

Mix-of-Granularity: Optimize the Chunking Granularity for Retrieval-Augmented Generation

Zijie Zhong

Shanghai AI Laboratory

zhongzijie@pjlab.org.cn

Hanwen Liu

Beihang University

Xiaoya Cui

Beihang University

xiaoya@buaa.edu.cn

Xiaofan Zhang*

Shanghai AI Laboratory

zhangxiaofan@pjlab.org.cn

Zengchang Qin*

Beihang University

zcqin@buaa.edu.cn

Abstract

Integrating information from various reference databases is a major challenge for Retrieval-Augmented Generation (RAG) systems because each knowledge source adopts a unique data structure and follows different conventions. Retrieving from multiple knowledge sources with one fixed strategy usually leads to under-exploitation of information. To mitigate this drawback, inspired by Mix-of-Expert, we introduce Mix-of-Granularity (MoG), a method that dynamically determines the optimal granularity of a knowledge source based on input queries using a router. The router is efficiently trained with a newly proposed loss function employing soft labels. We further extend MoG to MoG-Graph (MoGG), where reference documents are pre-processed as graphs, enabling the retrieval of distantly situated snippets. Experiments demonstrate that MoG and MoGG effectively predict optimal granularity levels, significantly enhancing the performance of the RAG system in downstream tasks. The code of both MoG and MoGG will be made public.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has become a popular method for enhancing Large Language Models (LLMs). The core concept of RAG involves retrieving relevant information from external knowledge bases to provide additional context to the LLM, enabling it to generate more precise and grounded responses. RAG offers a promising and practical solution to mitigate LLMs’ hallucinations because (1) it can be applied to any LLM, even those accessible only via APIs, and (2) the reference information is easy to modify or update. Many LLM-based products are supported by RAG systems, with examples spanning various industries such as customer service, advertising and marketing, education and e-learning,

healthcare, and e-commerce and retailing (Khan, 2023; Xiong et al., 2024; Soman and Roychowdhury, 2024; Ke et al., 2024; Radeva et al., 2024).

The quality of the retrieved snippets is crucial for the final generation, consequently, much research has focused on the retrieval phase. Currently, most RAG systems follow the Dual-Encoder Architecture (Dong et al., 2022) (DEA) paradigm, in which the reference documents are divided into small snippets (chunks), encoded by specific encoders, and then stored in the vector database (e.g. FAISS (Johnson et al., 2017) or Neo4j (Company, 2012)) as embeddings. Thanks to its scalability, the DEA paradigm shows great potential for connecting LLMs with knowledge database of different formats, including knowledge graphs, textbooks, or Wikipedia articles.

Optimizing the chunk size of the reference knowledge database is essential for enhancing the precision and recall during the retrieval phase. However, this optimization presents significant challenges in the implementation of RAG systems for two primary reasons: (1) An optimal chunking size needs to be determined for each knowledge database due to their dissimilar data structures and information densities. For example, the optimal chunking size for medical textbooks is longer than the one for Medical Knowledge Graph Database (MKGD like Hetionet (Himmelstein et al., 2017)), as Medical textbooks typically contain lengthy passages, while an MKGD consists of entities represented by shorter terms. (2) Even within a single knowledge database, using a uniform chunking size for all input queries can yield suboptimal retrieval results, as the queries themselves exhibit varying levels of granularity. As shown in Figure 1), when the user asks about one disease (fine-grained question), chunking the reference document in finer granularity is better; whereas, when the user asks for broader information (coarse-grained question), a more coarse granularity is preferred. In practice,

*Corresponding authors

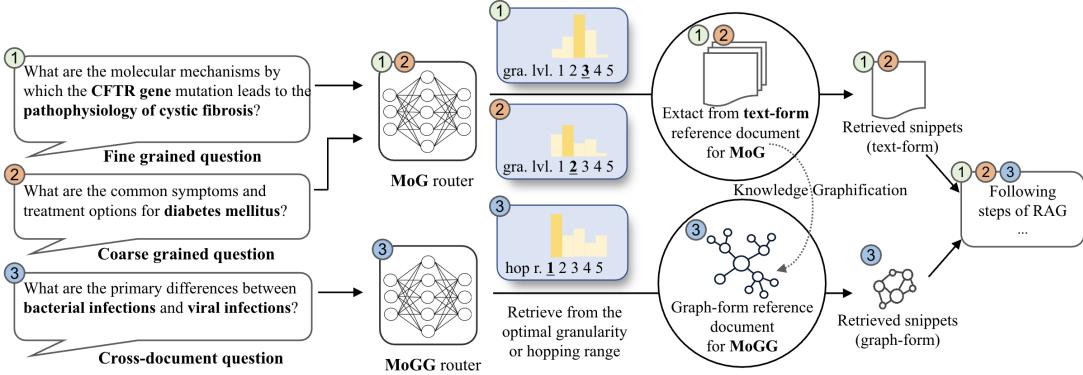


Figure 1: MoG automatically selects the optimal granularity when extracting information from the reference database (scenarios 1 and 2), achieving both high pertinence and coverage. When relevant information is dispersed across distant sections (scenario 3), the reference documents are pre-processed as graph, then MoGG is applied to retrieve these separate snippets from the best hopping range.

the optimal "uniform chunking size" is determined through parameter tuning, which is not only tedious but also does not guarantee high precision and recall.

Therefore, the community calls for a method to dynamically determine the optimal chunking size, for which we propose the Mix-of-Granularity (MoG). We draw inspiration from Mix-of-Experts (Chen et al., 2022), which is a machine learning architecture that dynamically selects the most pertinent “expert sub-network” for each input token using a router module. Similarly, MoG involves leveraging a router to select the best granularity levels from which the reference snippets are extracted.

Even with this flexibility newly introduced, MoG still has difficulty dealing with broader queries that require distantly situated snippets, e.g. snippets stored in different knowledge databases. In such cases (cross-document question in Figure 1), adjusting only the granularity is not helpful because the necessary information is so distantly located that it can never be covered by a reasonably large chunking window. To better answer these broader queries, we extend MoG to MoG-Graph (MoGG). In MoGG, the reference documents are pre-processed as a graph, allowing relevant snippets to be included as neighbors of each other, regardless of their distance in the original databases. This extension further improves the performance of cross-source retrieval.

When training MoG(G) under the supervised learning setting, the backward propagation is blocked by the top- k selection at the end of the retrieval phase, which is a common practice in most RAG systems. To solve it, we introduce a

loss function using soft labels. Soft labels are approximate training signals generated using offline algorithms or models like TF-IDF (Ramos, 2003) or RoBERTa (Liu et al., 2019). With the soft labels, the top- k selection is excluded from the training process, thus the issue of backward propagation is circumvented and the training is accelerated.

In conclusion, the main contributions of this work are:

(1) We propose MoG, which dynamically determines the optimal granularity level for retrieval with the help of a router, achieving a balanced trade-off between precision and recall.

(2) We extend MoG to MoGG by reorganizing the reference document in the form of a graph, thereby further improving the quality of cross-source retrieval involving multiple knowledge databases.

(3) We introduce a loss function utilizing soft labels to overcome the challenges associated with training with top- k selection.

2 Related Work

2.1 Retrieval-Augmented Generation

RAG (Lewis et al., 2020) has emerged as a standard practice to enhance the LLMs by mitigating their problems of “hallucinations” and knowledge cut-off. A RAG system typically includes a Retriever that extracts relevant information from an external knowledge database, and a backbone LLM to generate grounded responses by in-context learning (Dong et al., 2023). Previous retrieval-focused methods have evolved from retrieving tokens (Khandelwal et al., 2020) or entities (Nishikawa et al., 2022) to more complex structures like chunks

(Ram et al., 2023) or graphs (Kang et al., 2023). Granularity matters a lot in retrieval, as coarse-granularity-retrieval yields more information with lower precision, while fine-granularity-retrieval offers comprehensive information at the cost of efficiency. More strategies like single (Wang et al., 2023b; Shi et al., 2023), adaptive (Jiang et al., 2023; Huang et al., 2024), or multiple retrieval (Izacard et al., 2022b) are introduced to improve the retrieval phase’s performance. Regarding the generation phase, various information fusion techniques are developed to integrate retrieval results to LLM in its input (Khattab et al., 2023), intermediate (Borgeaud et al., 2022), or output layers (Liang et al., 2024).

2.2 Chunking Optimization

Optimal chunk size is crucial for RAG system as breaking down documents into small chunks is the first step to encode them. Naive chunking strategies, such as “fixed-size” chunking and “recursive chunking,” attempt to create snippets of identical size. Later works explored more chunking optimization techniques. One line of work focuses on increasing the recall of retrieval. For example, the “Sliding Window Chunking” (Safjan, 2023) allows layered retrieval by merging globally related information across multiple processes. The “Parent Document Retrieval” (team, 2023) retrieves using small chunks and returns larger blocks of context for later generation. Another line of work seeks to include more semantic information of the context to improve retrieval accuracy. “Metadata Filtering” (Siebler, 2024) leverages document metadata to filter snippets; “Context-Enriched Chunking” (team, 2024) breaks down information into segments and adds semantic summaries before retrieval; while “Windowed Summarization Chunking” (team, 2024) enriches each chunk with a windowed summary of the previous chunks. Many of these techniques, including MoG(G), are compatible with each other and can be combined to achieve better performance.

2.3 Graph-Based Text Processing

Graph-based text processing techniques combine research in graphs and text retrieval. Previous works exploit semantic similarities between small snippets (a sentence or several words) and reorganize the text material into a graph using Entity Recognition and Relation Construction algorithms (Melnyk et al., 2022; Guo et al., 2020; You et al.,

2018). Breaking the constraint of the single dimension of text corpus, these methods allow chunks of the same topic to be grouped as neighbors in a graph, thus show great potential in tasks requiring long context reasoning or multi-hop reasoning. For example, “graph indexing” (Gao et al., 2024) transforms entities and relationships into nodes and connections, improving the relevance of retrieved snippets significantly. RAPTOR (Sarthi et al., 2024) organizes snippets as a tree (a special form of a graph) by recursively clustering them, where all non-leaf nodes correspond to summaries of their child nodes. This processing allows access to information at different granularity levels, which is beneficial to summarization tasks. In GMoE (Wang et al., 2023a), authors use different expert networks to handle hop-1, hop-2, and mixed hop-1 & hop-2 neighbors of a node in a graph, which inspired our design of MoGG.

3 Methodology

3.1 Preliminaries

MoG(G) is designed to enhance the Retriever of a RAG system. A typical RAG system comprises a Retriever \mathcal{R} , a Generator \mathcal{G} , and a series of external reference corpus $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, \dots, \mathcal{K}_k\}$. A RAG system takes a user query q as input, retrieves reference document pieces (snippets or chunks) from \mathcal{K} , and uses these snippets to help the \mathcal{G} produce the final responses. A popular architecture for the Retriever is DEA (Dong et al., 2022), where the query q and all the snippets in \mathcal{K} are encoded into embeddings (e_q and e_s) using the same encoder \mathcal{E} . The extraction of relevant snippets is achieved by calculating the similarity between e_q and e_s . The snippets with highest similarity scores are extracted and injected into the backbone LLM via prompt.

3.2 Naive MoG

3.2.1 Multi-granularity Router

We apply the idea of Mix-of-Expert (Chen et al., 2022) (MoE) to automatically determine the best granularity level in the retrieval phase. In a MoE system, different input tokens are routed to the best expert network based on the weights output by the router. Similarly, in MoG, a router is trained to predict the importance weight of different granularity levels based on the user’s input, so that the snippets from the best granularity level are prioritized. By employing such a routing optimization method, we can effectively adjust the chunk size ac-

cording to different scenarios. Take one corpus \mathcal{K} as example, its documents are chunked in n_{gra} candidate granularities before the retrieval (in Figure 2, $n_{gra} = 5$). Although some methods can dynamically adjust chunks using well-trained models, we prioritize the most commonly used and simplest chunking method from the perspective of practical efficiency. The snippets are chunked without overlap, and each chunk in j ($j \in [2, n_{gra}]$) granularity level is formed with 2 adjacent chunks in $j - 1$ granularity level (granularity level 1 is the finest). Each chunk is assigned a similarity score using BM25 (Robertson and Zaragoza, 2009) with respect to user’s input query q . In each granularity, k_r most relevant snippets are extracted ($k_r = 3$ in Figure 2), forming a pool of candidate snippets of size $n_{gra} \times k_r$. In parallel, q is encoded via RoBERTa before being mapped to a weight w (a vector of the same length as n_{gra}) by the router. The chunks’ similarity scores are then weighted and summed with w .

Empirically, directly retrieving the top snippets across different granularities based on their weighted similarity scores will give biased results, because the scores of more coarsened granularity levels are systematically higher (more discussions in Appendix A). Therefore, we adopt another selection strategy involving two steps: (1) select the top- k ($k = 1$ in Figure 2) finest-grained chunk $chunk_r$; (2) retrieve the chunk containing $chunk_r$ from the optimal granularity level g_r . The selection process can be formalized as follows:

$$chunk_r = \text{top}_k\text{-argmax}_{c \in \mathcal{C}}(t_{rs}(c) \cdot w); \quad (1)$$

$$g_r = \underset{g \in [1, n_{gra}] | w_g \neq 0}{\text{argmax}} w_g \quad (2)$$

where \mathcal{C} represents the set of all chunks of the finest granularity level, $t_{rs}(\cdot)$ represents the relevance score of a chunk, and w_g represents the g -th element in w .

Two snippets are highlighted in Figure 2 as examples to help understanding. For each chunk of the finest granularity level, if it is retrieved among the top- k_r snippets from one granularity level, its relevance score for that level is recorded as the actual relevance score (like the red chunk); otherwise, it is filled with 0 (like some zeroes padded for the blue chunk). In this way, we get the relevance scores t_{rs} of each finest snippet. The blue chunk has highest weighted similarity score, it is

therefore selected as $chunk_r$. In this example, the final snippet selected is the one extracted from 3rd granularity level containing $chunk_r$.

3.2.2 Soft-labels

In MoG, we train a Multi-Layer Perceptron (MLP) as the router in a supervised learning method. The input for the router is the embedding of the input query q generated with RoBERTa (Liu et al., 2019), which is then mapped to w by the router. The intrinsic training signal is the “labels” (l) in the Medical Question-Answering (MQA) datasets. In some MQA datasets, the ground truth snippets to be retrieved are provided, so we use them as “labels” directly; otherwise, the concatenation of the strings of the “Question” and “Answer” is used as the “label”. A natural training objective is to maximize the semantic similarity between $chunk_r$ and l by adjusting w . Such a label helps us to train the router to choose the best granularity based on the input query, which is a *prior*, with solid *posterior* information (the similarity between the ans and the text of different granularities). Unfortunately, this label can not be used to guide the training directly because there is a non-differentiable top- k selection in the way. The soft labels are proposed to bypass the top- k selection during the training: For each query q , the most relevant snippet (S_{best}) is retrieved from the reference documents of each granularity level with BM25 (Robertson and Zaragoza, 2009). The semantic similarity between each snippet in S_{best} and the label l is then calculated (with static models including TF-IDF (Ramos, 2003), RoBERTa (Liu et al., 2019), or hitrate score) and stored in sim_{best} . We create a soft label of 0.8 (resp. 0.2) for the most (resp. the second) similar snippet in S_{best} , and pad 0 for the other snippets.

For example, the soft labels corresponding to $sim_{best,1}$ [0, 0.32, 0.11, 0.88, 0.45] and $sim_{best,2}$ [0.95, 0.07, 0.22, 0.11, 0.19] are sl_1 [0, 0, 0, 0.8, 0.2] and sl_2 [0.8, 0, 0.2, 0, 0], respectively.

The values of the soft labels are designed to guide the router in distinguishing the relative importance among the granularity candidates. Empirically, setting these values to either [0.8, 0.2, 0] or [0.7, 0.3, 0] yields similar results. With the soft labels (sl) built, we can train the router to predict a high value (0.8) for the optimal granularity level, while conserving certain flexibility to choose the second-best granularity level. The router is trained by minimizing a Binary Cross Entropy loss func-

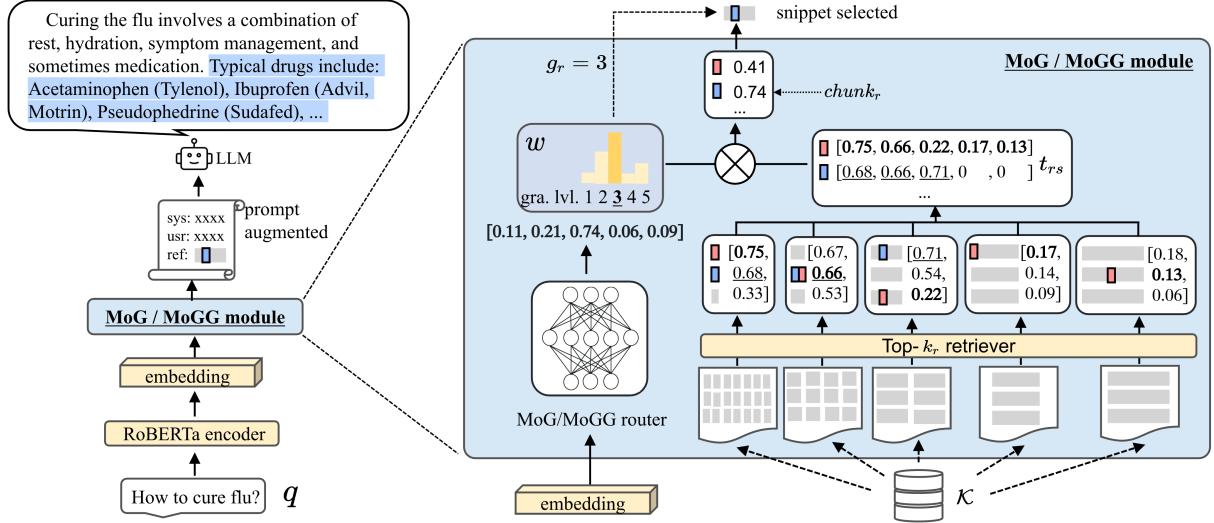


Figure 2: MoG mechanism prioritizes the chunks retrieved from optimal granularity level, which is determined by the router based on the user input query.

tion (l_{bce}):

$$l_{bce} = \sum_{i \in \text{len}(w)} -[sl_i \cdot \log(w_i) + (1-sl_i) \cdot \log(1-w_i)] \quad (3)$$

3.3 MoGG: MoG with Graph-context

With MoG, the adjacent snippets with relevant knowledge can be retrieved altogether by adjusting the granularity level. This method is particularly effective when information centered around the same topic is stored in adjoining sentences. However, in most cases, answering a complex question requires reasoning over information stored in different paragraphs or even different documents. A common solution is to perform more retrievals at a finer granularity level, retrieving only highly relevant small pieces to form a comprehensive reasoning chain. This approach is inconvenient because determining the optimal number of retrievals often involves manually adjusting k , which is challenging because the whole tuning process is time-consuming and k can not go infinitely large.

To overcome this challenge, an intuitive approach is to reorganize the reference document and group the relevant information together. Motivated by this, we propose a more applicable framework, MoGG, by enhancing MoG with a preprocessing step that organizes the documents in \mathcal{K} as a graph. As illustrated in Figure 3, each document is initially split into small pieces consisting of one or two sentences, and each piece is treated as a separate node in the graph. To determine the edges in the graph, an index is first created with all these

nodes. Then, each node is used as a “query” to search for the k_{graph} (set as 3 in Figure 3) most relevant nodes using BM25. An edge is then added between two nodes if the similarity between them meets a predefined threshold T_{graph} .

This pre-processing extends the concept of “context” in linear text to “neighbors” in a graph. Also, granularity levels are adapted to “hopping ranges”: in a non-graph setting, a larger granularity level corresponds to more adjacent sentences being grouped in a chunk; in a graph setting, a larger granularity level corresponds to nodes within a larger hopping range of a centered node being grouped as one chunk. To avoid context redundancy, duplicate nodes in the neighbors are considered only once. Similar to MoG, the documents in the external knowledge database \mathcal{K} are chunked with n_{gra} different hopping ranges and then encoded into embeddings, while the rest of the MoG remains unchanged. By transforming the documents into a graph and defining granularity based on hopping ranges, MoGG effectively captures dispersed relevant information, allowing for more comprehensive and efficient retrieval.

4 Experiments

4.1 Corpus and Medical QA datasets

A reference knowledge database used in a RAG system is often termed a “corpus”. To form the corpora, data from various sources were collected, including the widely-used PubMed (Galileo Mark Namata and Huang, 2012) corpus for all

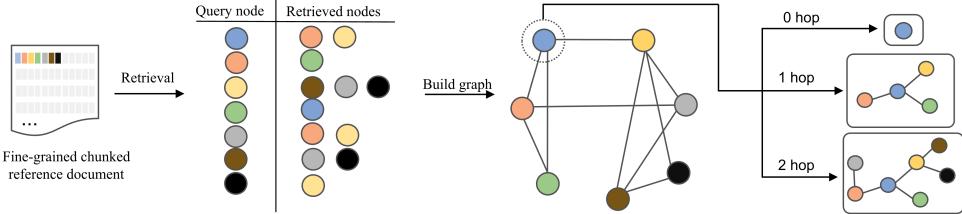


Figure 3: Pre-processing the reference document to form graphs. The concept of “granularity level” is changed into “hopping range” in graphs.

biomedical abstracts, the StatPearls (Publishing, 2024) corpus for clinical decision support, the Textbooks(Jin et al., 2020) corpus covering medical textbook knowledge, and the Wikipedia (Wikimedia Foundation, 2024) corpus for general knowledge. The four corpora are combined to form a larger corpus, named “MedCorp”. For each corpus used in our experiments, there are 5 granularity levels tested. The chunking size of the second granularity level is set to be the same as the one in MedRAG to facilitate the comparison of the results. The chunking size of the {1st, 3rd, 4th, 5th} granularity level is set to be {1/2, 2, 4, 8} times of the one in MedRAG.

The performance of the RAG system are evaluated using five MQA datasets following MIRAGE benchmark (Xiong et al., 2024): MMLU-Med (Hendrycks et al., 2021), MedQA-US (Zhang et al., 2018), MedMCQA (Pal et al., 2022), PubMedQA* (Jin et al., 2019), and BioASQ-Y/N (Tsatsaronis et al., 2015). Specifically, only the biomedical questions are kept and the ground truth supporting contexts are removed during testing.

The detailed descriptions of each copora and each QA dataset, as well as their important statistics, are included in Appendix B. We have begun to test our method using medical question-answering datasets, as they represent a knowledge-intensive domain. We posit that significant improvements demonstrated by the tests on this knowledge-intensive field suggest MoG’s potential effectiveness in other domains with lower knowledge dependencies and higher error tolerances.

4.2 Experiment Setup

All backbone LLMs, whether accessed via API or local deployment, are run under off-the-shelf settings. The exact versions of the backbone LLMs are listed in Appendix C. Experiments are conducted on Nvidia GeForce 3090 and 4090 GPUs. The code is written using the PyTorch framework, utilizing an Adam optimizer with a learning rate

of 0.001. Each training job is run until the convergence of the loss value. During the experiment, there are two top-k selections. Unless specified, when retrieving snippets from each corpus, we select the top-3 snippets; when all the snippets are retrieved from each corpus, we select the top-2 snippets with the highest relevance scores to pass to the backbone LLM. When snippets are too long, they are truncated automatically to fit the LLM’s context window size. The router requires approximately 12GB of GPU memory for training and 6GB for inference. Utilizing a caching mechanism, we efficiently completed 35 training sessions and over a hundred inferences, each training session taking around 4 hours for 1000 epochs.

4.3 Performance of MoG on MQA Task

As mentioned above, we test the effectiveness of MoG on MQA datasets. For each question, the RAG system is tasked with choosing the best answer(s) from the given options. The performance of the entire RAG system is measured by the Exact Matching accuracy of the answers. To prevent knowledge leakage, only the question is used (options excluded) to retrieve reference documents from the external knowledge database. The router of MoG guides the retrieval system to choose the optimal granularity. Qualitatively speaking, when tested on different datasets, the router trained with Textbook corpus shows a preference for different granularity levels. For instance, on the PubMedQA dataset, the finest granularity snippets are selected most frequently. This is because the questions in PubMedQA are typically precise and can be answered with short reference snippets. An example result figure is shown in Figure 4, with a more detailed discussion included in Appendix D.

MoG is integrated into one same RAG system, with which we conduct the MQA task. Backbone LLMs are altered to cover some of the popular ones, such as ChatGPT (Brown et al., 2020), InternLM2 (Cai et al., 2024), Llama3 (AI, 2024), GLM3 (Du

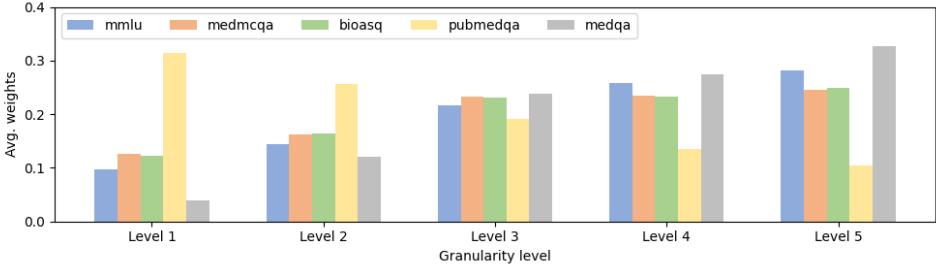


Figure 4: Averaged weights of different granularity levels on different QA datasets

LLM	Method	MIRAGE Benchmark Dataset (Acc.)					
		MMLU	MedQA	MedMCQA	PubMedQA	BioASQ	Avg.
GLM3	CoT	0.4356±0.04	0.3451±0.03	0.3250±0.02	0.3400±0.05	0.5081±0.04	0.3908
	MedRAG	0.4950±0.04	0.3804 ±0.03	0.3632±0.02	0.5100±0.05	0.6532±0.04	0.4804
	MoG	0.5198 ±0.04	0.3529±0.03	0.3716 ±0.02	0.5400 ±0.05	0.6774 ±0.04	0.4923
GPT-3.5	CoT	0.7525 ±0.03	0.6118±0.03	0.5890 ±0.02	0.5200±0.05	0.7339±0.04	0.6571
	MedRAG	0.6931±0.03	0.6510 ±0.03	0.4671±0.02	0.6000±0.05	0.8306 ±0.03	0.6484
	MoG	0.7129±0.03	0.6471±0.03	0.5532±0.02	0.6200 ±0.05	0.7823±0.04	0.6631
InternLM	CoT	0.7426 ±0.03	0.6118 ±0.03	0.5269±0.02	0.3500±0.05	0.7258±0.04	0.5914
	MedRAG	0.6040±0.04	0.5294±0.03	0.3847±0.03	0.4300±0.05	0.7661 ±0.04	0.5428
	MoG	0.7129±0.03	0.5961±0.03	0.5436 ±0.02	0.4100 ±0.05	0.7661 ±0.04	0.6057
Llama3	CoT	0.7079±0.03	0.6431 ±0.03	0.5663 ±0.02	0.5500±0.05	0.7258±0.04	0.6386
	MedRAG	0.6040±0.03	0.5725±0.03	0.4313±0.02	0.5600±0.05	0.7823±0.04	0.5900
	MoG	0.7228 ±0.03	0.6000±0.03	0.5627±0.02	0.6400 ±0.05	0.7984 ±0.04	0.6648
Qwen1.5	CoT	0.4604±0.04	0.3255±0.03	0.3883±0.02	0.2000±0.03	0.5484±0.04	0.3845
	MedRAG	0.5594±0.03	0.4353 ±0.03	0.4038±0.02	0.3400±0.05	0.5403±0.04	0.4558
	MoG	0.5941 ±0.04	0.4235±0.03	0.4301 ±0.02	0.4700 ±0.05	0.6694 ±0.04	0.5174

Table 1: Accuracy of Medical Question-Answering task with MoG (trained with MedCorp), best results in **bold**.

et al., 2022), and Qwen1.5 (Bai et al., 2023). The router is trained with the soft labels built using RoBERTa (Liu et al., 2019), this choice is justified by the experiment in Appendix E. To investigate the effect of varying the number of candidate snippets on RAG system performance, we conducted experiments with different snippet counts (details are provided in Appendix F). The retriever is fixed as BM25 (Robertson and Zaragoza, 2009), with a further discussion on the performance of different retrievers included in Appendix G. In Table 1 we present the results obtained with the router trained with MedCorp corpus (the results obtained with routers trained on four single corpora are presented in Appendix H). The results are compared with two baselines: CoT and MedRAG. CoT baseline adopts Chain-of-Thought (Wei et al., 2022) prompting and does not leverage any external knowledge database to enhance its response. MedRAG baseline is a simple RAG system with only 1 candidate granularity level introduced in MedRAG paper (Xiong et al., 2024).

The results demonstrate that MoG consistently enhanced the performance of the RAG system across different backbone models when compared with MedRAG, though not necessarily better than CoT. The reason is that the RAG system we used has no noise filters or any quality control mechanism, thus the noise is injected along with the knowledge via prompts. A detailed analysis of the number of samples improved or degraded by the application of MoG is included in the Appendix I, in which we manually verified that the majority of degradation is caused by noise. We also find that MoG improves the accuracy score more when applied on weaker LLMs (like ChatGLM and Qwen), probably because they have less knowledge stored in their internal parameters and, thus could benefit more from the retrieved snippets.

4.4 Performance of MoGG on MQA Task

Similarly to the experiment of MoG, we test the performance of MoGG on MQA datasets. We tested only the performance of MoGG with routers trained on Textbooks and StatPearls corpora because train-

LLM	Method	MIRAGE Benchmark Dataset (Acc.)					
		MMLU	MedQA	MedMCQA	PubMedQA	BioASQ	Avg.
GLM3	CoT	0.4356 \pm 0.04	0.3451 \pm 0.03	0.3250 \pm 0.02	0.3400 \pm 0.05	0.5081 \pm 0.04	0.3908
	MedRAG	0.4802 \pm 0.04	0.3569 \pm 0.03	0.3811 \pm 0.02	0.3600 \pm 0.05	0.5565 \pm 0.04	0.4269
	MoG	0.5545 \pm 0.04	0.2941 \pm 0.03	0.3548 \pm 0.02	0.4700 \pm 0.05	0.5726 \pm 0.04	0.4492
	MoGG	0.5347 \pm 0.04	0.3176 \pm 0.03	0.3608 \pm 0.02	0.4500 \pm 0.05	0.5645 \pm 0.04	0.4455
GPT-3.5	CoT	0.7525 \pm 0.03	0.6118 \pm 0.03	0.5890 \pm 0.02	0.5200 \pm 0.05	0.7339 \pm 0.04	0.6571
	MedRAG	0.7277 \pm 0.03	0.6745 \pm 0.03	0.4468 \pm 0.02	0.2600 \pm 0.04	0.5161 \pm 0.04	0.5250
	MoG	0.7525 \pm 0.03	0.6667 \pm 0.03	0.5603 \pm 0.02	0.5200 \pm 0.05	0.7016 \pm 0.04	0.6122
	MoGG	0.7673 \pm 0.03	0.6784 \pm 0.03	0.5233 \pm 0.02	0.4700 \pm 0.05	0.6452 \pm 0.04	0.6168
InternLM	CoT	0.7426 \pm 0.03	0.6118 \pm 0.03	0.5269 \pm 0.02	0.3500 \pm 0.05	0.7258 \pm 0.04	0.5914
	MedRAG	0.6188 \pm 0.03	0.5569 \pm 0.03	0.3118 \pm 0.02	0.1400 \pm 0.03	0.4839 \pm 0.04	0.4223
	MoG	0.7277 \pm 0.03	0.5725 \pm 0.03	0.5281 \pm 0.02	0.3400 \pm 0.05	0.7339 \pm 0.04	0.5804
	MoGG	0.7228 \pm 0.03	0.5882 \pm 0.03	0.5173 \pm 0.02	0.3400 \pm 0.05	0.7339 \pm 0.04	0.5804
Llama3	CoT	0.7079 \pm 0.03	0.6431 \pm 0.03	0.5663 \pm 0.02	0.5500 \pm 0.05	0.7258 \pm 0.04	0.6386
	MedRAG	0.6485 \pm 0.03	0.5961 \pm 0.03	0.4146 \pm 0.02	0.3800 \pm 0.05	0.5242 \pm 0.04	0.5127
	MoG	0.7228 \pm 0.03	0.6196 \pm 0.03	0.5484 \pm 0.02	0.5100 \pm 0.05	0.7097 \pm 0.04	0.6221
	MoGG	0.7030 \pm 0.03	0.5961 \pm 0.03	0.5460 \pm 0.02	0.5200 \pm 0.05	0.7661 \pm 0.04	0.6262
Qwen1.5	CoT	0.4604 \pm 0.04	0.3255 \pm 0.03	0.3883 \pm 0.02	0.2000 \pm 0.03	0.5484 \pm 0.04	0.3845
	MedRAG	0.5941 \pm 0.03	0.4000 \pm 0.03	0.3835 \pm 0.02	0.3300 \pm 0.05	0.4919 \pm 0.04	0.4399
	MoG	0.5792 \pm 0.03	0.3843 \pm 0.03	0.4110 \pm 0.02	0.3300 \pm 0.05	0.6129 \pm 0.04	0.4635
	MoGG	0.5594 \pm 0.03	0.4314 \pm 0.03	0.4480 \pm 0.02	0.3000 \pm 0.05	0.6371 \pm 0.04	0.4752

Table 2: Accuracy of Medical Question-Answering task with MoGG (trained with Textbooks), best results in **bold**.

ing with the other two much larger corpora is too time-consuming. In Table 2 we present the results obtained when trained with Textbook corpus (results obtained with StatPearls showed in Appendix J with similar patterns). From the table, we can tell that MoGG can further improve the averaged accuracy scores. By comparing Table 1 and Table 2, we can find that, even when trained with significantly fewer samples (Textbooks corpus is a tiny subset of MedCorp corpus, accounting for only about 0.2% of all the snippets in MedCorp), MoGG brings more significant improvement in terms of the averaged accuracy score with respect to MedRAG than MoG. This finding highlights that MoGG is more efficient than MoG thanks to its flexible way of organizing the reference snippets (in the form of a graph). We conducted an ablation test with Llama 3 and the results are included in Appendix K.

There is a general performance drop in the metrics in Table 2 compared to Tabel 1 because the Textbooks corpus is only a tiny subset of the Med-Corp used in Table 1). To facilitate the comparison, we conducted the same experiment in Table 1 with only the Textbooks corpus, and the detailed results are reported in Appendix L.

4.5 Execution time and storage efficiency

MoG(G) is proposed to increase the precision and recall of the retrieval phase at a reasonable cost

of computational efficiency because the quality of the retrieved chunks is prioritized for application scenarios like the medical environment. We measured the average inference time with the different number of candidate granularity levels, and the results show that increasing the number of granularity levels will only increase a marginal increase in execution time. (Details in Appendix M)

In terms of storage, while additional space is required to store the embeddings of the corpus at different granularities, we only need to store one copy of the corpus and five sets of embeddings. This engineering optimization results in a space requirement of only 2.7 times the size of the original corpus. We believe this represents an acceptable overhead. Furthermore, in our RAG system specialized in the medical domain, a corpus containing about 10GB of plain text can already cover a wide range of questions. Thus, we believe this will not be an obstacle to MoG(G)'s wider application.

We believe this overhead of computing resources added is worthwhile because in most cases, MoG performs much better than MedRAG and CoT baselines. On smaller models, MoG shows an average improvement of 5% compared to MedRAG and 8.7% compared to CoT. Given the dataset size of approximately 7000, these improvements are statistically and practically significant.

5 Limitations and Broader Impacts

Our work serves as an early trial in dynamic chunking strategy, with the following major directions for improvement. (1) MoG(G)'s candidate granularity levels are manually assigned. It could be more efficient if an algorithm automatically set these granularity levels to avoid excessive grid-searching for parameter optimization. (2) Current router uses only the semantic information of the input query to predict the best granularity level. (3) Previous studies have demonstrated that the use of length normalization is crucial in information retrieval-related fields. This paper primarily focuses on applying the concept of Mix-of-Experts to this application scenario. In future work, we will also take related issues into consideration. Incorporating more information (like query type or expected response length) into the router can potentially improve the results. However, the router also introduces a new security risk: a compromised router could redirect knowledge retrieval to malicious sources, injecting incorrect or even harmful information into the backbone LLM. Therefore, it is crucial to protect and monitor the router to mitigate this risk.

6 Conclusion

In this work, we present MoG, a mechanism to dynamically choose the best granularity when retrieving information from an external knowledge database. When applied to a RAG system, MoG helps retrieve more relevant information while reducing noise. MoG is further extended as MoGG, where reference documents are pre-processed as graphs. This extension allows distantly situated information to be retrieved simultaneously, overcoming the limitations of a fixed top- k selection strategy. Finally, we introduce a soft label guided loss function to address the difficulty of backward propagation with top- k selection, which could benefit future research.

Acknowledgements

We would like to express our gratitude to all the supervisors and colleagues at Shanghai Artificial Intelligence Laboratory for their invaluable insights, feedback, and support throughout the research process. We also thank the reviewers for their constructive comments, which greatly improved the quality of this paper.

References

- Meta AI. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta document*.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingen Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. *Preprint*, arXiv:2112.04426.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Zheng Cai, Maosong Cao, Haojong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Ying Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report. *Preprint*, arXiv:2403.17297.
- Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. 2022. Towards understanding mixture of experts in deep learning. *Preprint*, arXiv:2208.02813.
- Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. *Preprint*, arXiv:2004.07180.
- Neo4j Company. 2012. Neo4j - the world’s leading graph database.
- Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.
- Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhipang Sui. 2023. A survey on in-context learning. *Preprint*, arXiv:2301.00234.
- Zhe Dong, Jianmo Ni, Daniel M. Bikel, Enrique Alfonsena, Yuan Wang, Chen Qu, and Imed Zitouni. 2022. Exploring dual encoder architectures for question answering. *Preprint*, arXiv:2204.07120.
- Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhang Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. *Preprint*, arXiv:2103.10360.
- Lise Getoor Galileo Mark Namata, Ben London and Bert Huang. 2012. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, Edinburgh, Scotland.
- 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. *Preprint*, arXiv:2312.10997.
- Qipeng Guo, Zhijing Jin, Xipeng Qiu, Weinan Zhang, David Wipf, and Zheng Zhang. 2020. Cyclegt: Unsupervised graph-to-text and text-to-graph generation via cycle training. *Preprint*, arXiv:2006.04702.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

- Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. 2017. [Systematic integration of biomedical knowledge prioritizes drugs for repurposing](#). *eLife*, 6:e26726.
- Jie Huang, Wei Ping, Peng Xu, Mohammad Shoeybi, Kevin Chen-Chuan Chang, and Bryan Catanzaro. 2024. [Raven: In-context learning with retrieval-augmented encoder-decoder language models](#). *Preprint*, arXiv:2308.07922.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022a. [Unsupervised dense information retrieval with contrastive learning](#). *Preprint*, arXiv:2112.09118.
- Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022b. [Atlas: Few-shot learning with retrieval augmented language models](#). *Preprint*, arXiv:2208.03299.
- Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. [Active retrieval augmented generation](#). *Preprint*, arXiv:2305.06983.
- Di Jin, Eileen Pan, Nassim Oufattolle, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. [What disease does this patient have? a large-scale open domain question answering dataset from medical exams](#). *Preprint*, arXiv:2009.13081.
- Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. [Pubmedqa: A dataset for biomedical research question answering](#). *Preprint*, arXiv:1909.06146.
- Qiao Jin, Won Kim, Qingyu Chen, Donald C Comeau, Lana Yeganova, W John Wilbur, and Zhiyong Lu. 2023. [Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval](#). *Bioinformatics*, 39(11).
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. [Billion-scale similarity search with gpus](#). *Preprint*, arXiv:1702.08734.
- Minki Kang, Jin Myung Kwak, Jinheon Baek, and Sung Ju Hwang. 2023. [Knowledge graph-augmented language models for knowledge-grounded dialogue generation](#). *Preprint*, arXiv:2305.18846.
- YuHe Ke, Liyuan Jin, Kabilan Elangovan, Hairil Rizal Abdullah, Nan Liu, Alex Tiong Heng Sia, Chai Rick Soh, Joshua Yi Min Tung, Jasmine Chiat Ling Ong, and Daniel Shu Wei Ting. 2024. [Development and testing of retrieval augmented generation in large language models – a case study report](#). *Preprint*, arXiv:2402.01733.
- Assia Khan. 2023. [Retrieval augmented generation: 5 uses and their examples](#). *Lettria*.
- Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. [Generalization through memorization: Nearest neighbor language models](#). *Preprint*, arXiv:1911.00172.
- Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2023. [Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp](#). *Preprint*, arXiv:2212.14024.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- Han Liang, Wenqian Zhang, Wenxuan Li, Jingyi Yu, and Lan Xu. 2024. [Intergen: Diffusion-based multi-human motion generation under complex interactions](#). *International Journal of Computer Vision*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#). *Preprint*, arXiv:1907.11692.
- Igor Melnyk, Pierre Dognin, and Payel Das. 2022. [Knowledge graph generation from text](#). *Preprint*, arXiv:2211.10511.
- Sosuke Nishikawa, Ryokan Ri, Ikuya Yamada, Yoshimasa Tsuruoka, and Isao Echizen. 2022. [Ease: Entity-aware contrastive learning of sentence embedding](#). *Preprint*, arXiv:2205.04260.
- Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. [Medmcqa : A large-scale multi-subject multi-choice dataset for medical domain question answering](#). *Preprint*, arXiv:2203.14371.
- StatPearls Publishing. 2024. Statpearls. Treasure Island (FL): StatPearls Publishing. Available from: <https://www.ncbi.nlm.nih.gov/books/NBK430685/>.
- Irina Radeva, Ivan Popchev, Lyubka Doukovska, and Miroslava Dimitrova. 2024. [Web application for retrieval-augmented generation: Implementation and testing](#). *Electronics*, 13:1361.
- Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. [In-context retrieval-augmented language models](#). *Preprint*, arXiv:2302.00083.
- Juan Enrique Ramos. 2003. [Using TF-IDF to determine word relevance in document queries](#).

- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Found. Trends Inf. Retr.*, 3(4):333–389.
- Krystian Safjan. 2023. From fixed-size to nlp chunking – a deep dive into text chunking techniques. *Krystian’s Safjan Blog*.
- Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. [Raptor: Recursive abstractive processing for tree-organized retrieval](#). *Preprint*, arXiv:2401.18059.
- Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. [Replug: Retrieval-augmented black-box language models](#). *Preprint*, arXiv:2301.12652.
- Ryan Siegler. 2024. Optimizing vector search with metadata filtering. *KX systems*.
- Sumit Soman and Sujoy Roychowdhury. 2024. [Observations on building rag systems for technical documents](#). *Preprint*, arXiv:2404.00657.
- Antematter team. 2024. Optimizing retrieval-augmented generation with advanced chunking techniques: A comparative study. *Antematter*.
- LangChain team. 2023. Parent document retriever. *LangChain document*.
- George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael Alvers, Dirk Weißenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artieres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, and Georgios Palioras. 2015. [An overview of the bioasq large-scale biomedical semantic indexing and question answering competition](#). *BMC Bioinformatics*, 16:138.
- Haotao Wang, Ziyu Jiang, Yuning You, Yan Han, Gaowen Liu, Jayanth Srinivasa, Ramana Rao Komella, and Zhangyang Wang. 2023a. [Graph mixture of experts: Learning on large-scale graphs with explicit diversity modeling](#). *Preprint*, arXiv:2304.02806.
- Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023b. [Self-knowledge guided retrieval augmentation for large language models](#). *Preprint*, arXiv:2310.05002.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.
- Wikimedia Foundation. 2024. Wikipedia. The Free Encyclopedia. Available from: <https://www.wikipedia.org>.
- Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. [Benchmarking retrieval-augmented generation for medicine](#). *Preprint*, arXiv:2402.13178.
- Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. [Graphrnn: Generating realistic graphs with deep auto-regressive models](#). *Preprint*, arXiv:1802.08773.
- Xiao Zhang, Ji Wu, Zhiyang He, Xien Liu, and Ying Su. 2018. [Medical exam question answering with large-scale reading comprehension](#). *Preprint*, arXiv:1802.10279.

A More Discussion About Selection Process

The intuitive selection process would be to select the top snippets with the highest weighted similarity scores. However, this method is not adopted for the following reasons:

(1) It is biased, because more-coarsened snippets tend to have higher similarity scores, even though they might include more noise.

(2) It is not robust to rely on the router’s output weight to control such imbalance between granularity levels, because the router is not trained for this objective. The router is trained with soft labels, which are approximate training signals that can only teach the model to distinguish the 1st and 2nd optimal granularity level from the rest.

(3) In practice, when \mathcal{K} is too large, it has to be stored in separated vector databases. A common solution is to use one vectorbase to store one granularity. This solution will make the similarity scores across granularity levels NOT directly comparable with each other.

Based on these reasons, we designed the selection strategy presented in the paper. The rationale behind the proposed strategy is that: Since we aim to extract the globally optimal snippet, we need a way to calculate a “global highest similarity score”. Given that the finest-grained snippets are the only ones that appear across all granularity levels following our initial chunking approach (fixed-size, non-overlapping), we calculate the weighted sum based on these finest-grained snippets.

B Details of QA Datasets

B.1 MMLU-Med

The Massive Multitask Language Understanding (MMLU) benchmark (Hendrycks et al., 2021) evaluates the multitask learning capability of language models. While the full MMLU dataset encompasses 57 different tasks, we specifically extracted the medical questions for our tests, totaling 1089 questions.

B.2 MedQA-US

MedQA (Zhang et al., 2018) is a multiple-choice QA dataset derived from professional medical board exams. It is available in Simplified Chinese, Traditional Chinese, and English. For our experiments, we used 1273 questions from the English version.

B.3 MedMCQA

MedMCQA (Pal et al., 2022) comprises a large number of questions from the Indian medical entrance exam, covering 2400 healthcare topics and 21 medical subjects. For the MIRAGE benchmark, we utilized the “dev” set of the original MedMCQA dataset.

B.4 PubMedQA*

PubMedQA is a research QA dataset in the biomedical field, consisting of 1000 manually annotated questions constructed from PubMed abstracts. In the MIRAGE benchmark, the reference contexts were removed. We selected a subset of 500 questions, which we refer to as PubMedQA*.

B.5 BioASQ-Y/N

BioASQ (Tsatsaronis et al., 2015) is an annual biomedical QA competition. For the MIRAGE benchmark, we selected only the Machine Reading Comprehensive Track (Task B), focusing on 618 questions from recent years (2019-2023).

The important statistics of the corpora and the QA datasets are present in Table 3 below.

C Exact Versions of LLMs

The exact versions of the backbone LLMs tested are listed in Table 4.

D Qualitative Results of the Router

As illustrated in Figure 4, the router effectively assigns different weights to various granularity levels. From the figure, we can infer that a potential global peak in weights may exist at a granularity level smaller than level 1 or larger than level 5.

However, we did not test these smaller or larger granularity levels due to the following reasons: In our experiments, we focus exclusively on LLMs with a parameter size of around 7 billion. For these models, the chunking size at granularity level 5 approaches their maximum context window. Conversely, the chunking size at granularity level 1 consists of only a few dozen characters, which is already quite small.

E Experiment on Soft Labels

In this section, we evaluate the performance of building soft labels using different methods (TF-IDF (Ramos, 2003), RoBERTa (Liu et al., 2019), and hitrate score). For this experiment, we fix one retriever (BM25) and the backbone LLM

Corpus	#Doc	#Snip.	\bar{L} .	Domain	Datasets	Size	#Opt.	\bar{L}	Source
PubMed	23.9M	23.9M	196	Bio.-Med.	MMLU-Med	1,089	4	63	Exam.
StatPearls	9.3k	301.2k	119	Clinics	MedQA-US	1,273	4	177	Exam.
Textbooks	18	125.8k	182	Medicine	MedMCQA	4,183	4	26	Exam.
Wikipedia	6.5M	29.9M	162	General	PubMedQA*	500	3	24	Literature
MedCorp	30.4M	54.2M	221	Mixed	BioASQ-Y/N	618	2	17	Literature

Table 3: Important statistics of corpora and QA datasets

(Qwen1.5), then build the soft labels using these three different methods. Different routers are trained with these different soft labels and tested on various MQA datasets. The results are grouped in Table 5.

The results indicate that there is no universal best method for building soft labels. For instance, “hitrate score” is the most effective when training on MedMCQA, while RoBERTa shows advantages on PubMedQA. Overall, the soft labels built with RoBERTa demonstrate good performance across the board. Therefore, for the remainder of the experiments, RoBERTa is adopted as the default method to build soft labels.

F Impact of the Number of Candidate Snippets

In the previous experiment, three candidate snippets were retrieved from each granularity level of the external knowledge database. In this section, we investigate the effect of varying the number of candidate snippets on the overall performance of the RAG system. The rationale behind this exploration lies in the potential limitation of a small pool of candidate snippets, such as keeping only three. In such case, valuable snippets may be overlooked. For instance, a snippet ranked fourth or fifth in each retrieval may appear repeatedly across different granularity levels and thus might be selected after its relevance scores adjusted with weights assigned by the router. However, despite potentially high relevance, such snippets are excluded early in the retrieval process.

In this experiment, we test the RAG system equipped with MoG using different numbers of candidate snippets k_r ($k_r \in \{3, 8, 16, 32\}$). For simplicity, the backbone LLM was fixed as Qwen1.5. The experiment results are shown in Figure 5.

Generally, RAG performance improves as k_r increases, indicating the presence of helpful knowledge in the retrieved snippets. However, too many irrelevant snippets mislead the inference of LLM based on its knowledge. MoG stands out with

its high initial performance even at low k_r values ($k_r <= 8$ in this case), thanks to its multi-granularity filtering, which efficiently selects relevant snippets. This enhances the LLM’s accuracy without requiring a large snippet pool. MoG even strikes a balance at high k_r values ($k_r >= 16$ in this case) due to its threshold limiting of the different corpora, avoiding the pitfalls of excessive information retrieval. MoG’s consistently high performance demonstrates its superiority in optimizing the number of candidate snippets for effective medical question-answering.

G Choice of Retriever

In the previous experiment, BM25 (Robertson and Zaragoza, 2009) was used as the retriever because it is a lightweight and popular choice in practice. In this section, we replace BM25 with other popular retrievers to evaluate their performance. Referring to the setup in MedRAG (Xiong et al., 2024), we select a general-domain semantic retriever called Contriever (Izacard et al., 2022a), a scientific-domain retriever called SPECTER (Cohan et al., 2020), and a biomedical-domain retriever called MedCPT (Jin et al., 2023). Additionally, the Reciprocal Rank Fusion (RRF) method (Cormack et al., 2009) is utilized to combine results from different retrievers, including RRF-2 (fusion of BM25 and MedCPT) and RRF-4 (fusion of all four retrievers). From Table 6, we can observe the following: Each retriever has a well-performing QA set, which results in minimal overall differences among these retrievers. Considering these conclusions, we decide to continue using BM25 for all other experiments.

H Performance of MoG on Medical QA Task (Other results)

In this section, we present the experiment result of MoG on the Medical QA task with four different corpora as the training datasets for the router. The results are grouped in Table 7.

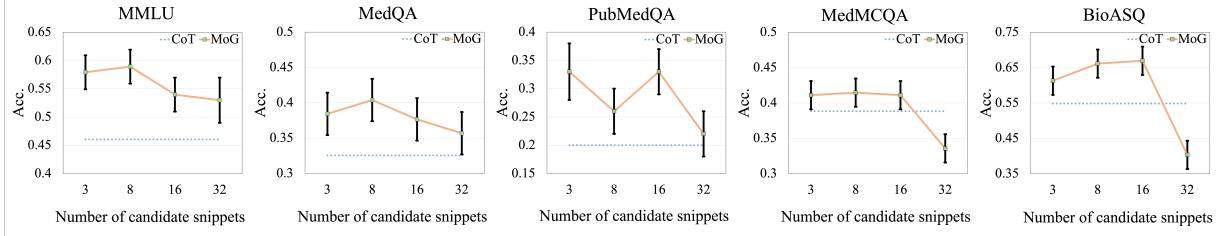


Figure 5: Accuracy of Medical Question-Answering task with different number of candidate snippets

I Analysis on Samples Improved or Degraded

We counted the number of samples of four categories before and after applying MoG. The four categories are 1. improved (CoT gives wrong answer, MoG gives correct answer); 2. degraded (CoT gives correct answer, MoG gives wrong answer); 3. remain_correct (both CoT and MoG give correct answers); 4. remain_wrong (both CoT and MoG give wrong answers). The results are visualized as Figure 6. In this presented figure, the MoG is trained on the Wikipedia corpus. The pattern observed from this figure is similar to the ones shown in other test results. The zeroes in bars named “cot_err” and “mog_err” indicate that all the responses are correctly parsed when counting.

Around 10% of the degraded samples are randomly chosen and manually verified. We confirmed that in most (95%) degraded cases being verified, all the following statements are true:

- several candidate snippets are retrieved correctly;
- top-2 candidate snippets are correctly selected with the weights calculated by the router;
- the prompt is correctly augmented and passed to backbone LLM;
- the LLM generates the response without error;
- the final choice (A/B/C/D or Yes/No) was correctly parsed.

In other words, the LLM changed its answer based on the snippet retrieved, it is infected by the introduced noises. The degradation is caused by the fact that the tested RAG system lacks a noise filtering mechanism, rather than by the default of MoG or MoGG.

J Performance of MoGG on Medical QA Task (Other results)

In this section, we present the experiment result of MoGG on the Medical QA task with StatPearls as the training corpus for the router. The results are grouped in Table 8.

K MoGG’s Abaltion test

In this section, we rerun the MoGG without the router on Llama 3 to highlight its effectiveness. The results of MoGG’s ablation test is in Table 9 (highlighted in bold). The results are similar to MedRAG, because, without the router, MoGG is essentially the same as MedRAG, the added flexibility of the graph will only show when used with a router.

L MoG Trained With Only Textbooks Corpus

In this Appendix, we present the results of training MoG only with Textbooks corpus. As Textbooks are only a subset of MedCorp, the performance is not as good as the ones shown in Table 1. Compared with Table 10 and Table 2, we can see the advantage of MoGG more clearly.

M Detailed Analysis of Time Consumption

In this appendix, we present the detailed experiment results of execution time. We measure the wall-clock execution time with different number of granularities using Llama3 as the backbone LLM. The results organized in Table 11 show that:

(1) the introduction of the router module will effectively increase the inference time by about 60%;

(2) increasing the number of granularity levels will only increase a marginal increase in execution time. The reason is that the bottleneck of execution time is the API calling time of backbone LLMs.

Exp 51905 Comparison of MoG w/ respect to CoT

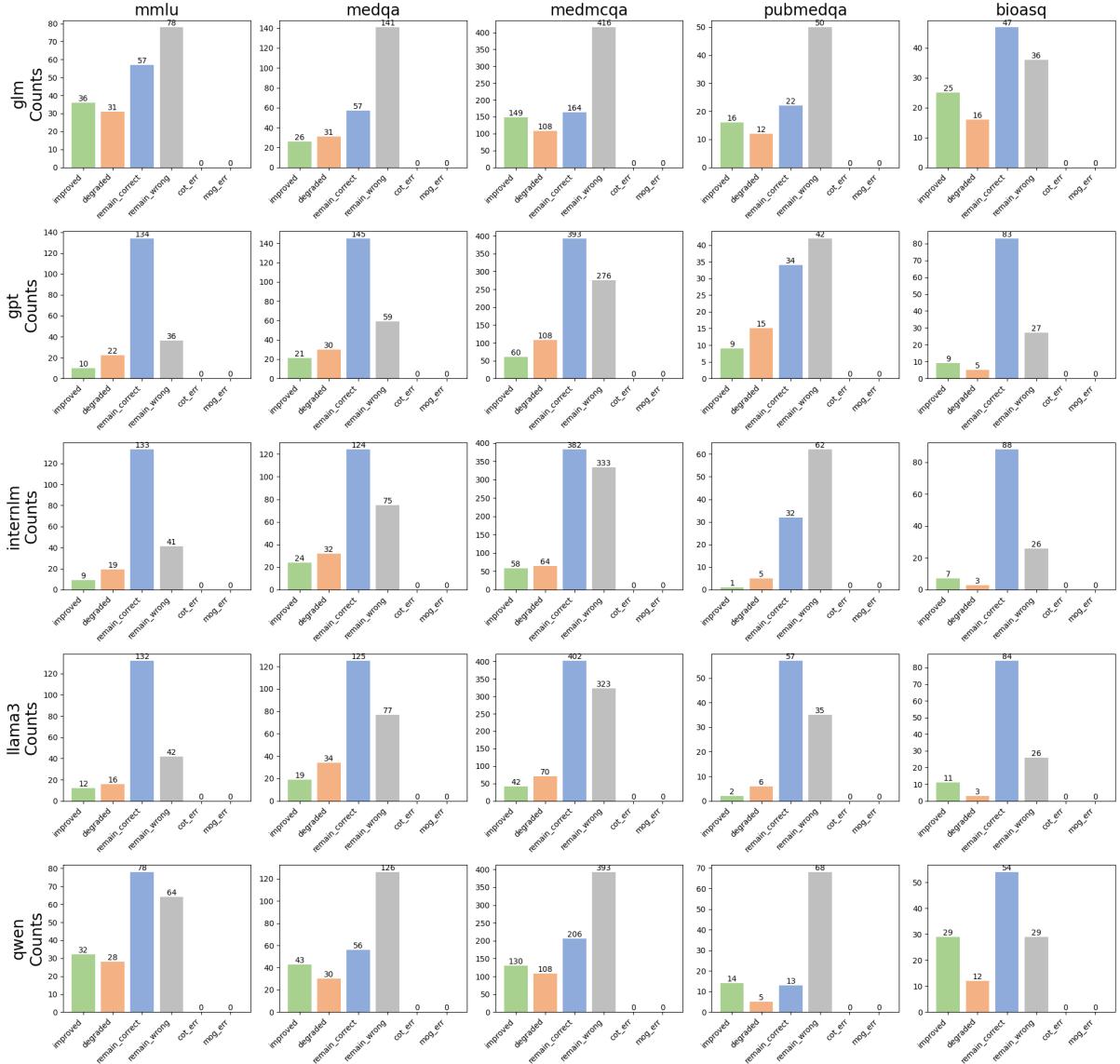


Figure 6: Number of samples improved or degraded after application of MoG

LLM name	LLM version	LLM site
ChatGPT	gpt-3.5-turbo-16k	https://platform.openai.com/docs/models/gpt-3.5-turbo
Llama3	Meta-Llama-3-8B	https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
InternLM2	internlm2-123b	https://www.sensetime.com/en/news-detail/51167237
ChatGLM3	chatglm3-6b	https://huggingface.co/THUDM/chatglm3-6b
Qwen1.5	Qwen1.5-MoE-A2.7B	https://huggingface.co/Qwen/Qwen1.5-MoE-A2.7B-Chat

Table 4: Versions of the backbone LLMs

Training datasets	Methods	MIRAGE Benchmark Dataset (Acc.)						Avg.
		MMLU	MedQA	MedMCQA	PubMedQA	BioASQ		
MedQA	RoBERTa	0.4265 \pm 0.04	0.3711 \pm 0.04	0.3831 \pm 0.02	0.5000 \pm 0.06	0.6234 \pm 0.06	0.4608	
	hitrate	0.4779 \pm 0.04	0.4088 \pm 0.04	0.4330 \pm 0.02	0.4194 \pm 0.06	0.6753 \pm 0.05	0.4829	
	TF-IDF	0.4559 \pm 0.04	0.3019 \pm 0.04	0.4023 \pm 0.02	0.4355 \pm 0.06	0.5714 \pm 0.06	0.4334	
MedMCQA	RoBERTa	0.4485 \pm 0.05	0.3459 \pm 0.04	0.4023 \pm 0.02	0.4839 \pm 0.06	0.5195 \pm 0.06	0.4400	
	hitrate	0.4118 \pm 0.04	0.3774 \pm 0.04	0.3678 \pm 0.02	0.3387 \pm 0.06	0.5065 \pm 0.06	0.4004	
	TF-IDF	0.4412 \pm 0.04	0.3899 \pm 0.04	0.4119 \pm 0.02	0.5161 \pm 0.06	0.5844 \pm 0.06	0.4687	
PubMedQA	RoBERTa	0.4485 \pm 0.04	0.4528 \pm 0.04	0.3966 \pm 0.02	0.4677 \pm 0.06	0.6623 \pm 0.05	0.4856	
	hitrate	0.4118 \pm 0.04	0.4088 \pm 0.04	0.3851 \pm 0.02	0.4355 \pm 0.06	0.6104 \pm 0.06	0.4503	
	TF-IDF	0.4044 \pm 0.04	0.3208 \pm 0.04	0.3908 \pm 0.02	0.4677 \pm 0.06	0.5584 \pm 0.06	0.4284	

Table 5: The performance of the routers trained with the soft labels created with different methods, best method marked in **bold**.

Corpus	Retriever	MIRAGE Benchmark Dataset (Acc.)						Avg.
		MMLU	MedQA	MedMCQA	PubMedQA	BioASQ		
Textbooks	BM25	0.6139 \pm 0.04	0.3961 \pm 0.04	0.3907 \pm 0.02	0.3100 \pm 0.06	0.5000 \pm 0.05	0.4421	
	Contriever	0.5644 \pm 0.03	0.4235 \pm 0.03	0.4074 \pm 0.01	0.2500 \pm 0.04	0.5161 \pm 0.04	0.4323	
	SPECTER	0.5941 \pm 0.03	0.4118 \pm 0.03	0.3919 \pm 0.01	0.1900 \pm 0.04	0.4839 \pm 0.04	0.4143	
	MedCPT	0.5347 \pm 0.04	0.4549 \pm 0.03	0.4026 \pm 0.02	0.3100 \pm 0.05	0.5565 \pm 0.04	0.4517	
	RRF-2	0.5396 \pm 0.03	0.4196 \pm 0.03	0.4241 \pm 0.02	0.2800 \pm 0.04	0.5403 \pm 0.04	0.4407	
	RRF-4	0.5495 \pm 0.04	0.4392 \pm 0.03	0.4397 \pm 0.02	0.2600 \pm 0.04	0.6129 \pm 0.04	0.4603	

Table 6: Performance of different retrievers

LLM	Method	Training Corpora (Avg Acc.)			
		Textbooks	StatPearls	PubMed	Wikipedia
GLM3	CoT	0.3908	0.3908	0.3908	0.3908
	MedRAG	0.4269	0.4343	0.4716	0.4582
	MoG	0.4492	0.4465	0.4911	0.4241
GPT-3.5	CoT	0.6571	0.6571	0.6571	0.6571
	MedRAG	0.5250	0.5229	0.6322	0.5332
	MoG	0.6122	0.5921	0.6795	0.6154
InternLM	CoT	0.5914	0.5914	0.5914	0.5914
	MedRAG	0.4223	0.4380	0.5559	0.4506
	MoG	0.5800	0.5886	0.6394	0.5810
Llama3	CoT	0.6386	0.6386	0.6386	0.6386
	MedRAG	0.5127	0.5133	0.6170	0.5206
	MoG	0.6221	0.6415	0.6394	0.6328
Qwen1.5	CoT	0.3845	0.3845	0.3845	0.3845
	MedRAG	0.4399	0.4623	0.4499	0.4391
	MoG	0.6129	0.4622	0.5145	0.4547

Table 7: Accuracy of Medical Question-Answering task with MoG (trained with different corpora)

LLM	Method	MIRAGE Benchmark Dataset (Acc.)					
		MMLU	MedQA	MedMCQA	PubMedQA	BioASQ	Avg.
GLM3	CoT	0.4901 \pm 0.04	0.3294 \pm 0.03	0.3799 \pm 0.02	0.3900 \pm 0.05	0.5645 \pm 0.04	0.4181
	MedRAG	0.4901 \pm 0.04	0.3294 \pm 0.03	0.3799 \pm 0.02	0.4400 \pm 0.05	0.5323 \pm 0.04	0.4343
	MoG	0.4851 \pm 0.04	0.2902 \pm 0.03	0.3799 \pm 0.02	0.4400 \pm 0.05	0.6371 \pm 0.04	0.4465
	MoGG	0.4802 \pm 0.04	0.3059 \pm 0.03	0.3668 \pm 0.02	0.4600 \pm 0.05	0.5806 \pm 0.04	0.4387
GPT-3.5	CoT	0.7624 \pm 0.03	0.6706 \pm 0.03	0.5866 \pm 0.02	0.4200 \pm 0.05	0.7339 \pm 0.04	0.6347
	MedRAG	0.6881 \pm 0.03	0.6549 \pm 0.03	0.4552 \pm 0.02	0.2600 \pm 0.04	0.5565 \pm 0.04	0.5229
	MoG	0.7277 \pm 0.03	0.6471 \pm 0.03	0.5400 \pm 0.02	0.3200 \pm 0.05	0.7258 \pm 0.04	0.5921
	MoGG	0.7624 \pm 0.03	0.6314 \pm 0.03	0.5460 \pm 0.02	0.3700 \pm 0.05	0.7177 \pm 0.04	0.6055
InternLM	CoT	0.7228 \pm 0.03	0.6000 \pm 0.03	0.5352 \pm 0.02	0.3700 \pm 0.05	0.7339 \pm 0.04	0.5924
	MedRAG	0.6584 \pm 0.03	0.5137 \pm 0.03	0.3596 \pm 0.02	0.1500 \pm 0.04	0.5081 \pm 0.04	0.4380
	MoG	0.6881 \pm 0.03	0.5961 \pm 0.03	0.5448 \pm 0.02	0.3800 \pm 0.05	0.7339 \pm 0.04	0.5886
	MoGG	0.6782 \pm 0.03	0.5725 \pm 0.03	0.3500 \pm 0.02	0.7500 \pm 0.05	0.7500 \pm 0.04	0.6201
Llama3	CoT	0.7277 \pm 0.03	0.6392 \pm 0.03	0.5663 \pm 0.02	0.5900 \pm 0.05	0.7177 \pm 0.04	0.6482
	MedRAG	0.6089 \pm 0.03	0.5882 \pm 0.03	0.4170 \pm 0.02	0.3800 \pm 0.05	0.5242 \pm 0.04	0.5133
	MoG	0.7376 \pm 0.03	0.5961 \pm 0.03	0.5317 \pm 0.02	0.6000 \pm 0.05	0.7419 \pm 0.04	0.6415
	MoGG	0.7030 \pm 0.03	0.6275 \pm 0.03	0.5436 \pm 0.02	0.5900 \pm 0.05	0.7258 \pm 0.04	0.6380
Qwen1.5	CoT	0.4604 \pm 0.04	0.3255 \pm 0.03	0.3883 \pm 0.02	0.2000 \pm 0.04	0.5484 \pm 0.04	0.3845
	MedRAG	0.5495 \pm 0.04	0.4471 \pm 0.03	0.4146 \pm 0.02	0.3600 \pm 0.05	0.5403 \pm 0.04	0.4623
	MoG	0.5446 \pm 0.04	0.3882 \pm 0.03	0.4337 \pm 0.02	0.3400 \pm 0.05	0.6048 \pm 0.04	0.4623
	MoGG	0.5446 \pm 0.04	0.4157 \pm 0.03	0.4253 \pm 0.02	0.3500 \pm 0.05	0.6290 \pm 0.04	0.4729

Table 8: Accuracy of Medical Question-Answering task with MoGG (trained with StatPearls), best results marked in **bold**

Method	MMLU	MedQA	MedMCQA	PubMedQA	BioASQ	Avg.
CoT	0.7079	0.6431	0.5663	0.5500	0.7258	0.6386
MedRAG	0.6485	0.5961	0.4146	0.3800	0.5242	0.5127
MoG	0.7228	0.6196	0.5484	0.5100	0.7097	0.6221
MoGG w/o router	0.6683	0.5451	0.4182	0.3800	0.5484	0.5120
MoGG	0.7070	0.5961	0.5460	0.5200	0.7661	0.6262

Table 9: Ablation test with Llama3 as backbone LLM

LLM	Method	MIRAGE Benchmark Dataset (Acc.)					
		MMLU	MedQA	MedMCQA	PubMedQA	BioASQ	Avg.
GLM3	MedRAG	0.4802	0.3569	0.3811	0.3600	0.5565	0.4269
	MoG	0.5545	0.2941	0.3548	0.4700	0.5726	0.4492
GPT-3.5	MedRAG	0.7277	0.6745	0.4468	0.2600	0.5161	0.5250
	MoG	0.7525	0.6667	0.5603	0.5200	0.7016	0.6122
InternLM	MedRAG	0.6188	0.5569	0.3118	0.1400	0.4839	0.4223
	MoG	0.7277	0.5725	0.5281	0.3400	0.7339	0.5804
Llama3	MedRAG	0.6485	0.5961	0.4146	0.3800	0.5242	0.5127
	MoG	0.7228	0.6196	0.5484	0.5100	0.7097	0.6221
Qwen1.5	MedRAG	0.5941	0.4000	0.3835	0.3300	0.4919	0.4399
	MoG	0.5792	0.3843	0.4110	0.3300	0.6129	0.4635

Table 10: Accuracy of Medical Question-Answering task with MoG (trained with only Textbooks)

# (Granularity)	MMLU	MedQA	MedMCQA	PubMedQA	BioASQ	Global Avg.
1 w/o router	6.61	9.02	10.01	7.40	7.53	8.12
1 w/ router	10.15	10.76	15.81	15.99	13.09	13.16
2	13.25	12.60	20.53	17.30	14.98	15.73
3	16.30	16.01	19.03	16.73	17.12	17.04
4	13.43	20.65	22.72	18.93	14.05	17.96
5	13.20	20.43	23.00	19.36	13.66	17.93

Table 11: Performance Metrics by Granularity