shown in Section 5.5). We use `gpt-4o-mini` as the LM for all text generation.

Datasets

We finetune the models on the train-splits of three Wikipedia-based multi-hop QA datasets: HotpotQA (Yang et al., 2018), 2WikiMultihopQA (2Wiki) (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). Together, these splits yield 277k question-document pairs for training[2].

To evaluate the transferability capabilities of the GLM retriever, we follow Luo et al. (2025), who chose seven transferability datasets from various domains: TechQA (Castelli et al., 2020), eManual (Nandy et al., 2021), and DelucionQA (Sadat et al., 2023) have a customer support focus, MS MARCO (Nguyen et al., 2016), HAGRID (Kamalloo et al., 2023), and ExpertQA (Malaviya et al., 2024) contain general knowledge questions, while PubMedQA (Jin et al., 2019) requires biomedical knowledge. However, since these datasets do not require multi-hop reasoning, they can be solved effectively using vanilla RAG. Therefore, we also include several multi-hop benchmarks: MultiHop-RAG (Tang and Yang, 2024), a dataset based on English news articles, G-Bench Novel and Medical (Xiang et al., 2026) and G-Bench Computer Science (CS) (Xiao et al., 2025). Please refer to Apx. A for dataset statistics.

Baselines

To contextualize the results of our proposed GLM-RAG method, we compare it to the following baselines:

a. A **vanilla RAG** baseline encodes the documents as well as the queries with the `all-mpnet-base-v2` embedding model, and retrieves the top-5 documents according to their cosine similarity. This baseline tests the need for graph RAG methods.

b. As a strong comparative baseline, we use the latest release of **GFM-RAG** on Hugging Face, which uses an identical graph RAG pipeline.

c. A **finetuned-only** version of GFM-RAG that we refer to as **GFM-RAG***. We opt for a finetuned-only version of GFM-RAG to test the effectiveness of the pretraining stage. This is a more comparable baseline to our GLM-RAG, considering we also only perform finetuning.

[2] In contrast, Luo et al. (2025) use 60k Q-Doc pairs for their results. However, in their follow-up work, Luo et al. (2026) publish new results trained with 277k Q-Doc pairs.

d. A finetuned-only version of GFM-RAG where nodes are initialized with text embeddings rather than zero vectors. We refer to this as **GFM-RAG+**. Compared to GLM-RAG, this approach also utilizes node semantics, though only via static sentence embeddings.

e. For QA performance we additionally test a **no-context baseline** that evaluates the LM's capability to answer questions without any retrieved context.

Metrics

We use Recall@2 at the document level to measure retrieval performance. We evaluate document-level rather than entity-level performance, since the correct retrieval of documents directly affects the downstream QA performance. For QA performance, we report exact match (EM). All metrics are established in prior work, enabling a direct comparison. We also report Recall@5 in Apx. D and F1 scores in Apx. E. For G-Bench, we follow standard evaluation practice for each benchmark and report evidence recall for retrieval and accuracy (answer correctness and answer score) for QA performance. We use `gpt-4o-mini` for all LLM-as-a-judge calls. Following prior work, we report one run per experiment due to computational constraints and provide significance tests in Appx H.

5.2 Retrieval Performance

Figure 3 shows retrieval performance on the test sets of the three Wikipedia-based datasets. All graph-enhanced methods outperform the RAG baseline, highlighting the contribution of graph RAG methods in multi-hop tasks. GFM-RAG and GFM-RAG* (i.e., with and without pretraining) show near-identical performance on most datasets, calling into question the need for costly pretraining. Hence, we also only finetune our GLM retriever.

Overall, GFM-RAG+ and GLM-RAG perform the best, showing the advantage of utilizing node semantics. On HotpotQA and 2Wiki, GFM-RAG+ slightly outperforms GLM-RAG, while GLM-RAG achieves the best result on MuSiQue. This confirms GLM-RAG's competitive in-domain performance, despite being restricted to smaller subgraphs.

Table 17 shows the performance of additional graph RAG methods, including new SOTA results by Luo et al. (2026) in concurrent work. Their main innovation is to extend GFM-RAG by including more information in the KG-indexing step. This is orthogonal to our work and is compatible with our proposed GLM retriever. Combining both approaches is beyond the scope of this paper and is left for future work.

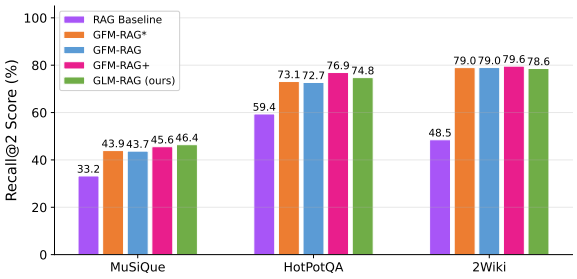


Figure 3: Retrieval performance (Recall@2) on in-domain Wikipedia datasets.

5.3 QA Performance

A similar trend can be observed in downstream QA performance in Figure 4. The no-context and RAG baselines get the lowest scores, showing the value of the retrieved context as well as graph-enhanced retrievers. GFM-RAG and GFM-RAG* perform comparably across all three datasets, with a maximum difference of 1.7 points on HotpotQA, confirming that the effect of pretraining is negligible.

The overall performance trends of GLM-RAG and the baselines are comparable to the results in retrieval. The only difference is that GLM-RAG achieves the best EM score on HotpotQA. QA results of further methods are shown in Table 20.

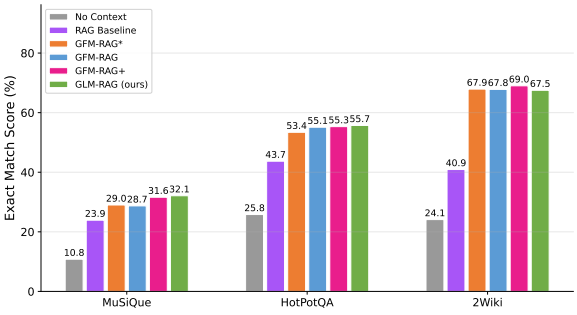


Figure 4: QA performance (exact match) on in-domain Wikipedia datasets.

5.4 Transferability

While GLM-RAG's in-domain performance is not significantly better than that of GFM-RAG+, transferability experiments reveal better generalization to OOD datasets.

In the following, we first train our models on only one of the Wikipedia datasets, and examine its transfer to the remaining Wikipedia datasets (§5.4.1). Then we test the fully-trained models on

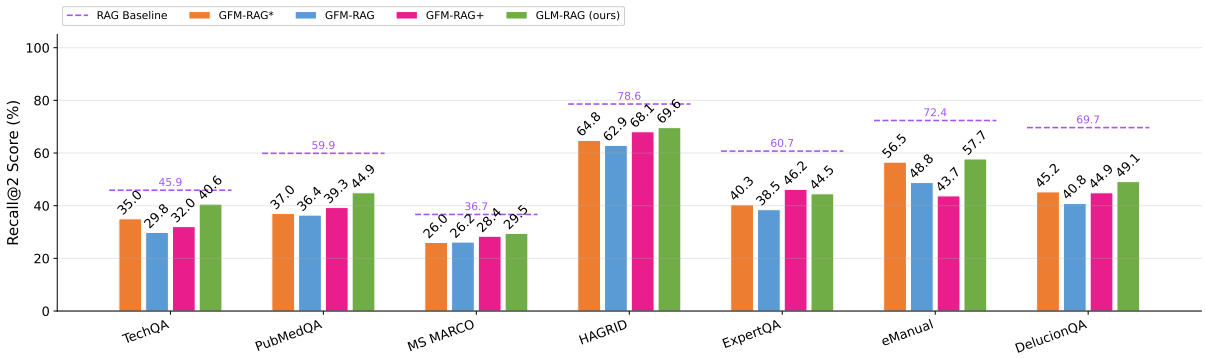


Figure 5: Retrieval performance (Recall@2) on single-hop out-of-domain datasets.

seven single-hop OOD datasets (§5.4), and four multi-hop OOD datasets (§5.4.3).

Table 1: Generalization abilities of GLM-RAG and GFM-RAG(*,+) when finetuned on one Wikipedia dataset.

		Recall@2		
Train Set	**Test Set**	**GLM-RAG**	**GFM-RAG***	**GFM-RAG+**
MuSiQue	HotpotQA	**65.7**	59.4	60.3
	MuSiQue	**44.6**	40.8	41.7
	2Wiki	**74.0**	71.3	72.6
HotpotQA	HotpotQA	74.2	73.5	**77.0**
	MuSiQue	**45.4**	42.6	43.6
	2Wiki	**76.4**	75.6	75.7
2Wiki	HotpotQA	**64.3**	57.7	57.8
	MuSiQue	**43.3**	39.8	39.1
	2Wiki	77.7	78.9	**79.6**

5.4.1 Transferability on Wikipedia Datasets

We train GLM-RAG, GFM-RAG* and GFM-RAG+ on each Wikipedia dataset and investigate transferability to the other two Wikipedia datasets.

Table 1 shows that in all cases, GLM-RAG performs better in **out-of-domain** settings than the GFM-RAG variants, even surpassing GFM-RAG+ (i.e., GFM-RAG with additional node embeddings). For example, when trained on HotpotQA, GLM-RAG outperforms GFM-RAG* by 2.8 points and GFM-RAG+ by 1.8 points on MuSiQue.

On MuSiQue, GLM-RAG is consistently better than both models, whereas GFM-RAG+ performs better **in-domain** on HotpotQA, and both GFM-RAG variants on 2Wiki.

5.4.2 Transferability on Single-hop Datasets

Motivated by previous results, we test the models' generalization abilities to unseen domains on seven single-hop datasets.

Figure 5 shows that GLM-RAG consistently outperforms GFM-RAG models, except for Recall@2 on ExpertQA. Considering that GFM-RAG is a foundation model trained to generalize well, this indicates strong transferability of GLM-RAG. We hypothesize that this is largely enabled by the GLM's deep integration of the KG's rich semantics.

Still, it is important to note that a vanilla **RAG baseline** outperforms all tested graph RAG models on all datasets. While this may seem surprising, it is due to the nature of the datasets: Single-hop datasets contain direct questions that do not require multi-hop reasoning capabilities, so the RAG's "simpler" similarity metric is often sufficient to find relevant documents. Graph RAG methods, by contrast, show their strengths in multi-hop datasets, where retrieving documents that are similar to the question is not sufficient to answer compositional questions.

5.4.3 Transferability on Multi-hop Datasets

We challenge the models by testing their retrieval and QA performance for zero-shot transferability on MultiHop-RAG (Tang and Yang, 2024), G-Bench Novel, G-Bench Medical (Xiang et al., 2026), and G-Bench CS (Xiao et al., 2025), all requiring multi-hop reasoning. Here, we also include the recently introduced SOTA method, G-Reasoner (Luo et al., 2026).

Table 2 shows the retrieval and QA results for all methods. GLM-RAG's retrieval performance excels in this multi-hop zero-shot setting, surpassing the RAG baseline, GFM-RAG variants, and G-Reasoner by 20 points in Recall@2 on MultiHop-RAG. The same holds for the three G-Bench benchmarks, where GLM-RAG outperforms all baselines including G-Reasoner. At the time of writing, GLM-RAG achieves SOTA results for the Medical[3] and CS[4] benchmarks, and ranks second on the

[3] https://graphrag-bench.github.io/
[4] https://deep-polyu.github.io/RAG/

Table 2: Retrieval (Recall@2 and evidence recall) and QA (accuracy) performance on multi-hop OOD datasets.

	MultiHop-RAG	G-Bench (Novel)		G-Bench (Medical)		G-Bench (CS)
Method	R@2	Recall	ACC	Recall	ACC	ACC
RAG	32.5	55.9	47.9	75.1	61.0	71.7
GFM-RAG	34.1	75.9	58.6	82.2	72.2	72.1
GFM-RAG+	39.0	86.1	60.6	93.3	75.7	76.1
G-Reasoner	34.9	87.7	58.9	93.8	73.3	73.9
GLM-RAG	**60.0**	**88.0**	**61.9**	**94.6**	**76.9**	**76.6**

Novel[3] benchmark. AutoPrunedRetriever (Wang et al., 2026), a graph RAG method that persists the minimal reasoning subgraph built for earlier questions and incrementally extends it for later ones, is the current SOTA method for the Novel benchmark. Integrating this pruning idea into GLM-RAG is left for future work.

This challenging test case underlines GLM-RAG's merit as a graph foundation model with strong capabilities for zero-shot multi-hop settings.

To conclude the main experiments, our findings suggest that even though GLM-RAG outperforms other graph-enhanced methods on the transfer datasets, vanilla RAG is sufficient for single-hop questions. On multi-hop OOD datasets, however, GLM-RAG clearly outperforms both the vanilla RAG baseline and all GFM-RAG variants.

5.5 Ablations

We aim to show that GLM-RAG's transferability gains stem from its deeper integration of textual features rather than input filtering or model scale. To test this, the following section ablates the two primary differences between GLM-RAG and GFM-RAG variants: graph input and model size. For these ablations, we train all models on a smaller subset of the training data with 60k instances, and use `T5`-base as the GLM's base model.

Effect of Subgraph Restriction

One of the key differences between GLM-RAG and GFM-RAG is the restricted graph visibility of GLM-RAG due to context size limitations. Here, we test whether this restriction acts as an information bottleneck or a noise filter for retrieval performance. We evaluate this by adjusting the number of visible triplets in the 2-hop neighborhood subgraph in GLM-RAG as well as GFM-RAG*.

Table 3 shows that GFM-RAG*'s performance is negatively affected by this restriction. Moreover, as we tighten the restriction for both models, the recall drops monotonically. This supports our claim that GLM-RAG's restricted visibility **limits** rather than aids its performance, and that relaxing it would improve results.

Table 3: Retrieval results (Avg. Recall@2) with restricted GFM-RAG* and GLM-RAG. Recall results are averaged over HotpotQA, MuSiQue and 2Wiki test sets.

	GFM-RAG*	GLM-RAG
unrestricted	**63.5**	-
600 triplets	60.4	**63.3**
500 triplets	59.5	62.8
400 triplets	59.5	61.8
300 triplets	58.6	60.8

Effect of Model Size

One might also argue that the gains of GLM-RAG are due to its higher parameter count, the GFM-RAG* and GLM retrievers having 8M and 336M parameters, respectively. To rule out this hypothesis, we compare the models in a **capacity-matched** setting by increasing the hidden dimensions of the GNN-based retrievers. This results in three capacity-matched pairs, shown in Table 18.

Figure 6 shows the scaling trend of GNN- and GLM-based retrievers with increasing parameter count. GFM-RAG* does not show any performance gains from more parameters, and GFM-RAG+'s upward trend only holds for in-domain datasets (see also Figure 11 and Table 19). In contrast, GLM-RAG shows an upward trend in both settings, showing that its transferability advantage holds even when controlling for model size.

This finding is in line with work on the overfitting and oversmoothing problems of GNNs (Li et al., 2018; Rong et al., 2020; Oono and Suzuki, 2020; Chen et al., 2022), indicating that GNN scaling lags behind that of transformer-based models (Kaplan et al., 2020; Dehghani et al., 2023).

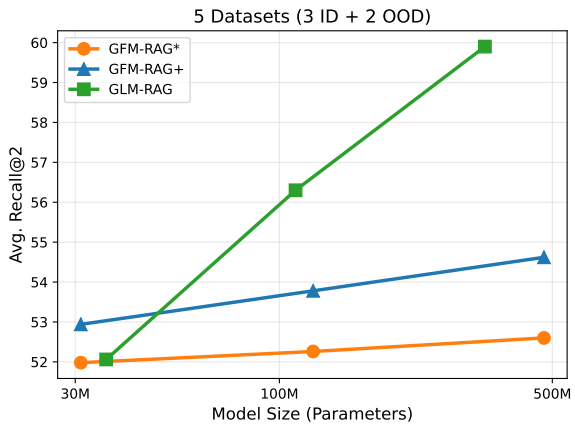


Figure 6: Model scaling comparison of GLM-RAG vs capacity-matched GFM-RAG* and GFM-RAG+.

To conclude, we showed that GLM-RAG's transferability claims still hold under matched visibility and capacity, suggesting that the generalizability gains stem from utilizing the existing large-scale pretrained LMs. The scalability of GLMs is an additional benefit that indicates potential for further improvements from switching to larger and more recent LMs as the GLM's base.

6 Analysis

After evaluating the in- and out-of-domain retrieval and QA quality of the compared methods, we further investigate the strengths and weaknesses of each graph-based retriever model. Although these properties are not a proxy for better retrieval, we analyze them to better understand the characteristics of each model's retrieval strategy. Our analysis shows that the GLM retriever relies more on semantic knowledge to retrieve the right documents, whereas GNN-based retrievers rely more on structural signals. Appendix G shows the results and calculations for these measures.

6.1 Similarity Measures

First, we test the hypothesis that the GLM retriever utilizes semantic features more effectively than the other models. For this analysis, we compute the **semantic similarity** between the question and the retrieved entities. Table 4 shows that GLM-RAG and GFM-RAG+ consistently retrieve more semantically similar entities than GFM-RAG and GFM-RAG* across all 11 datasets.

The GLM retriever's strength is especially evident on the multi-hop datasets, having better semantic understanding of the constructed graphs.

Table 4: Comparison of cosine similarity between the question and the retrieved entities.

Models	GLM-RAG	GFM-RAG*	GFM-RAG	GFM-RAG+
Multi-Hop Datasets				
MuSiQue	0.309	0.283	0.286	0.297
HotpotQA	0.312	0.292	0.293	0.298
2Wiki	0.285	0.262	0.265	0.268
MultiHop-RAG	0.245	0.209	0.212	0.241
Single-Hop Datasets				
TechQA	0.255	0.234	0.228	0.245
PubMedQA	0.336	0.298	0.289	0.314
MS MARCO	0.294	0.280	0.280	0.314
HAGRID	0.345	0.329	0.323	0.355
ExpertQA	0.284	0.249	0.235	0.286
eManual	0.276	0.234	0.221	0.286
DelucionQA	0.294	0.287	0.286	0.313

6.2 Distance Measures

To compare the **graph coverage** for both models, we compute the shortest path from each retrieved entity to any question entity and aggregate it per sample and per dataset. The results in Table 9 show that the GNN-based retrievers reach further on average compared to the GLM-based retriever. Although this is expected due to their architectural differences, it suggests that a strength of GNN-based retrievers lies in their wider coverage of the graph.

7 Conclusion

In this paper, we present an alternative approach to handling multi-hop questions in graph RAG settings by training a GLM-based retriever. Our experiments show that GLM-RAG is a more generalizable method with better scalability, in line with work on building graph foundation models. By integrating the semantic knowledge contained in a text-attributed graph, a GLM-based retriever can better understand and select relevant documents, relying more on assessing the similarity between document entities and the question, whereas GNN-based retrievers rely more on structural signals. Our extensive comparison of vanilla RAG, GNN-based and GLM-based retrievers reveals that **(i)** for single-hop questions, which do not require multi-hop reasoning, a vanilla RAG baseline is sufficient, **(ii)** for in-domain multi-hop questions, a finetuned GFM-RAG model initialized with node embeddings (GFM-RAG+) performs competitively with GLM-RAG, and **(iii)** for zero-shot multi-hop questions, GLM-RAG shows consistently better transferability capabilities compared to its GNN-based counterparts.

Limitations

We build on GFM-RAG (Luo et al., 2025), which was the SOTA at the time of our experiments. Recently, Luo et al. (2026) extended GFM-RAG's KG indexing to capture more diverse information, establishing a new SOTA. Although we do not combine our GLM retriever with this indexing strategy, we expect the two approaches to be complementary, making their integration a promising direction for future research.

We have tested our approach with base LMs of up to `T5`-large (770M parameters, of which the 336M-parameter encoder is used). Based on our scaling experiments, we see great potential in using larger base LMs, which we leave for future work.

Finally, being transformer-based, our GLM-based retriever is computationally more demanding than a GNN-based retriever. Consequently, we restrict the size of the processed subgraphs to enable more efficient training. Future work could scale up GLM-based retrievers to combine their strong semantic capabilities with broader graph coverage.

Acknowledgments

We sincerely thank Fabien Benureau for all the contributions he made to the project.

References

Aijun Bai, Rolf Jagerman, Zhen Qin, Le Yan, Pratyush Kar, Bing-Rong Lin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2023. Regression compatible listwise objectives for calibrated ranking with binary relevance. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 4502–4508, New York, NY, USA. Association for Computing Machinery.

Vittorio Castelli, Rishav Chakravarti, Saswati Dana, Anthony Ferritto, Radu Florian, Martin Franz, Dinesh Garg, Dinesh Khandelwal, Scott McCarley, Michael McCawley, Mohamed Nasr, Lin Pan, Cezar Pendus, John Pitrelli, Saurabh Pujar, Salim Roukos, Andrzej Sakrajda, Avi Sil, Rosario Uceda-Sosa, Todd Ward, and Rong Zhang. 2020. The TechQA dataset. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1269–1278, Online. Association for Computational Linguistics.

Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. 2022. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2769–2781.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd Van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby. 2023. Scaling vision transformers to 22 billion parameters. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7480–7512. PMLR.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph rag approach to query-focused summarization.

Beni Egressy and Jan Stühmer. 2025. Set-LLM: A permutation-invariant LLM. *Advances in Neural Information Processing Systems*, 38:62798–62834.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. 2024. Towards foundation models for knowledge graph reasoning. In *The Twelfth International Conference on Learning Representations*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. Pmlr.

Haisong Gong, Zhibo Liu, Qiang Liu, Shu Wu, and Liang Wang. 2026. NAG: A unified native architecture for encoder-free text-graph modeling in language models.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and fast retrieval-augmented generation.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. Curran Associates, Inc.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to memory:

Non-parametric continual learning for large language models. In *Forty-second International Conference on Machine Learning*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. 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 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Ehsan Kamalloo, Aref Jafari, Xinyu Zhang, Nandan Thakur, and Jimmy Lin. 2023. HAGRID: A human-llm collaborative dataset for generative information-seeking with attribution.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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 *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021. Few-shot knowledge graph-to-text generation with pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1558–1568, Online. Association for Computational Linguistics.

Mufei Li, Siqi Miao, and Pan Li. 2025. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Qimai Li, Zhichao Han, and Xiao-ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Lei Liang, Zhongpu Bo, Zhengke Gui, Zhongshu Zhu, Ling Zhong, Peilong Zhao, Mengshu Sun, Zhiqiang Zhang, Jun Zhou, Wenguang Chen, Wen Zhang, and Huajun Chen. 2025. KAG: Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, page 334–343, New York, NY, USA. Association for Computing Machinery.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. 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 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. 2025. GFM-RAG: Graph foundation model for retrieval augmented generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Linhao Luo, Zicheng Zhao, Junnan Liu, Zhangchi Qiu, Junnan Dong, Serge Panev, Chen Gong, Thuy-Trang Vu, Gholamreza Haffari, Dinh Phung, Alan Wee-Chung Liew, and Shirui Pan. 2026. G-reasoner: Foundation models for unified reasoning over graph-structured knowledge. In *The Fourteenth International Conference on Learning Representations*.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

Chaitanya Malaviya, Subin Lee, Sihao Chen, Elizabeth Sieber, Mark Yatskar, and Dan Roth. 2024. ExpertQA: Expert-curated questions and attributed answers. 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 3025–3045, Mexico City, Mexico. Association for Computational Linguistics.

Costas Mavromatis and George Karypis. 2025. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16682–16699, Vienna, Austria. Association for Computational Linguistics.

Abhilash Nandy, Soumya Sharma, Shubham Maddhashiya, Kapil Sachdeva, Pawan Goyal, and NIloy Ganguly. 2021. Question answering over electronic devices: A new benchmark dataset and a multi-task learning based QA framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4600–4609, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Kenta Oono and Taiji Suzuki. 2020. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*.

Moritz Plenz and Anette Frank. 2024. Graph language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4477–4494, Bangkok, Thailand. Association for Computational Linguistics.

Moritz Plenz, Juri Opitz, Philipp Heinisch, Philipp Cimiano, and Anette Frank. 2023. Similarity-weighted construction of contextualized commonsense knowledge graphs for knowledge-intense argumentation tasks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6130–6158, Toronto, Canada. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2021. Investigating pretrained language models for graph-to-text generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 211–227, Online. Association for Computational Linguistics.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.

Mobashir Sadat, Zhengyu Zhou, Lukas Lange, Jun Araki, Arsalan Gundroo, Bingqing Wang, Rakesh Menon, Md Parvez, and Zhe Feng. 2023. DelucionQA: Detecting hallucinations in domain-specific question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 822–835, Singapore. Association for Computational Linguistics.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

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

Martin Schmitt, Sahand Sharifzadeh, Volker Tresp, and Hinrich Schütze. 2020. An unsupervised joint system for text generation from knowledge graphs and semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7117–7130, Online. Association for Computational Linguistics.

Yixuan Tang and Yi Yang. 2024. Multihop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *First Conference on Language Modeling*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Dario Vajda. 2026. Teaching LLMs to see graphs: Unifying text and structural reasoning.

Ning Wang, Kuanyan Zhu, Daniel Yuehwoon Yee, Yitang Gao, Shiying Huang, Zirun Xu, and Sainyam Galhotra. 2026. Pruning minimal reasoning graphs for efficient retrieval-augmented generation.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2026. When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation. In *The Fourteenth International Conference on Learning Representations*.

Yilin Xiao, Junnan Dong, Chuang Zhou, Su Dong, Qian wen Zhang, Di Yin, Xing Sun, and Xiao Huang. 2025. GraphRAG-Bench: Challenging domain-specific reasoning for evaluating graph retrieval-augmented generation.

Masatsugu Yamada and Mahito Sugiyama. 2026. When graph language models go beyond memorization.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. In *Advances in Neural Information Processing Systems*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025a. Qwen3 embedding: Advancing text embedding and reranking through foundation models.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2025b. Siren's song in the AI ocean: A survey on hallucination in large language models. *Computational Linguistics*, 51(4):1373–1418.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*.

Appendix

A Details on Datasets

Table 10 shows statistics of the training datasets and provides a high-level overview of the constructed KGs, including the total number of queries, documents, entities, relations, and triplets. The latest version of GFM-RAG in (Luo et al., 2026) uses the full training dataset. In this work, we use a validation set for hyperparameter tuning and therefore train on 1k fewer queries per dataset. As a result, we use a total of 273,830 queries and 2,208,920 documents for training. We also provide statistics for the test datasets in Table 11. The in-domain test sets each contain 1k queries, whereas the out-of-domain test sets contain varying numbers of queries, ranging from 132 (eManual) to 2,450 (PubMedQA).

Tables 12 and 13 provide more detailed statistics about the constructed KGs using the following metrics:

The **average degree** is the mean number of edges incident to each node within the graph.

The **density** measures the ratio of actual edges to the total number of possible edges in a simple graph. Given the high values of $|V|$ in these datasets, density values near 0.0001 indicate highly sparse graphs, which is characteristic of large-scale knowledge bases where only specific and meaningful relations exist.

The **number of components** (# Components) indicates the count of maximal subgraphs in which any two vertices are connected to each other by paths, but which are disconnected from the rest of the graph. It reflects the level of fragmentation within the dataset's knowledge structure.

The **largest connected component** (CC in %) is the ratio of nodes in the graph's largest connected subgraph to the total number of nodes. The high percentages for the in-domain graphs (approximately 99.5%) indicate that the vast majority of the knowledge space is reachable through path-based traversal, whereas some of the smaller out-of-domain graphs are more fragmented (e.g., 90.0% for ExpertQA).

B Supplementary Implementation Details

Table 14 shows the implementation and training settings of the GFM-RAG, GFM-RAG*, GFM-RAG+, and GLM-RAG models. To highlight the most important points once again:

- They share the **KG-index construction** stage, since we use pre-constructed KGs for all datasets except MultiHop-RAG. Only for this dataset is `Qwen3-8B` used for OpenIE.
- The graph is represented via relative position encoding with additional buckets, depending on the setting. The default setting is `global` with `FullyConnected` enabled, resulting in three additional buckets.
- After hyperparameter search, the loss weights were set to 0.44 for the binary cross-entropy (BCE) loss and 0.56 for the list cross-entropy (ListCE) loss in the GLM retriever.

- The GLM retriever uses differential learning rates: 5e-4 for the head (`entity_scorer`, `question_proj`) and 1e-4 for the T5 backbone.
- The GLM retriever is trained for 2 epochs with a batch size of 2, whereas the GNN retriever is trained for 5 epochs with a batch size of 4. However, the batch size was also decreased to 2 for GFM-RAG+ due to the increased memory usage caused by including node embeddings.

B.1 GFM-RAG+ Details

We tested two ways to implement **GFM-RAG+** and adopt the better-performing one as a stronger baseline. Both initialize nodes with entity embeddings; however, they differ in how the query is injected.

The first, "only seed entities get query", multiplies only the seed entities by the query embedding. The second, "all entities get query", initializes every node as $query_emb * node_emb$, contextualizing all entities with the query. This mirrors the `FullyConnected` mode of the GLM retriever, where entities also interact with the query.

In both variants we use the same embedding model for entities, queries, and relations, and the final node features combine the entity and query embeddings. GFM-RAG+ is therefore a query-dependent GNN, like GFM-RAG. As shown in Table 5, contextualizing all entities outperforms seeding alone on both recall metrics, so we use it as our GFM-RAG+ baseline.

Table 5: Different implementations of GFM-RAG+

GFM-RAG+ Methods	Recall@2	Recall@5
Only seed entities get query	65.7	79.9
All entities get query	**67.4**	**81.5**

B.2 GLM-RAG Details

The differences between the models are largely explained in Section 4.2; however, we explain further implementation details for **GLM-RAG**:

Subgraph Selection for Scalability: Due to its transformer-based architecture, the GLM retriever exhibits higher computational overhead than the GNN retriever, making it infeasible to encode an entire large-scale graph simultaneously. To address this GPU bottleneck, the GLM retriever employs a subgraph selection strategy that scores local neighborhoods rather than the global graph.

We control the selection "budget" using two hyperparameters: the number of hops (`max_hops`) and the maximum number of triplets (`max_triplets`). An analysis of the interaction between these parameters is provided in Section C. Based on these results, we set `max_hops=2` and `max_triplets=600`.

While this means that the GNN processes a larger portion of the graph, we expect that the GLM's more sophisticated ranking mechanism compensates for this restricted view.

Hop Prioritization: The seed entities are extracted from the queries during the QA dataset construction process, and they serve as the starting points in the graph, similar to Plenz et al. (2023). If the `max_hops` neighborhoods around the seed entities contain more triplets than `max_triplets`, we sub-sample the neighborhoods to satisfy the budget constraint. We first prioritize triplets that directly connect pairs of seed entities. The remaining quota is then filled using a randomized breadth-first search starting from the seed entities.

Adjustment of the Losses: For GFM-RAG, Luo et al. (2025) employ a weighted combination of BCE and sigmoid-based ListCE (ranking) loss. Since the GLM operates only on a selected subgraph rather than the full graph, the majority of entities remain unscored with a logit of zero. In the sigmoid-based ListCE formulation (Bai et al., 2023), these unscored entities still contribute to the partition function, where each zero logit contributes $\sigma(0) = 0.5$. For large entity sets (e.g., $N \approx 42{,}000$), this results in a large accumulated constant contribution ($\approx 21{,}000$) in the denominator, which overwhelms the signal from positive samples and substantially weakens the learning signal. To address this issue, we mask unscored entities and restrict the normalization term only to the selected subgraph, ensuring that the model optimizes over the ranking of observed entities.

$$\mathcal{L}_{\text{ListCE}} = -\log\left(\frac{\sigma(s_{pos})}{\sum_{i\in\mathcal{V}_s}\sigma(s_i) + \sum_{j\in\mathcal{V}_u}\sigma(s_j)}\right)$$

where $\mathcal{V}_s$ and $\mathcal{V}_u$ denote the scored (subgraph) and unscored entities, respectively. In GLM-RAG, the second sum over $\mathcal{V}_u$ is masked out.

Computational Costs: GFM-RAG* is trained on 8 NVIDIA A100 GPUs (80GB) with 7 hours of supervised finetuning, whereas GLM-RAG `[t5-large, max_triplets=600]` is trained on 16 GPUs with 20 hours of supervised finetuning. The total estimated computational cost for all experiments is approximately 370 GPU hours.

We recognize that GLM-RAG has higher inference latency and memory usage than its GNN-based counterparts. Table 6 shows the average latency and peak GPU memory usage during retrieval on the 2Wiki dataset. While optimization could potentially reduce the inference latency, we believe that the latency is acceptable for complex RAG tasks.

Table 6: Latency and memory usage comparison during retrieval.

	Latency in ms	Allocated memory in GiB
GLM-RAG	692.3	6.8
GFM-RAG*	18.6	1.3

C Extensive Ablations

Figure 7 shows how both the GFM-RAG* and GLM-RAG models scale with increasing amounts of training data. Following this trend, and in order to reduce computational costs, we use a smaller subset of the data consisting of approximately 60k queries and 700k documents, instead of approximately 277k queries and 2.2M documents, for the following ablation analyses:

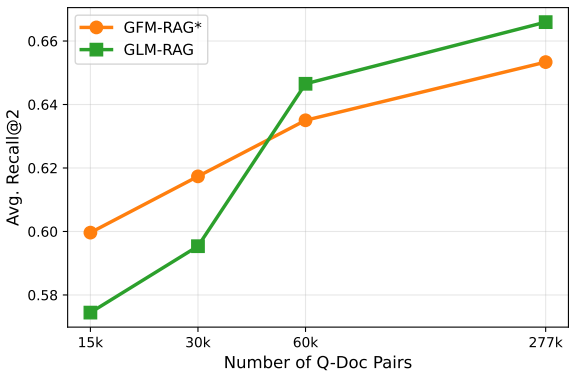


Figure 7: Data Scaling: GFM-RAG* vs GLM-RAG with increasing training data.

In the ablations, the default GLM was initialized with **T5-base**, used **mean** embedding aggregation, and extracted a subgraph from the **2-hop** neighborhood containing up to **600 triplets**.

The query interaction was performed in two ways: by enabling the **FullyConnected** mode and by preserving the later interaction with the query embedding.

Table 15 lists the ablation studies we conducted, along with the Recall@2 and Recall@5 scores averaged across the three training datasets: HotpotQA, MuSiQue, and 2Wiki.

Embedding Aggregation: We evaluate three methods for obtaining node embeddings from the GLM encoder's output: **(i)** taking the mean of the node's token embeddings, **(ii)** using the embedding of the first token, and **(iii)** using the embedding of the last token. Our experiments reveal that choosing one method over another does not lead to substantial performance differences. Therefore, we use **mean** embedding aggregation as the default setting in order to represent all tokens equally.

Query Interaction: The node representations interact with the query in two places: **(i)** the query tokens are appended to the retrieved triplets and passed together to the T5 encoder when `use_text` is set to `FullyConnected`, and **(ii)** the node representations are multiplied with the precomputed query embedding from the sentence encoder at a later stage. Disabling this later interaction is partially compensated for when `FullyConnected` remains enabled. Disabling both interactions leads to the worst results, whereas enabling both yields the best Recall@5 and a Recall@2 on par with the best configuration; therefore, this configuration is used as the default setting.

Number of Hops: The graph coverage of the GLM retriever is bounded by the number of entity-level hops it can traverse. Our experiments reveal that increasing the number of hops from 1 to 2 yields strong performance gains, whereas further increases provide only marginal improvements. Therefore, we set the retrieval scope of the model to a 2-hop neighborhood.

Maximum Number of Triplets: The extracted subgraph is also constrained by the maximum number of triplets that can be included within its 2-hop neighborhood. Here, we observe a clear trend in which retrieval performance improves as more triplets are included in the subgraph. To avoid exhausting computational resources, we ultimately set `max_triplets` to 600; however, we hypoth-

esize that the performance could further improve with larger triplet budgets.

T5-Size: Here, we also observe a scaling trend in which retrieval performance improves with larger encoder models. Considering that the parameter sizes explored in this work are still relatively small compared to modern LLMs, which often contain tens to hundreds of billions of parameters, we hypothesize that using larger models could further benefit retrieval performance.

Graph-Aware Attention: We implemented a baseline that linearizes the graph into a sequence of triplets and passes it through T5's encoder to rank entities. This ablates the effect of the **graph-aware attention**. Note that this discards GLM's graph-aware relative-position encoding and sparsity mask, the components that inject graph structure, leaving only T5's standard sequential positional bias. Because of linearization, the token sequence becomes longer than the graph encoding. Thus our default value of 600 triplets did not fit in memory. Consequently we trained the linearized GLM retriever with max_triplets=300. To enable a direct comparison, we also train GLM-RAG restricted to 300 triplets. In line with our expectations, the declining retrieval results clearly show that both the stronger language understanding (GLM-RAG vs. GFM-RAG+) and the graph structure (GLM-RAG vs. linearized T5) contribute to GLM-RAG's success.

D Extensive Retrieval Results

Figures 8 to 10 show the Recall@2 and Recall@5 scores of each model. Additionally, Table 16 shows a breakdown of the retrieval performance under the low-data and full-data settings.

There were several issues with the originally reported results of GFM-RAG. The results in the initial paper (Luo et al., 2025) were affected by a bug in the code. Later, the authors reported updated results of GFM-RAG in their follow-up paper (Luo et al., 2026). However, in this version they evaluated HotpotQA on a different test set. To avoid confusion arising from these inconsistencies, we report the reproduced results of GFM-RAG in our main paper, and Tables 17 and 20 compare the originally reported and reproduced results of GFM-RAG. As shown in Table 17, the only notable difference appears, as expected, on HotpotQA.

Table 17 also shows retrieval comparisons with previous works. The methods are categorized as *"non-structure methods"*, meaning that they do not utilize graph-based approaches, and *"graph-enhanced methods"*. We would like to highlight again that integrating our proposed GLM retriever into the current SOTA method, G-Reasoner, is feasible, since it currently employs a GNN-based retriever, and leave this integration to future work.

D.1 Retrieval Performance by Question Complexity

Furthermore, we analyze the retrieval performance of each model across different levels of question complexity, grouping questions by complexity at both the document and the entity level.

Document Level

To evaluate retrieval success at the document level, we grouped questions by the number of supporting documents they require. Figure 12 shows the recall degradation with increasing numbers of supporting documents for the multi-hop datasets, while Figure 13 shows the same analysis for the single-hop datasets.

Entity Level

Our entity-level measure of question complexity captures how far the hardest-to-reach supporting entity lies from the question entities in the KG.

As shown in Listing 1, there are almost always multiple question (seed) entities as well as multiple gold (supporting) entities for a question. Therefore, for each supporting entity, we calculate the distance to its closest question entity and use the maximum of these distances for each question.

Let $G = (V, E)$ be the undirected KG, $S = \{s_1, \ldots, s_m\} \subseteq V$ the set of question entities, and $T = \{t_1, \ldots, t_n\} \subseteq V \setminus S$ the set of supporting entities. For each supporting entity t_j, we compute its minimum shortest-path distance to any seed entity:

$$d(t_j, S) = \min_{s \in S} d_G(s, t_j)$$

The entity-level distance is then defined as the maximum over all reachable supporting entities:

$$\text{depth}(S, T) = \max_{\substack{j=1,\ldots,n \\ d(t_j,S) \neq \infty}} d(t_j, S)$$

This captures the minimum number of relational hops the model must traverse to reach the most dis-

tant piece of evidence. Questions are grouped according to this depth value to analyze how retrieval performance degrades as reasoning complexity increases.

It is important to note that the entity-level distances in the "single-hop datasets" do not imply that these are "multi-hop datasets", since these hops do not cross document boundaries.

Figures 14 and 15 show the recall degradation with increasing entity-level distance for both single-hop and multi-hop datasets. All models follow a similar trend, with performance decreasing as the entity-level distance increases.


```
{"id": "cc7f68a9086c11ebbd61ac1f6bf848b6",
"question": "Are The M Machine and
Signals Midwest from the same country?",
"answer": "yes",
"answer_aliases": [],
"supporting_facts":
    ["The M Machine",
    "Signals Midwest"],
"question_entities":
    ["the m machine",
    "signals midwest"],
"supporting_entities":
    ["ohio",
    "cleveland",
    "american punk rock band",
    "signals midwest",
    "2011",
    "san francisco",
    "united states",
    "american electronic music duo",
    "four eps",
    "skrillex s label owsla",
    "overall  1 slot on beatport",
    "the m machine",
    "ben swardlick",
    "two remix collections",
    "california",
    "eric luttrell",
    "a single"]
}
```


Listing 1: An example test question from 2Wiki.

D.2 Retrieval Performance by Subgraph Budget

In order to analyze the subgraph budget behavior on different neighborhoods, we separate questions into three categories, namely sparse, medium and dense, using the number of edges around the seed entities. The density distribution of the queries is skewed toward sparse neighborhoods; we account for this by splitting the queries into equal-sized terciles. We then analyze whether the gold documents are reachable with our subgraph selection method under varying hop and triplet counts. This analyzes the **coverage** with increasing subgraph budget.

Figure 16 shows that in dense neighborhoods, 1-hop alone can already exceed the budget, so increasing max_hops without increasing the triplet budget does not help. In sparse neighborhoods, the opposite holds: extra hops help, and the triplet budget is rarely reached. For instance, on HotpotQA, 2 hops and 200 triplets are sufficient to reach >96% of the gold documents in all three density categories.

Following these findings, we have also analyzed how the **retrieval performance** is affected by the density of the neighborhood. Figure 17 shows that performance degrades for questions with denser neighborhoods on 2Wiki and MuSiQue for both models; however, the trend is less clear on HotpotQA. Despite GLM-RAG's restricted visibility and GFM-RAG*'s full visibility, both show similar degradation in dense neighborhoods, indicating that the subgraph restriction does not cause disproportionate degradation in practice.

E Extensive QA Results

Figure 18 shows the Exact Match (EM) and F1 scores for the Wikipedia datasets, and Table 20 provides comparisons with additional baselines.

Figure 19 shows the prompt example provided to the LLM.

Table 7 shows an additional baseline where the model receives a subgraph instead of documents in its context. The subgraph selection uses the same method as GLM-RAG, starting from the question entities and selecting 600 triplets in a 2-hop neighborhood.

In our RAG pipeline, the knowledge graph is constructed by extracting entities from documents, and the retriever acts on a subgraph to find the relevant entities. The entities are then mapped back to the documents so that the "relevant" documents can be used for generation.

Although simply providing the entity-based knowledge subgraph would not be an apples-to-apples comparison and we expect it to underperform, we want to highlight the importance of giving the whole document as context instead of only the graph itself, since documents contain more information.

Table 7 shows how the **subgraph baseline** compares against other methods on MuSiQue, with `gpt-4o-mini` as the LLM. It underperforms all graph-based methods and even the RAG baseline, showing the need for full document context in question answering.

Table 7: QA results with a subgraph baseline. Just giving the subgraph underperforms the document baselines.

MuSiQue	EM	F1
No-Context Baseline	10.8	20.6
Subgraph Baseline	*22.9*	*30.7*
RAG Baseline	23.8	33.3
GFM-RAG	28.7	40.8
GFM-RAG+	31.6	43.3
GFM-RAG*	29.0	40.6
GLM-RAG	**32.1**	**43.8**

Table 8: Retrieval (Recall@2 and Evidence Recall from G-Bench (Xiang et al., 2026)) and QA performance (EM and ACC from G-Bench (Xiang et al., 2026)) on MultiHop-RAG, answers generated with `gpt-4o-mini`.

	R@2	RE	EM	ACC
No Context	-	-	72.8	75.7
RAG Baseline	32.5	55.8	**79.4**	**78.1**
GFM-RAG	34.1	54.5	71.1	73.5
GFM-RAG+	39.0	55.8	72.7	74.2
GLM-RAG (ours)	**60.0**	**58.1**	75.5	77.1

F Extensive Transferability Results

Table 21 shows the models' F1 scores on the downstream QA task. There are two reasons why the high retrieval performance is not fully reflected in the QA results:

Firstly, in comparison to the Wiki datasets, the gold answers for these transfer datasets are full sentences rather than single words or phrases. This means that QA metrics such as EM and F1 do not fully capture the quality of the generated answers.

Secondly, the strong performance of the no-context baseline indicates that much of this information is already contained within `gpt-4o-mini`'s parametric knowledge, meaning that the model does not necessarily require the retrieved context to answer the questions correctly. This potentially dilutes the impact a strong retrieval can have.

Nonetheless, GLM-RAG achieves the best or second-best F1 score on six of the seven single-hop datasets and the second-best EM and accuracy on MultiHop-RAG; however, the performance gaps remain relatively small due to the issues described above.

The same problem arises in the QA evaluation of MultiHop-RAG. Given that `gpt-4o-mini` already achieves high scores without any additional context (see the no-context baseline in Table 8), we conclude that neither EM nor accuracy reflects retrieval quality.

G Extensive Analysis

In the following, we explain how the similarity and distance measures (see Section 6) are computed. Tables 4 and 9 give an overview of the models' similarity and distance measures.

G.1 Similarity Measures

Semantic similarity between the question and retrieved entities is computed as the cosine similarity of their sentence-level embeddings. Specifically, we encode the question text q and each retrieved entity name e_i using a pretrained sentence embedding model (`all-mpnet-base-v2`), yielding embedding vectors $\mathbf{q} = \mathrm{Enc}(q)$ and $\mathbf{e}_i = \mathrm{Enc}(e_i)$. Each embedding is ℓ_2-normalized:

$$\hat{\mathbf{q}} = \frac{\mathbf{q}}{\|\mathbf{q}\|_2}, \quad \hat{\mathbf{e}}_i = \frac{\mathbf{e}_i}{\|\mathbf{e}_i\|_2}$$

The semantic similarity score for a retrieved entity e_i with respect to the question q is then defined as:

$$\mathrm{sim}(q, e_i) = \hat{\mathbf{q}}^\top \hat{\mathbf{e}}_i$$

which is equivalent to the cosine similarity $\cos(\mathbf{q}, \mathbf{e}_i) \in [-1, 1]$. For each sample, the scores are computed over the top-k retrieved entities and averaged to produce a per-sample semantic relevance score.

G.2 Distance Measures

Table 9: Comparison of mean hop distance between the seed entities and the retrieved entities.

Models	GLM-RAG	GFM-RAG*	GFM-RAG	GFM-RAG+
Multi-Hop Datasets				
MuSiQue	1.09	1.22	1.29	1.34
HotpotQA	1.05	1.19	1.23	1.24
2Wiki	1.10	1.19	1.22	1.24
MultiHop-RAG	0.88	1.02	0.99	1.24
Single-Hop Datasets				
TechQA	1.06	1.05	1.17	1.10
PubMedQA	1.04	1.19	1.31	1.44
MS MARCO	1.25	1.26	1.39	1.49
HAGRID	1.32	1.32	1.42	1.56
ExpertQA	1.53	1.59	1.89	2.06
eManual	1.29	1.38	1.50	1.55
DelucionQA	1.31	1.35	1.56	1.49

Structural proximity between retrieved entities and the question (seed) entities is measured as the shortest-path distance in the knowledge graph. Let $G = (V, E)$ be the undirected knowledge graph

and $S = \{s_1, \ldots, s_m\} \subseteq V$ the set of seed entities extracted from the question. For each retrieved entity e_i, we compute its minimum shortest-path distance to any seed entity:

$$d(e_i, S) = \min_{s \in S} d_G(s, e_i)$$

where $d_G(s, e_i)$ is the length of the shortest path between s and e_i in G, obtained via breadth-first search with a cutoff of 10 hops. If no path exists within the cutoff, the entity is considered unreachable. The per-sample structural proximity score is defined as the average distance over all reachable top-k retrieved entities:

$$\bar{d} = \frac{1}{|\{i : d(e_i, S) \neq \infty\}|} \sum_{\substack{i=1 \\ d(e_i,S) \neq \infty}}^{k} d(e_i, S)$$

Lower values indicate that the retrieval model favors entities that are structurally close to the question entities in the KG.

H Error Bars and Significancy Tests

Table 22 shows significance tests with paired bootstrap tests and 95% confidence intervals (CI) that compare GLM-RAG against all other baselines. Holm-Bonferroni correction is applied to control the family-wise error rate. Stars mark results where GLM-RAG is significantly better ($^{*}\, p < .05$, $^{**}\, p < .01$, $^{***}\, p < .001$), and daggers (†) mark results where it is significantly worse.

The results show that the differences between GLM-RAG and GFM-RAG+ are not significant for in-domain retrieval. Out of domain, GLM-RAG is significantly better than GFM-RAG+ on PubMedQA and MultiHop-RAG.

I Declaration of AI Usage

We used AI assistants to speed up programming and to aid with reformulations and proofreading. The content of this work is our own and not inspired by AI assistants.

Table 10: Statistics of the datasets and constructed KG-indexes used for **training**.

Dataset	# Queries	# Documents	# Entities	# Relations	# Triplets
HotpotQA (train)	89,447	874,784	8,259,397	4,134,914	27,362,937
HotpotQA (valid)	1,000	9,742	92,651	46,290	314,944
MuSiQue (train)	18,938	378,724	1,374,033	801,704	4,315,300
MuSiQue (valid)	1,000	20,000	70,080	42,690	213,797
2WikiMultihopQA (train)	165,454	955,412	7,120,577	2,876,828	22,188,618
2WikiMultihopQA (valid)	1,000	5,761	42,489	17,228	130,397
Total Train	273,830	2,208,920	18,579,900	3,814,333	53,866,855

Table 11: Statistics of the datasets and constructed KG-indexes used for **testing**.

Dataset	Domain	#Queries	#Documents	#Entities	#Relations	#Triplets
HotpotQA (test)	Wikipedia	1,000	9,221	87,768	45,112	279,112
MuSiQue (test)	Wikipedia	1,000	11,656	100,853	55,944	319,618
2Wiki (test)	Wikipedia	1,000	6,119	48,779	20,748	160,950
MultiHop-RAG	News	2,255	609	16,147	9,416	24,180
PubMedQA	Biomedical	2,450	5,932	42,389	20,952	149,782
DelucionQA	Customer Support	184	235	2,669	2,298	6,183
TechQA	Customer Support	314	769	10,221	4,606	57,613
eManual	Customer Support	132	102	695	586	1,329
ExpertQA	General Knowledge	203	808	11,079	6,810	16,541
MS MARCO	General Knowledge	423	3,481	24,740	17,042	63,995
HAGRID	General Knowledge	1,318	1,975	23,484	18,653	48,969
G-Bench (CS)	Computer Science	1,018	15,011	187,217	79,578	991,630
G-Bench (Novel)	Novels	2,010	1,881	39,870	37,344	68,483
G-Bench (Medical)	Medical	2,062	390	8,676	6,814	32,386

Table 12: Graph statistics of the datasets and constructed KG-indexes used for **training**.

Dataset	Avg Degree	Density	# Components	Largest CC %
HotpotQA (train)	5.89	0.0001	138	99.66
HotpotQA (valid)	6.02	0.0001	125	99.69
MuSiQue (train)	5.59	0.0001	140	99.53
MuSiQue (valid)	5.48	0.0001	137	99.55
2Wiki (train)	5.47	0.0001	92	99.50
2Wiki (valid)	5.40	0.0001	75	99.59

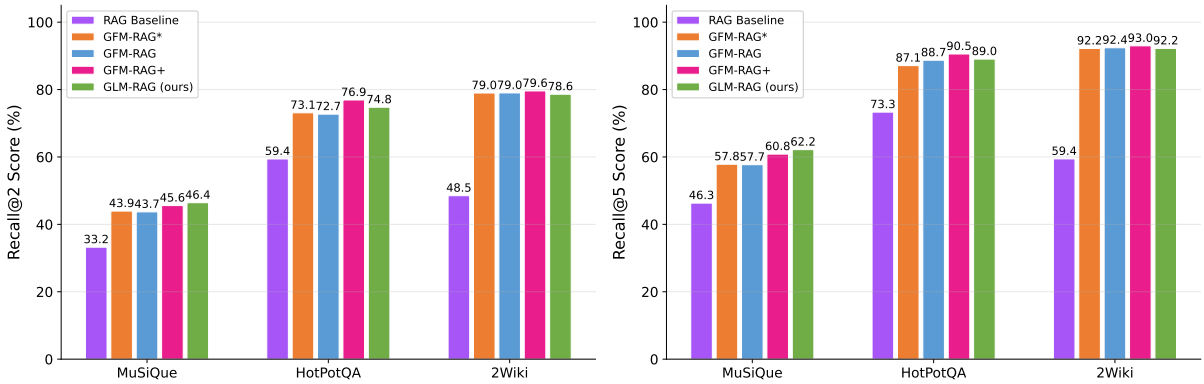


Figure 8: Retrieval performance (Recall@2 and Recall@5) on in-domain Wikipedia datasets.

Table 13: Graph statistics of the test datasets and constructed KG-indexes used for **testing**.

Dataset	Avg Degree	Density	# Components	Largest CC %
HotpotQA (test)	5.68	0.0001	150	99.60
MuSiQue (test)	5.70	0.0001	201	99.56
2Wiki (test)	5.77	0.0001	120	99.41
MultiHop-RAG	2.84	0.0002	378	93.63
PubMedQA	6.28	0.0001	129	99.27
DelucionQA	4.11	0.0015	31	97.45
TechQA	8.98	0.0009	98	97.36
eManual	3.42	0.0049	18	94.39
ExpertQA	2.77	0.0002	449	89.99
MS MARCO	4.62	0.0002	124	98.82
HAGRID	3.81	0.0002	253	97.26
G-Bench (CS)	9.60	0.0001	558	99.29
G-Bench (Novel)	3.17	0.0020	34.25	95.63
G-Bench (Medical)	6.60	0.0008	39	98.80

Table 14: The detailed implementation and training settings of GFM-RAG, GFM-RAG*, GFM-RAG+ and GLM-RAG.

Setting	**GFM-RAG (*, +)**	**GLM-RAG**
KG-index Construction		
OpenIE	GPT-4o-mini	GPT-4o-mini
Entity resolution	ColBERTv2	ColBERTv2
Models		
Backbone	GNN (based on Galkin et al. (2024))	GLM (based on T5-large)
# (Encoder) Layer	6	12
# Parameters	8,144,897	336M
Hidden dim (d_{model})	512	1024
Attention heads	-	16
Graph encoding	-	Levi graph + global RPE
Message	DistMult	-
Aggregation	Sum	-
Entity scorer	2-layer MLP	2-layer MLP
Sentence embedding model	all-mpnet-base-v2	all-mpnet-base-v2
Doc ranker entities	20	20
Finetuning		
α	0.3	0.44
Optimizer	AdamW	AdamW
(Head) Learning rate	5e-4	5e-4
Backbone learning rate	-	1e-4
Batch size	4	2
Training epochs	5	2
Max triplets	-	600
# Hops	-	2

Table 15: Ablation study results averaged over HotpotQA, MuSiQue, and 2Wiki test sets.

Ablation	Method	Recall@2	Recall@5
Embedding Aggregation	Mean (default)	63.3	79.6
	First	63.4	79.6
	Last	63.3	79.2
Query Interaction			
[yes/no and use_text=FullyConnected/no]	Yes and FC (default)	63.3	79.6
	No and FC	63.5	79.4
	Yes and No FC	58.7	74.6
	No and No FC	57.3	73.4
Number of Hops	1 Hop	49.4	66.4
	2 Hops (default)	63.3	79.6
	3 Hops	63.0	79.7
	4 Hops	63.6	79.6
Max Triplets	300	60.8	76.9
	400	61.8	78.4
	500	62.8	79.1
	600 (default)	63.3	79.6
T5 Encoder Size	T5-Small (36M)	60.4	76.7
	T5-Base (110M) (default)	63.3	79.6
	T5-Large (336M)	64.7	80.6
Graph Aware Attention max_triplets = 300	With	60.8	76.9
	Without (linearized)	38.7	51.3

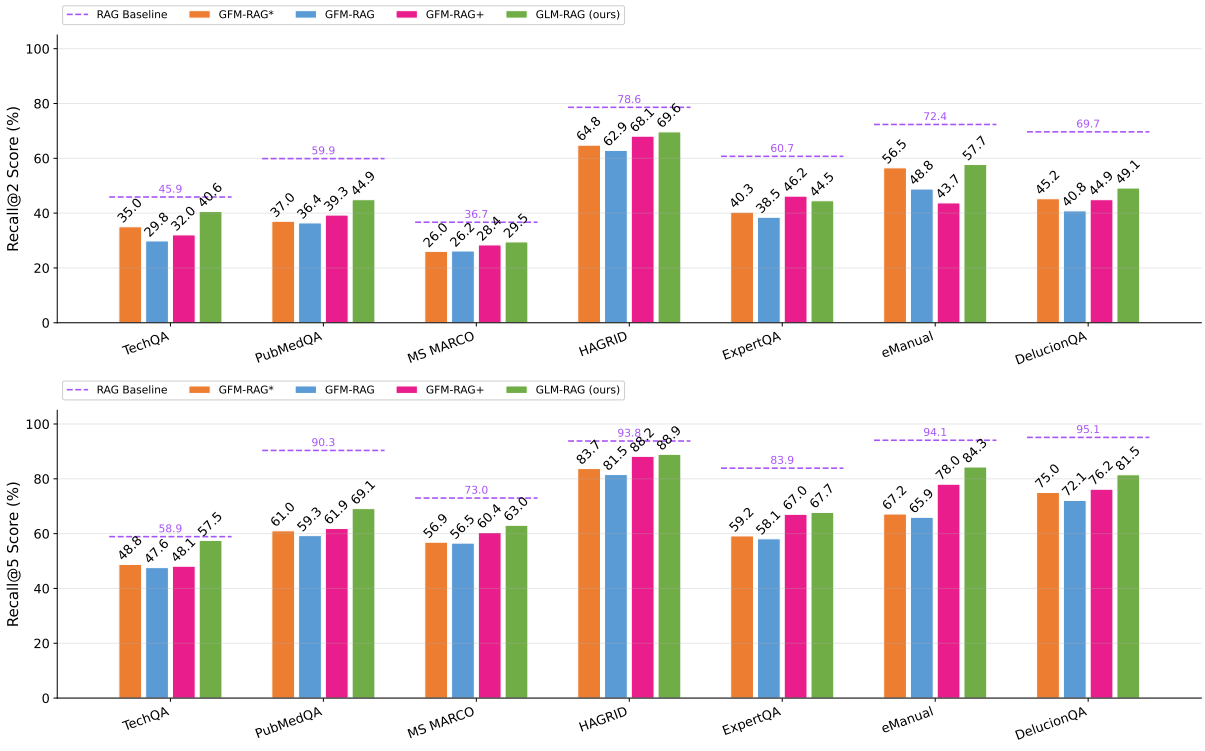


Figure 9: Retrieval performance (Recall@2 and Recall@5) on 7 out-of-domain single-hop datasets.

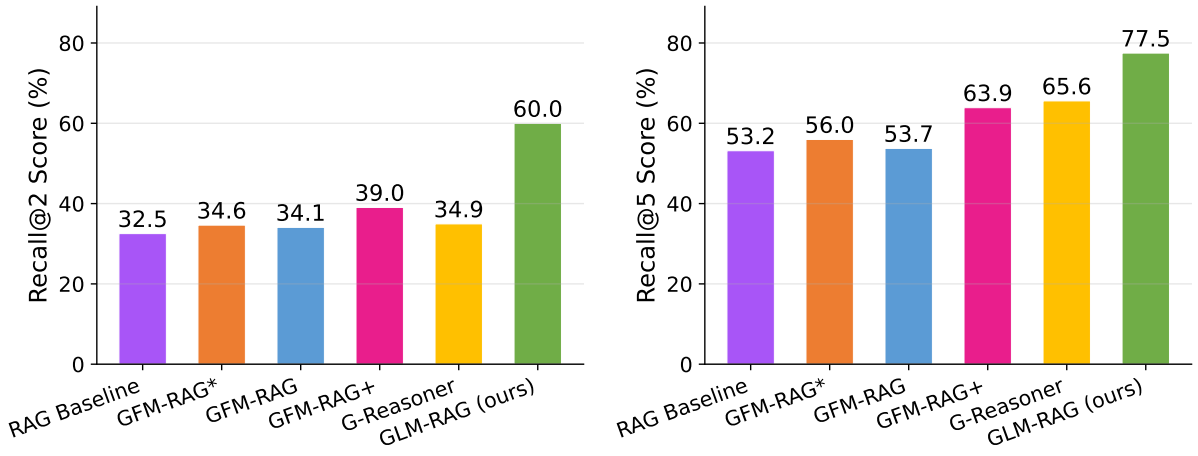


Figure 10: Retrieval performance (Recall@2 and Recall@5) on MultiHop-RAG, an out-of-domain dataset with multi-hop questions.

Table 16: Retrieval performance (Recall@2 and Recall@5) on in-domain Wikipedia datasets in low data and full data settings of comparable graph methods. Best results are **bolded** and second best are underlined.

		HotpotQA		MuSiQue		2Wiki	
Data Amount	Method	R@2	R@5	R@2	R@5	R@2	R@5
	RAG Baseline	59.2	73.0	33.2	46.3	48.5	59.4
60k Q-Doc Pairs	GFM-RAG (reproduced)	69.5	85.9	42.8	57.1	77.4	91.5
	GFM-RAG*	70.4	85.0	43.2	58.2	76.9	91.4
	GFM-RAG+	**72.7**	**88.8**	44.1	59.1	**77.4**	**92.2**
	GLM-RAG (ours)	71.2	88.1	**46.0**	**61.5**	76.8	92.1
277k Q-Doc Pairs	GFM-RAG (reproduced)	72.7	88.7	43.7	57.7	79.0	92.4
	GFM-RAG*	73.1	87.1	43.9	57.8	79.0	92.2
	GFM-RAG+	**76.9**	**90.6**	45.6	60.8	**79.6**	**93.0**
	GLM-RAG (ours)	74.8	89.0	**46.4**	**62.2**	78.6	92.2

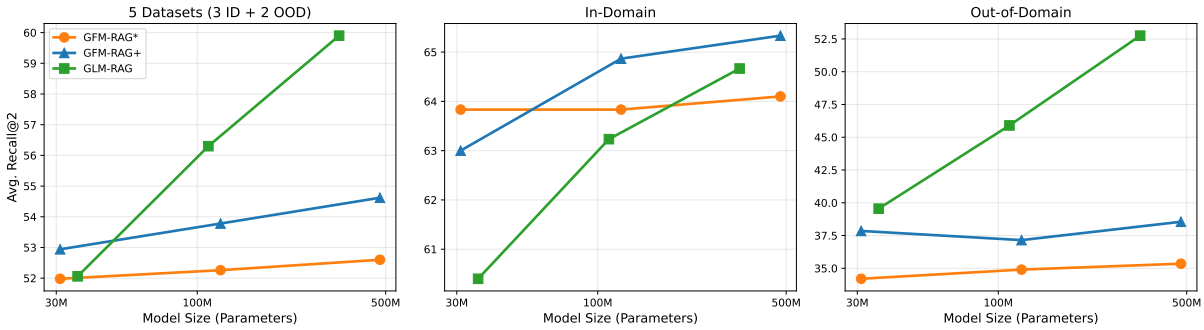


Figure 11: Comparison of the retrieval quality of GFM-RAG*, GFM-RAG+, and GLM-RAG on the three in-domain datasets (HotpotQA, MuSiQue, 2Wiki), as well as on one out-of-domain single-hop dataset (PubMedQA) and one out-of-domain multi-hop dataset (MultiHop-RAG). Scaling up the GFM-RAG variants does not yield the same gains as scaling up GLM-RAG in out-of-domain settings.

Table 17: Retrieval performance comparison (Recall@2 and Recall@5). Best results are **bolded**, second best are underlined, and third best are *italicized*. Baseline scores are adapted from Luo et al. (2026).

Method	HotpotQA		MuSiQue		2Wiki	
	R@2	R@5	R@2	R@5	R@2	R@5
Non-structure Methods						
BM25 (Robertson and Walker, 1994)	55.4	72.2	32.3	41.2	51.8	61.9
ColBERTv2 (Santhanam et al., 2022)	64.7	79.3	37.9	49.2	59.2	68.2
Qwen3-Emb (8B) (Zhang et al., 2025a)	74.1	88.8	*46.8*	62.1	66.2	74.1
Graph-enhanced Methods						
RAPTOR (Sarthi et al., 2024)	58.1	71.2	35.7	45.3	46.3	53.8
GraphRAG (MS) (Edge et al., 2025)	58.3	76.6	35.4	49.3	61.6	77.3
LightRAG (Guo et al., 2024)	38.8	54.7	24.8	34.7	45.1	59.1
KAG (Liang et al., 2025)	59.4	86.1	42.2	*62.4*	61.4	88.3
HippoRAG (Gutiérrez et al., 2024)	60.1	78.5	41.2	53.2	68.4	87.0
HippoRAG 2 (Gutiérrez et al., 2025)	80.5	95.7	53.5	74.2	80.5	95.7
SubgraphRAG (Li et al., 2025)	58.1	71.7	40.6	48.1	70.2	85.3
GFM-RAG (Luo et al., 2025)	75.6	89.6	43.5	57.6	79.1	92.4
GFM-RAG (reproduced)	72.7	88.7	43.7	57.7	79.0	92.4
GFM-RAG*	73.1	87.1	43.9	57.8	79.0	92.2
GFM-RAG+	*76.9*	*90.6*	45.6	60.8	*79.6*	*93.0*
GLM-RAG (ours)	74.8	89.0	46.4	62.2	78.6	92.2
G-Reasoner (Luo et al., 2026)	**85.9**	**97.7**	**54.8**	**74.9**	**81.2**	**98.2**

Table 18: GNN-based and GLM-based retriever models parameter comparison when scaling up the GFM-RAG* by increasing its hidden dimensions from 512 to **1024**, **2048** and **4096**, following Luo et al. (2026). Since we only use the encoder, the parameter count of GLMs is half the original parameter count of T5-models.

Models	Parameter Count
GLM-RAG (t5-small)	36M
GFM-RAG(*, +) (hidden size 1024)	31M
GLM-RAG (t5-base)	110M
GFM-RAG(*, +) (hidden size 2048)	122M
GLM-RAG (t5-large)	336M
GFM-RAG(*, +) (hidden size 4096)	476M

Table 19: Retrieval performance (Recall@2 and Recall@5) of the scaling experiments on the 60k dataset. Best results are **bolded**.

	HotpotQA		MuSiQue		2Wiki		PubMedQA		MultiHop-RAG	
Method	R@2	R@5	R@2	R@5	R@2	R@5	R@2	R@5	R@2	R@5
GFM-RAG* (hidden_dim=512, 8M)	70.4	85.0	43.2	58.2	76.9	91.4	35.8	58.8	29.7	50.1
GFM-RAG* (hidden_dim=1024, 31M)	70.3	86.6	43.5	57.4	77.7	92.0	35.1	59.8	33.3	53.4
GFM-RAG* (hidden_dim=2048, 122M)	71.3	87.0	42.3	57.5	77.9	91.3	35.0	57.4	34.8	57.6
GFM-RAG* (hidden_dim=4096, 476M)	71.8	86.7	42.9	57.4	77.6	91.7	36.1	59.8	34.6	56.1
GFM-RAG+ (hidden_dim=512, 8M)	72.7	**88.8**	44.1	59.1	77.4	**92.2**	38.3	60.3	41.4	62.4
GFM-RAG+ (hidden_dim=1024, 31M)	70.4	87.4	42.2	56.9	76.4	91.0	37.7	60.3	38.0	59.6
GFM-RAG+ (hidden_dim=2048, 122M)	**73.1**	88.1	44.0	57.9	77.5	91.8	39.0	60.5	35.3	57.0
GFM-RAG+ (hidden_dim=4096, 476M)	**73.1**	87.2	44.8	58.2	**78.1**	91.4	37.0	59.8	40.1	62.8
GLM-RAG (t5-small, 36M)	64.4	82.7	43.1	57.1	73.7	90.4	40.5	62.9	38.6	57.7
GLM-RAG (t5-base, 110M)	68.8	85.9	44.8	61.1	76.1	91.9	43.6	66.6	48.2	66.6
GLM-RAG (t5-large, 336M)	71.2	88.1	**46.0**	**61.5**	76.8	92.1	**46.0**	**68.8**	**59.5**	**78.1**

Table 20: QA reasoning performance comparison (Exact Match and F1). `gpt-4o-mini` is used as the LLM. Best results are **bolded**, second best are underlined, and third best are *italicized*. Baseline scores are adapted from Luo et al. (2026).

	HotpotQA		MuSiQue		2Wiki	
Method	EM	F1	EM	F1	EM	F1
Non-structure Methods						
BM25 (Robertson and Walker, 1994)	52.0	63.4	20.3	28.8	47.9	51.2
ColBERTv2 (Santhanam et al., 2022)	43.4	57.7	15.5	26.4	33.4	43.3
Qwen3-Emb (8B) (Zhang et al., 2025a)	53.4	67.6	31.9	44.1	57.2	63.2
Graph-enhanced Methods						
RAPTOR (Sarthi et al., 2024)	50.6	64.7	27.7	39.2	39.7	48.4
GraphRAG (MS) (Edge et al., 2025)	51.4	67.6	27.0	42.0	34.7	61.0
LightRAG (Guo et al., 2024)	9.9	20.2	2.0	9.3	2.5	12.1
KAG (Liang et al., 2025)	59.5	72.2	*33.8*	46.0	67.3	75.1
HippoRAG (Gutiérrez et al., 2024)	46.3	60.0	24.0	35.9	59.4	67.3
HippoRAG 2 (Gutiérrez et al., 2025)	*56.3*	*71.1*	35.0	49.3	60.5	69.7
SubgraphRAG (Li et al., 2025)	44.5	57.0	25.1	35.7	62.7	69.0
G-retriever (He et al., 2024)	41.4	53.4	23.6	34.3	33.5	39.6
GFM-RAG (Luo et al., 2025)	56.2	69.5	30.2	*49.2*	69.8	77.7
GFM-RAG (reproduced)	55.1	70.7	28.7	40.8	67.8	75.7
GFM-RAG+	55.3	70.6	31.6	43.3	*69.0*	*77.0*
GFM-RAG*	53.4	68.3	29.0	40.6	67.9	75.9
GLM-RAG (ours)	55.7	70.3	32.1	43.8	67.5	75.5
G-Reasoner (Luo et al., 2026)	**61.4**	**76.0**	**38.5**	**52.5**	**74.9**	**82.1**

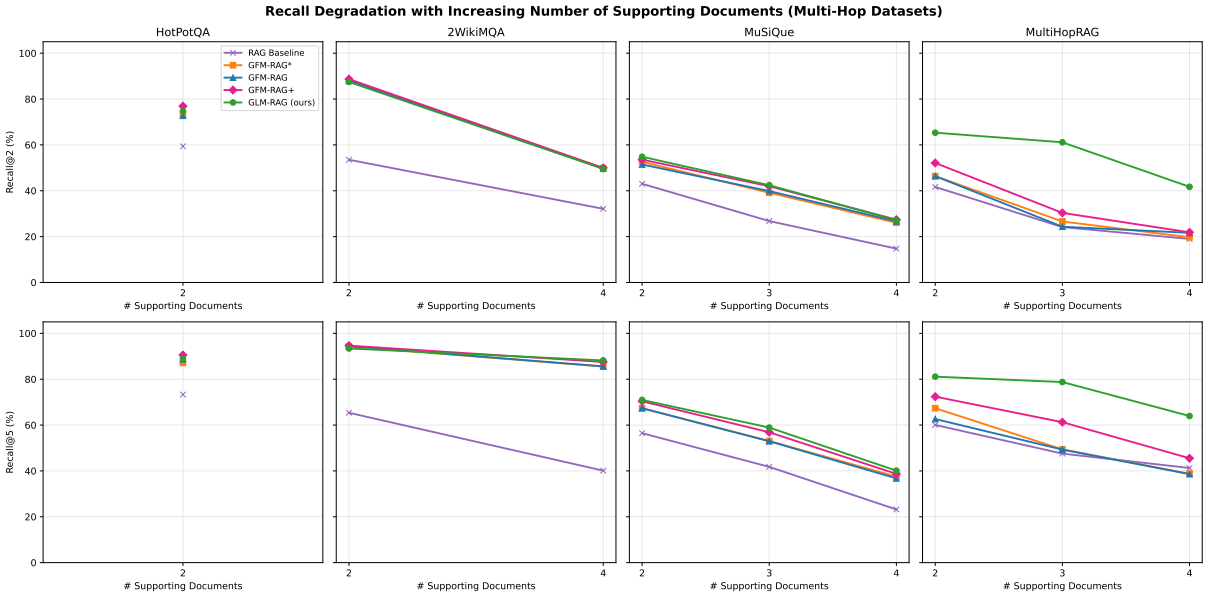


Figure 12: Recall degradation with increasing number of supporting documents for multi-hop datasets.

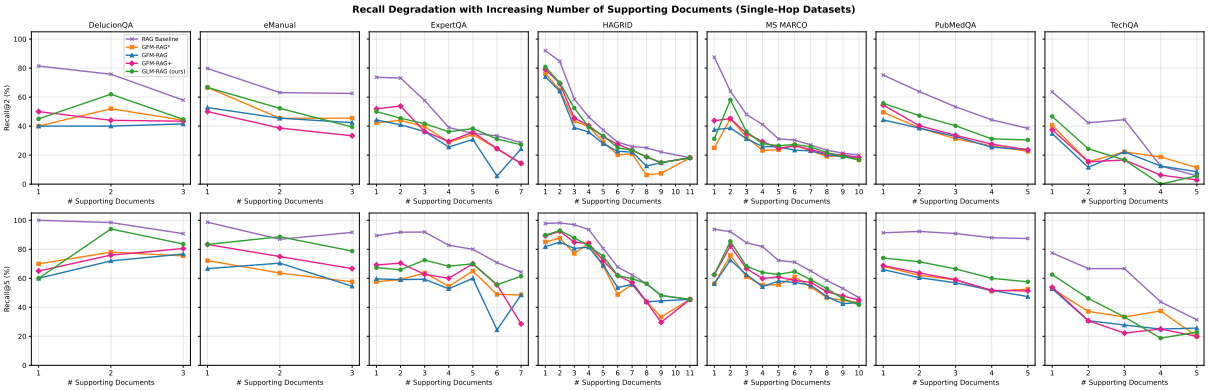


Figure 13: Recall degradation with increasing number of supporting documents for single-hop datasets.

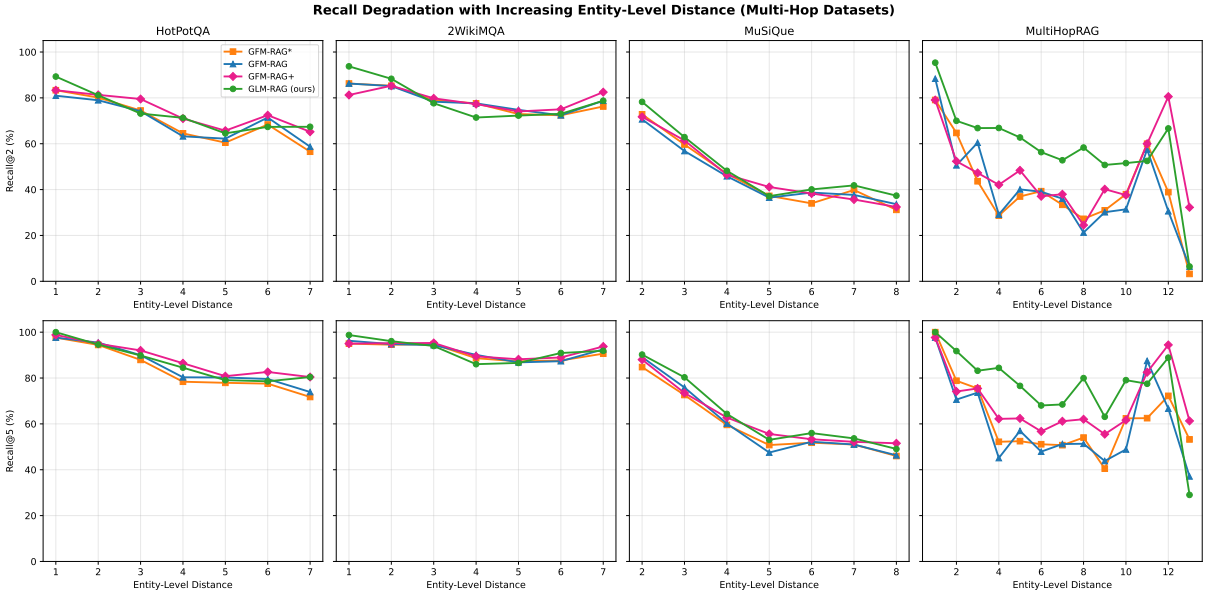


Figure 14: Recall degradation with increasing entity-level distance between question and supporting entities for multi-hop datasets.

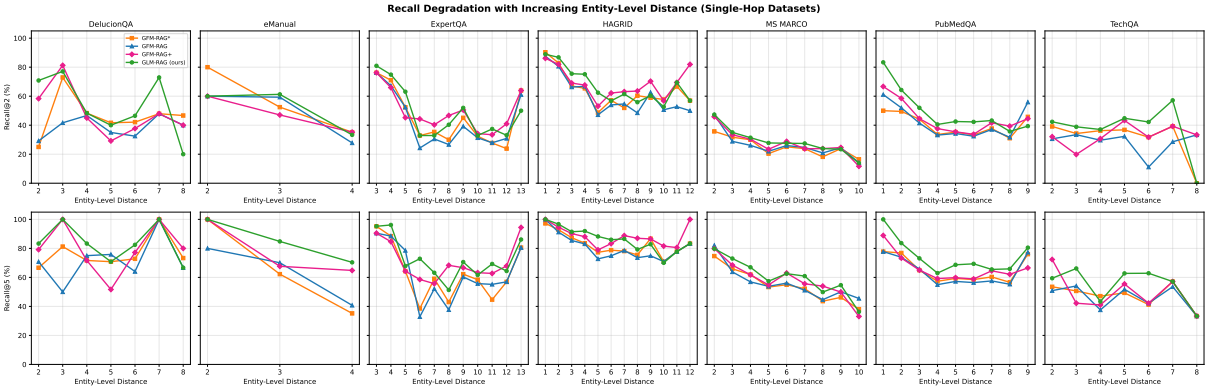


Figure 15: Recall degradation with increasing entity-level distance between question and supporting entities for single-hop datasets.

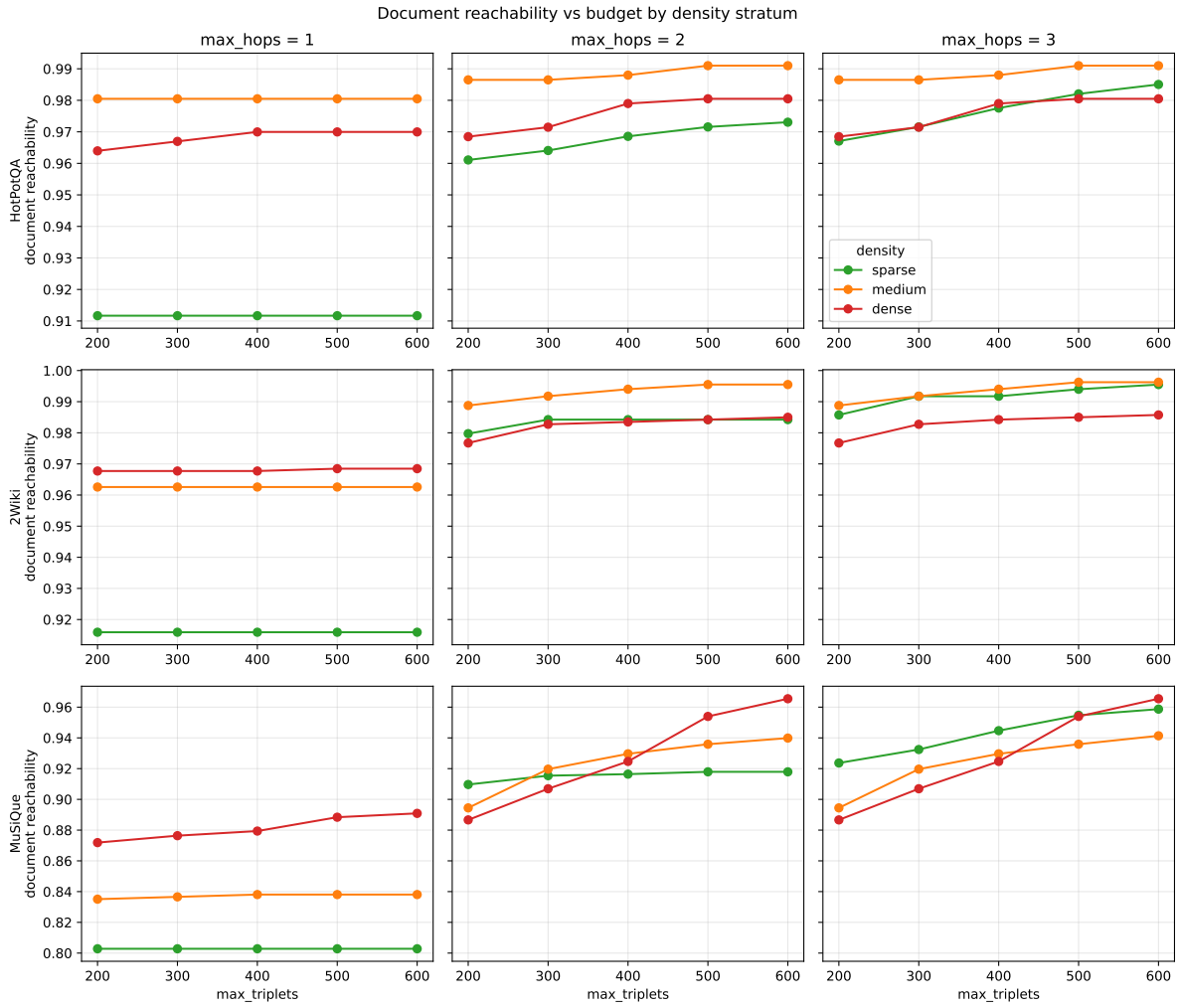


Figure 16: Golden document reachability of GLM-RAG split into 3 question categories regarding their neighborhood density with increasing number of hops and triplets.

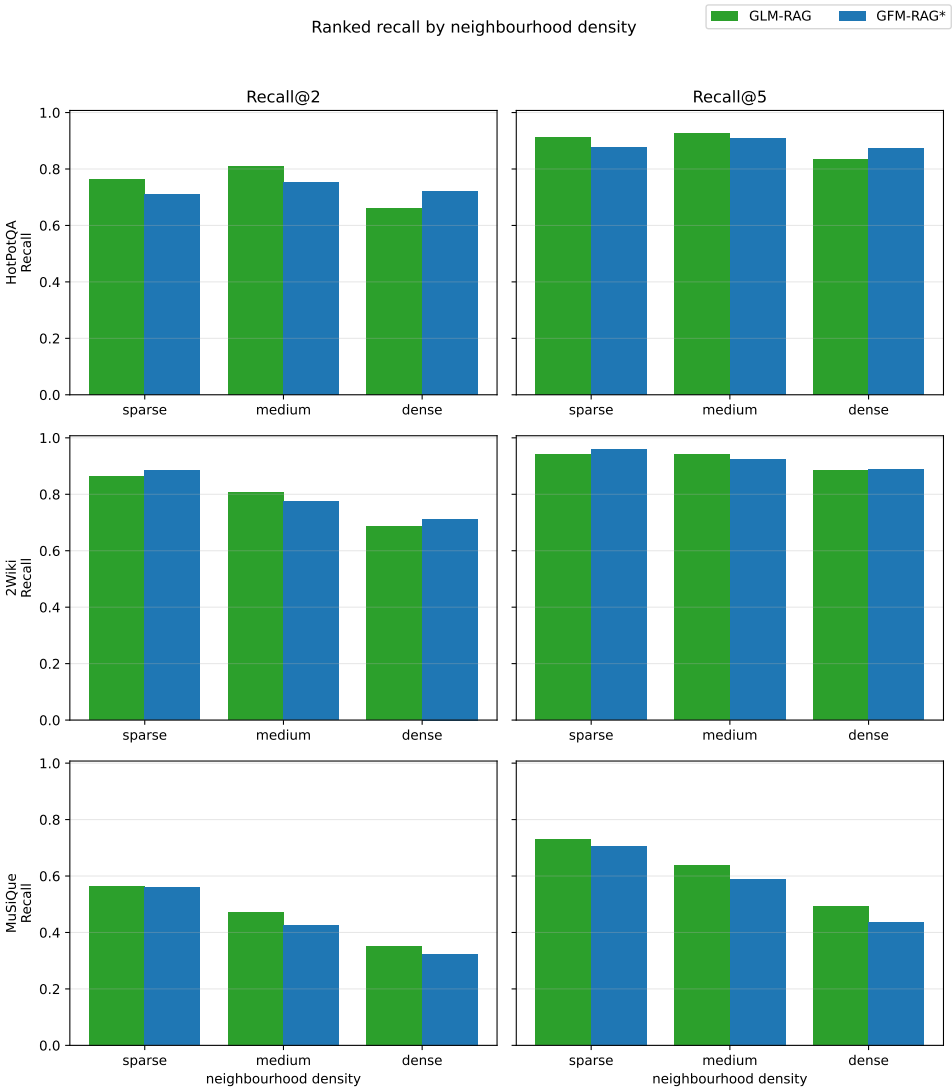


Figure 17: Retrieval recall of GLM-RAG and GFM-RAG* split into 3 question categories regarding their neighborhood density.

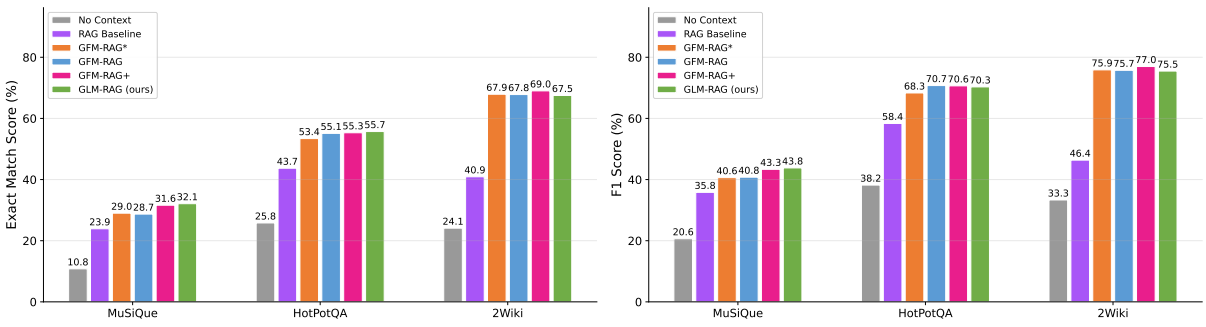


Figure 18: QA performance (Exact Match and F1) on in-domain Wikipedia datasets with `gpt-4o-mini`.

Table 21: QA performance (F1 score) on out-of-domain datasets, answers generated with `gpt-4o-mini`.

Method	TechQA	PubMedQA	MS MARCO	HAGRID	ExpertQA	eManual	DelucionQA
No Context	19.1	20.8	29.9	21.5	16.7	19.6	19.1
RAG Baseline	**25.4**	**21.7**	**36.7**	**27.1**	19.4	<u>31.7</u>	32.1
GFM-RAG*	23.6	21.0	35.7	27.1	19.6	31.2	<u>33.6</u>
GFM-RAG	23.5	21.1	35.3	26.9	<u>19.7</u>	31.5	**34.3**
GFM-RAG+	21.3	20.3	33.0	26.8	17.7	27.7	30.8
GLM-RAG (ours)	<u>25.2</u>	<u>21.3</u>	<u>35.9</u>	<u>27.1</u>	**19.9**	**32.5**	33.2

Table 22: Retrieval recall (mean $\pm$ half the 95% bootstrap CI width) on the question set shared by all systems within each dataset. Markers give a paired bootstrap test of GLM vs each system, corrected across all 88 comparisons with Holm-Bonferroni (FWER): $^{*}/^{**}/^{***}$ = GLM significantly better ($p < .05/.01/.001$), † = significantly worse.

	Recall@2					**Recall@5**				
Dataset	RAG	GFM-RAG	GFM-RAG*	GFM-RAG+	GLM-RAG	RAG	GFM-RAG	GFM-RAG*	GFM-RAG+	GLM-RAG
2WikiMultihopQA	$.485_{\pm.017}^{***}$	$.790_{\pm.016}$	$.790_{\pm.017}$	$.796_{\pm.016}$	$.785_{\pm.016}$	$.594_{\pm.017}^{***}$	$.924_{\pm.011}$	$.922_{\pm.011}$	$.930_{\pm.011}$	$.922_{\pm.011}$
DelucionQA	$.687_{\pm.077}^{\dagger\dagger\dagger}$	$.423_{\pm.107}$	$.473_{\pm.098}$	$.460_{\pm.093}$	$.500_{\pm.097}$	$.946_{\pm.044}$	$.717_{\pm.100}$	$.747_{\pm.087}$	$.770_{\pm.088}$	$.813_{\pm.083}$
eManual	$.700_{\pm.121}$	$.514_{\pm.148}$	$.581_{\pm.145}$	$.457_{\pm.143}$	$.600_{\pm.140}$	$.967_{\pm.038}$	$.676_{\pm.140}$	$.686_{\pm.133}$	$.776_{\pm.119}$	$.848_{\pm.098}$
ExpertQA	$.638_{\pm.052}^{\dagger\dagger\dagger}$	$.385_{\pm.058}^{*}$	$.403_{\pm.056}$	$.462_{\pm.055}$	$.445_{\pm.056}$	$.892_{\pm.037}^{\dagger\dagger\dagger}$	$.581_{\pm.064}^{***}$	$.592_{\pm.060}^{***}$	$.670_{\pm.057}$	$.677_{\pm.056}$
HAGRID	$.807_{\pm.017}^{\dagger\dagger\dagger}$	$.629_{\pm.023}^{***}$	$.648_{\pm.022}^{***}$	$.681_{\pm.020}$	$.696_{\pm.021}$	$.961_{\pm.008}^{\dagger\dagger\dagger}$	$.815_{\pm.019}^{***}$	$.837_{\pm.018}^{***}$	$.882_{\pm.015}$	$.889_{\pm.015}$
MS MARCO	$.374_{\pm.022}^{\dagger\dagger\dagger}$	$.262_{\pm.022}^{*}$	$.260_{\pm.022}$	$.284_{\pm.021}$	$.295_{\pm.021}$	$.736_{\pm.021}^{\dagger\dagger\dagger}$	$.565_{\pm.028}^{***}$	$.569_{\pm.029}^{***}$	$.604_{\pm.026}$	$.630_{\pm.027}$
PubMedQA	$.603_{\pm.016}^{\dagger\dagger\dagger}$	$.354_{\pm.020}^{***}$	$.364_{\pm.019}^{***}$	$.382_{\pm.019}^{***}$	$.437_{\pm.019}$	$.914_{\pm.011}^{\dagger\dagger\dagger}$	$.582_{\pm.021}^{***}$	$.601_{\pm.021}^{***}$	$.610_{\pm.021}^{***}$	$.684_{\pm.020}$
TechQA	$.561_{\pm.083}$	$.290_{\pm.073}^{***}$	$.340_{\pm.078}$	$.310_{\pm.081}$	$.404_{\pm.080}$	$.715_{\pm.073}$	$.457_{\pm.079}^{***}$	$.482_{\pm.080}$	$.480_{\pm.080}$	$.571_{\pm.078}$
HotpotQA	$.592_{\pm.019}^{***}$	$.728_{\pm.020}$	$.733_{\pm.021}$	$.770_{\pm.019}$	$.745_{\pm.019}$	$.730_{\pm.019}^{***}$	$.887_{\pm.016}$	$.871_{\pm.017}$	$.906_{\pm.014}$	$.890_{\pm.015}$
MultiHop-RAG	$.312_{\pm.013}^{***}$	$.341_{\pm.019}^{***}$	$.346_{\pm.020}^{***}$	$.390_{\pm.019}^{***}$	$.600_{\pm.019}$	$.525_{\pm.015}^{***}$	$.537_{\pm.021}^{***}$	$.560_{\pm.020}^{***}$	$.639_{\pm.020}^{***}$	$.775_{\pm.017}$
MuSiQue	$.332_{\pm.016}^{***}$	$.437_{\pm.019}^{***}$	$.439_{\pm.019}$	$.456_{\pm.019}$	$.463_{\pm.018}$	$.463_{\pm.017}^{***}$	$.577_{\pm.020}^{***}$	$.578_{\pm.020}^{***}$	$.608_{\pm.020}$	$.620_{\pm.019}$

Question Answering Prompt

<system_prompt>
As an advanced reading comprehension assistant, your task is to analyze text passages and corresponding questions meticulously. Your response start after "Thought: ", where you will methodically break down the reasoning process, illustrating how you arrive at conclusions. Conclude with "Answer: " to present a concise, definitive response, devoid of additional elaborations.'

<examples>
input: |-
Wikipedia Title: Kurram Garhi
Kurram Garhi is a small village located near the city of Bannu, which is the part of Khyber Pakhtunkhwa province of Pakistan. Its population is approximately 35000. Barren hills are near this village. This village is on the border of Kurram Agency. Other nearby villages are Peppal, Surwangi and Amandi Kala.

Wikipedia Title: 2001–02 UEFA Champions League second group stage
Eight winners and eight runners- up from the first group stage were drawn into four groups of four teams, each containing two group winners and two runners- up. Teams from the same country or from the same first round group could not be drawn together. The top two teams in each group advanced to the quarter- finals.

Wikipedia Title: Satellite tournament
A satellite tournament is either a minor tournament or event on a competitive sporting tour or one of a group of such tournaments that form a series played in the same country or region.

Wikipedia Title: Trojkrsti
Trojkrsti is a village in Municipality of Prilep, Republic of Macedonia.

Wikipedia Title: Telephone numbers in Ascension Island
Country Code:+ 247< br> International Call Prefix: 00 Ascension Island does not share the same country code(+290) with the rest of St Helena.

Question: Are both Kurram Garhi and Trojkrsti located in the same country?
Thought:

response: |-
Kurram Garhi is located in the country of Pakistan. Trojkrsti is located in the country of Republic of Macedonia. Thus, they are not in the same country.
Answer: no.

<doc_prompt>
"Wikipedia Title: {title}
{content}"

<question>
"Question: {question}
Thought: "

Figure 19: The one-shot prompt template for 2Wiki. MuSiQue and HotpotQA have similar templates, adjusted to their examples. For the no-context baseline, only the <system_prompt> is given.