FUNNELRAG: A Coarse-to-Fine Progressive Retrieval Paradigm for RAG

Anonymous ARR submission

Abstract

Retrieval-Augmented Generation (RAG) prevails in Large Language Models. It mainly consists of *retrieval* and *generation*. The retrieval modules (*a.k.a.* retrievers) aim to find useful information used to facilitate generation modules (*a.k.a.* generators). As such, generators' performance largely depends on the effectiveness and efficiency of retrievers. However, the retrieval paradigm that we design and use remains flat, which treats the retrieval procedures as a one-off deal with constant granularity. Despite effectiveness, we argue that they suffer from two limitations: (1) **flat retrieval** exerts a significant burden on one retriever; (2) **constant granularity** limits the ceiling of retrieval performance. In this work, we propose a progressive retrieval paradigm with coarse-to-fine granularity for RAG, termed FUNNELRAG, so as to balance effectiveness and efficiency. Specifically, FUNNELRAG establishes a progressive retrieval pipeline by collaborating coarse-to-fine granularity, large-to-small quantity, and low-to-high capacity, which can relieve the burden on one retriever and also promote the ceiling of retrieval performance. Extensive experiments manifest that FUNNELRAG achieves comparable retrieval performance while the time overhead is reduced by nearly 40 percent.

1 Introduction

Retrieval-Augmented Generation (RAG) has been shown highly effective in enhancing Large Language Models (LLMs) (Gao et al., 2023; Shi et al., 2023) and has been widely adopted in the industry, such as Microsoft's GraphRAG (Edge et al., 2024), Google's REALM (Guu et al., 2020), and Meta's RA-DIT (Lin et al., 2024). Its effectiveness mainly comes from retrieving external non-parametric knowledge into LLMs to remedy their incomplete, incorrect, or outdated internal parametric knowledge (Karpukhin et al., 2020; Min et al., 2019). The de facto RAG framework usually segments documents into short retrieval units, such as 100-word passages (Jiang et al., 2024), resulting in a massive corpus with tens of millions of candidate units. Then, the retriever is tasked to find the "needle" (*i.e.,* the golden retrieval units) from the "haystack" (*i.e.,* the enormous candidate corpus) (Lee et al., 2024; Kamradt, 2023). Finally, the retrieved units serve as the input context to the generator to facilitate generation. Its working flow is shown in Figure 1(a). Wikipedia dump is used as the non-parametric knowledge source (Lewis et al., 2020), where each document is segmented into disjoint 100-word chunks, resulting in a total of 21M short passages. Then, the retriever seeks through a vast number of 21M candidates to get several potentially valuable passages, such as four.

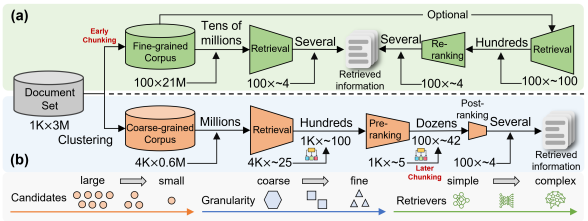


Figure 1: Comparison between **(a)** the flat retrieval and **(b)** the progressive retrieval paradigm, where is the segmentation operation. FUNNELRAG performs progressive retrieval from large to small quantity, from coarse to fine granularity, and with simple to complex retrievers, which balances effectiveness and efficiency.

Despite effectiveness, the existing retrieval paradigms still suffer from two major limitations:

- **Flat Retrieval.** Most RAG frameworks approach the retrieval stage as a one-off deal, where the retriever is requested to take tens of millions of candidates as input and to find the golden retrieval units at a sitting. However, these practices inflict a heavy burden on one retriever, which makes the retrieval stage less effective and efficient, since it is very tough and computationally complex to find golden units immediately from such an enormous candidate pool (*e.g.,* 21M).

Retrieval Paradigm	Retrieval Unit	Time to Chunk	Corpus Size	Unit Size
Flat Retrieval	Passage → Passage (Optional)	Early Chunking	21M	100
Progressive Retrieval	Clustered Document→ Document → Passage	Later Chunking	600K	4K

Table 1: Key feature comparison between the prevailing flat retrieval and the proposed progressive retrieval.

- **Constant Granularity.** Some RAG frameworks draw inspiration from recommender systems (Wang et al., 2020; Covington et al., 2016; Grbovic and Cheng, 2018), following a multi-stage cascade architecture, where candidates are extracted through *retrieval* and *reranking* as shown in Figure 1(a). However, these practices directly introduce recommender systems' experience into RAG and neglect the characteristics of RAG. Each entry in recommender systems is usually inseparable, while entries in RAG are usually separable. Unfortunately, existing RAG methods commonly treat entries as constant-grained retrieval units, ultimately restricting performance.

The above issues inflict a heavy burden on one retriever and neglect the subdivisible characteristic of entries in RAG. In this work, we propose a progressive retrieval paradigm with coarse-to-fine granularity for RAG, termed FUNNELRAG. Specifically, it progressively reduces the scale of candidate entries, refines the granularity of retrieval units, and increases the level of retrievers' capacity, whose working flow is shown in Figure 1(b). Particularly, there are two important designs in FUNNELRAG:

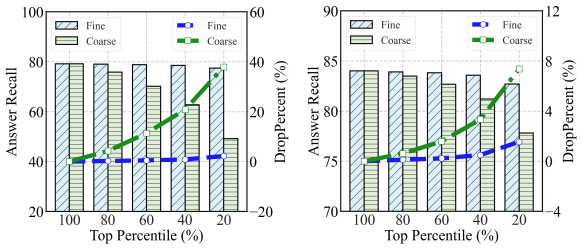
- **Progressive Retrieval.** Different from flat retrieval, we progressively reduce the scale of candidates (*e.g.,* millions (0.6M) → hundreds (100) → dozens (42)) and increase the level of retrievers' capacity (*e.g.,* sparse retrievers (SR)→ dense retrievers (DR)→small language models (SLM)), to make a better balance of effectiveness and efficiency. This design enables load balancing and improves retrieval accuracy by using mixed-capacity retrievers. To improve the retrieval accuracy of the entire pipeline, we train SLM with retrieval-augmented fine-tuning to gain retrieval capacity. Then, we further distill the aggregated retrieval signals from SLM to DR so that DR could align with language models' preferences.

- **Coarse-to-Fine Granularity.** To complement mixed-capacity retrievers with each other, it is necessary to segment coarse-grained units into fine-grained ones, as the high-capacity retrievers (*e.g.,* DR) perform relatively poor than low-capacity ones (*e.g.,* SR) for coarse-grained units (Chen et al., 2024). As such, we first construct coarse-grained units with approximate 4K tokens by clustering multiple related documents before retrieval, which can considerably reduce the corpus size (*e.g.,* 21M→0.6M). Then, coarse-grained units are segmented into document-level units (*e.g.,* 4K→1K) before pre-ranking. Finally, we segment document-level units into passage-level units (*e.g.,* 1K→100) before post-ranking. These three groups of varied granularity units are sequentially fed into SR, DR, and SLM to locate the golden units with high accuracy and low cost.

These two novel designs jointly contribute to the considerable improvement of retrieval accuracy and efficiency in open-domain question answering (QA), such as Natural Question (NQ) (Kwiatkowski et al., 2019) and Trivia QA (TQA) (Joshi et al., 2017). Table 1 shows the key differences between the progressive retrieval and the flat one. FUNNELRAG features **(1)** Coarse-to-fine granularity which balances load and accuracy; **(2)** Later chunking which perceives the contextual information of retrieval units well[1]; **(3)** Compressed corpus (30x smaller from 21M to 600K) which reduces the burden on the retrieval stage; and **(4)** Long retrieval unit which improves answer recall to the full extent. More details can be found in §3.

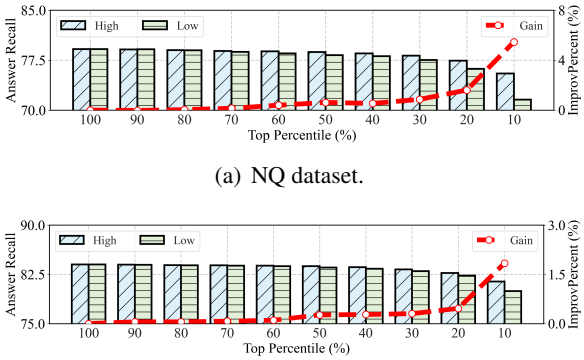
The main contributions of this work are summarized as three-folds: **(1)** This work highlights the issues commonly encountered in real-world RAG systems while overlooked in the existing studies, *i.e.,* the flat retrieval and constant granularity issues. **(2)** This work proposes FUNNELRAG, a coarse-to-fine progressive retrieval paradigm for RAG, satisfying the three properties of being time-saving, fine-grained, and contextual-integrity. **(3)** Extensive experiments demonstrate that FUNNELRAG can considerably reduce time overhead while the retrieval performance is comparable or even better in comparison with existing retrieval paradigms.

[1]Flat retrieval reranks passages that are not contextual integrity. However, our progressive retrieval perceives contextual information well in the post-ranking stage, since these passages are derived from documents with contextual integrity.



(a) NQ dataset. (b) TQA dataset.

Figure 2: AR *w.r.t.* coarse- and fine-grained retrieval. The bar denotes AR, while the line denotes the percentage of performance degradation compared to the cutoff position of 100%. The X-axis represents the percentage of units retrieved. Under the same percentile, the number of tokens retrieved by 'Fine' and 'Coarse' is equal.



(a) NQ dataset.

(b) TQA dataset.

Figure 3: Answer Recall *w.r.t.* high- and low-capacity retrievers. The line denotes the percentage of performance improvement compared to the low-capacity retriever.

2 Preliminaries

2.1 General Retrieval Paradigm for RAG

In retrieval-augmented generation, we are given a query q and a document set $\mathcal{D} = \{d_1, d_2, ..., d_{|\mathcal{D}|}\}$, where each document will be usually segmented into passage-level units, which results in a fine-grained corpus $\mathcal{P} = \{p_1, p_2, ..., p_{|\mathcal{P}|}\}$ consisting of millions of short retrieval units. The de facto retrieval paradigm comprises two stages: *retrieval* and *reranking*. The retrieval stage mostly employs a dense retriever, such as the Dense Passage Retriever (Karpukhin et al., 2020), which projects each passage $p \in \mathcal{P}$ to an embedding $\mathbf{E}(p)$ and projects the query q to an embedding $\mathbf{E}(q)$[2]. Then, the top-k relevant passages for query q are retrieved based on the query-passage embedding similarity, which is often computed by the dot product $\mathbf{E}(q)^{\mathrm{T}}\mathbf{E}(p)$. After that, the reranking stage employs a higher capacity reranker (such as RankGPT (Sun et al., 2023)) to filter out irrelevant passages. In the last, the top-n ranked passages serve as the context of the generator to facilitate its generation.

2.2 Experiment for Assumption Validation

As stated in Section 1, we emphasize the significance of (i) reducing the scale of candidates, (ii) refining the granularity of units, and (iii) increasing the level of retrievers' capacity, throughout the retrieval paradigm. Here, we conduct empirical studies to verify claims (ii) and (iii), as it is obvious that claim (i) will lead to better retrieval accuracy[3] (Sawarkar et al., 2024). We aim to evaluate whether these two claims can contribute to better retrieval performance, so as to validate that the progressive retrieval paradigm can balance effectiveness and efficiency well. To empirically verify these claims, we construct two synthetic datasets for NQ and TQA. Specifically, for each query in NQ and TQA, we retrieve the top-10 relevant clustered documents from the coarse-grained corpus[4] using BM25 (Robertson and Zaragoza, 2009), where each one has approximate 4K tokens. Then, we segment each clustered document into document-level units with about 1K tokens, striking the coarse-fine contrast. On the other hand, we adopt bge-reranker-v2-m3 (Chen et al., 2024) and BM25 to make a contrast between high- and low-capacity retrievers:

- **Coarse-Fine Contrast.** To simulate this scenario, we fix the retriever as bge-reranker-v2-m3 and use it to rerank coarse-grained (*i.e.,* clustered documents) and fine-grained units (*i.e.,* documents), respectively. We report the answer recall (AR) performance on the NQ and TQA[5], which are presented in Figures 2(a) and 2(b), respectively. From the results, the answer recall with fine-grained retrieval substantially outperforms that with coarse-grained one. Additionally, the performance degradation with fine-grained retrieval is significantly slower than that with coarse-grained one. For example, on the NQ dataset, fine-grained retrieval only drops 2.25% of its original performance, while coarse-grained

[2]Note that the embedding encoder $\mathbf{E}(\cdot)$ used for encoding queries and passages may not be the same, refer to (Karpukhin et al., 2020). For simplicity, we do make no distinction here.

[3]Assuming a fixed number of golden units that exists in candidates, increasing the scale of candidates reduces the signal-to-noise ratio, inevitable degrading retrieval accuracy.

[4]Referring to Section 3.1.1 and Algorithm 1 for more technical details about how the clustered documents formulated.

[5]Answer Recall (AR) measures the recall of the answer string in all the retrieved units. We treat it as retrieval metric.

retrieval drops 37.93%, when the cutoff position is top 20%. Given the above, it is necessary and valuable to segment coarse-grained units into fine-grained units along progressive retrieval stages in order for better retrieval performance.

- **High-Low Contrast.** To simulate this scenario, we fix the granularity of retrieval units as fine-grained ones and employ bge-reranker-v2-m3 and BM25 to rerank them, respectively. The experimental results are presented in Figure 3(a) and 3(b), respectively. From the results, we observe that the answer recall with the high-capacity retriever is consistently higher than that with the low-capacity one. In particular, the gain brought by the high-capacity retriever generally increases as the cutoff position decreases. For example, on the NQ dataset, the retrieval performance improvement of 'High' over 'Low' is 5.45% and 0.60% in terms of cutoff@10% and cutoff@20%. These observations indicate that using high-capacity retrievers in later stages can retain the golden retrieval units to a large extent.

3 Methodology

The overall framework of FUNNELRAG is shown in Figure 4. Then, we introduce three progressive retrieval stages from coarse to fine (§3.1) and then describe how to distill aggregated signals from the succeeding retriever to the preceding one (§3.2).

3.1 Coarse-to-Fine Progressive Retrieval

To better balance effectiveness and efficiency, we propose a progressive retrieval paradigm, which enables load balancing and improves retrieval accuracy by collaborating mixed-capacity retrievers, refining retrieval granularity, and reducing candidate scale. It can be formulated into three stages: (1) Retrieval, which usually uses relatively simple bi-encoder models to retrieve coarse-grained units; (2) Pre-ranking, which adopts cross-encoder models to rank the previously retrieved units; and (3) Post-ranking, which employs relatively complex list-wise models to rank these fine-grained units. We provide an intuitive example in Appendix C.5.

3.1.1 Retrieval Stage

This stage plays a role in improving answer recall (AR) to the full extent, without considering the granularity of the retrieval unit. Toward this end, we propose to convert the document set $\mathcal{D}$ into long retrieval (coarse-grained) units, which is proven to be effective in improving AR (Jiang et al., 2024). Specifically, we cluster documents in $\mathcal{D}$ based on their relationships, *i.e.,* hyperlinks embedded within each document, and finally results in a coarse-grained corpus $\mathcal{C} = \{c_1, c_2, ..., c_{|\mathcal{C}|}\}$.

The clustering algorithm is presented in Appendix D in Algorithm 1, where we draw inspiration from (Jiang et al., 2024) but make several major modifications. Specially, we sort documents by their local cluster coefficient from high to low in Line 1, so that documents with highly relevant will be clustered first. Furthermore, we measure the closeness centrality between document d and its related clusters $\mathcal{R}$ and sort them from high to low in Line 10, so that the most related cluster will be merged first. Each cluster c in $\mathcal{C}$ is a list of documents related to each other. We set the max cluster size S as 4K tokens because the experimental results (§4.4) demonstrate that overlength retrieval units do not bring much benefits to answer recall.

Currently, it is challenging and underperforming for dense retrievers to handle long context, even with the SOTA dense retrievers, *e.g,* BGE-M3 (Chen et al., 2024). In LongRAG (Jiang et al., 2024), they handle long context by maximizing the scores of all chunks within each long retrieval unit, where they segment each long unit into 512-token chunks. However, this practice has no essential difference with direct retrieval of short units and is computation-consuming. As the goal of this stage is to improve AR regardless of granularity, we use sparse retrievers to handle long context, which shows more effective than dense retrievers in long-doc retrieval (Chen et al., 2024). Without loss of generality, we adopt the representative sparse retriever in this work, *i.e.,* BM25 (Robertson and Zaragoza, 2009), whose computational cost is far below dense retrievers. Lastly, we use BM25 to retrieve the top-K relevant long units $\mathcal{C}_l = \{c_1^l, c_2^l, ..., c_K^l\}$. We set K as a proper value (*e.g.,* 80) to achieve relatively high AR and reduce the computational burden of the following stages.

3.1.2 Pre-ranking Stage

This stage plays a role in finding the "needle" (*i.e.,* the document containing the answer string) from the "needle case" (*i.e.,* the retrieved clusters) rather than the "haystack" (millions in most cases). The corpus size is the key difference between the existing retrieval paradigm and ours, where the corpus size of ours is hundreds, making it easy to find the "needle". Formally, we first segment each cluster

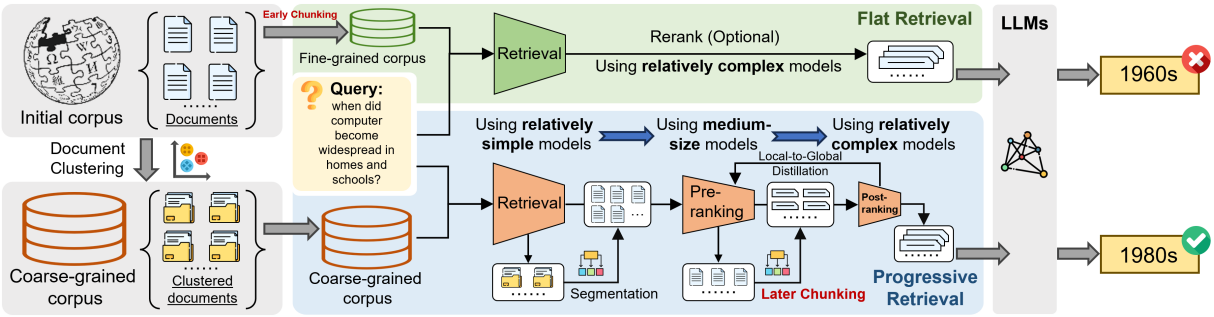


Figure 4: The overall system framework of FUNNELRAG. The upper layer illustrates the working flow of the **flat retrieval paradigm**, while the bottom layer illustrates the working flow of our **progressive retrieval paradigm**.

c^l in $\mathcal{C}_l$ into document-level units d. And then, we utilize a cross-encoder model, denoted as $f(\cdot)$, to measure their pre-ranking scores to the query q:

$$S^{pre} = f(q, d). \quad (1)$$

Given pre-ranking scores, we retrieve top-N documents closest to the query $\mathcal{D}_m = \{d_1^m, d_2^m, ..., d_N^m\}$. Figure 2 shows the performance comparison between fine-grained and coarse-grained units. The results indicate that the finer retrieval units can significantly reduce the loss of retrieval performance.

3.1.3 Post-ranking Stage

This stage plays a role in identifying fine-grained units that align with language models' preferences. We segment each document into passage-level units p in this relatively late stage. Referring to (JinaAI, 2024), we term this technique as "**Later Chunking**", which retrieves long units in the previous stages to preserve contextual semantics well and then segments long units into fine-grained units in the later stages for better retrieval applications. After chunking, we obtain M passages $\{p_1^f, p_2^f, ..., p_M^f\}$. Then, a practical question naturally arises: *How do efficiently post-rank these fine-grained units to align language models' preferences?* A straightforward way is to serve LLMs as agents to post-rank passages with their preferences, *e.g.,* RankGPT (Sun et al., 2023) and RankVicuna (Pradeep et al., 2023). Nevertheless, directly using LLMs as rankers inevitably inflicts a significant computational burden on the post-ranking stage, making it very difficult to deploy in production.

Instead of heuristically instructing LLMs to post-rank, our post-ranker builds upon FiD (Fusion-in-Decoder) (Izacard and Grave, 2021b), a retrieval-augmented encoder-decoder language model. Specifically, each passage p is paired with the query q to be processed separately by the T5 (Raffel et al., 2020) encoder. Subsequently, the encoded representations are concatenated along the sequence dimension. Lastly, the T5 (Raffel et al., 2020) decoder attends all of parts simultaneously:

$$\text{FiD}(q, \mathcal{P}_f) = \text{Dec}\big(\mathbf{H}_{q,p_1^f} \| , ..., \| \mathbf{H}_{q,p_M^f}\big), \quad (2)$$

where $\mathbf{H}_{q,p^f} = \text{Enc}(q \oplus p^f)$; $\oplus$ and $\|$ are the concatenation operator. By independently processing each (q, p^f) pair, the encoder computes self-attention within one passage at a time. As such, the computational cost scales linearly with the number of passages, making the post-ranking stage highly efficient. Furthermore, the decoder's cross-attention scores have shown to be remarkably effective in estimating the relative importance of the retrieval-augmented passage from the language models' preferences (Izacard and Grave, 2021a; Yu et al., 2023; Izacard et al., 2023; Shi et al., 2024). Given that, we average FiD cross-attention scores corresponding to each (q, p^f) pair from the first decoder token[6] over all the layers, all the heads, as well as all the representative tokens within $q \oplus p^f$:

$$S^{post} = \frac{1}{ln \times lh \times lr} \sum_{i=0}^{ln} \sum_{j=0}^{lh} \sum_{k=0}^{lr} \alpha_{0,i,j,\mathbf{r}[k]}, \quad (3)$$

where ln, lh, and lr denote the number of the layers, heads, and representative tokens, respectively; $\alpha_{0,i,j,\mathbf{r}[k]}$ represents the score of the first decoder token in terms of the i-th layer, j-th head, and k-th representative token; $\mathbf{r}[k]$ lookups the index of k-th representative token. We select lr tokens with the highest scores to serve as representative ones[7]. The cross-attention mechanism is provided

[6]Here, we take the score of the first token, as it leads to satisfactory performance in general. More analysis about attention aggregation schema can be found in Appendix C.1.

[7]Note that we do not consider query tokens in q when selecting representative tokens, refer to the ablation study in Appendix C.2 for more details. More technical details about representative token selection can be found in Appendix A.2.

in Appendix A.1. Before estimating the relative importance score, we train FiD to predict the answer with retrieval-augmented passages, so that FiD can learn to look for cues. After that, we obtain the post-ranking score and treat these passages with scores ranked in the top-H as oracle passages $\mathcal{P}_f = \{p_1^f, p_2^f, ..., p_H^f\}$. The oracle passages $\mathcal{P}_f$ will be fed into generators to facilitate generation.

3.2 Distillation with Aggregated Signals

Ideally, we would like the retrieval units containing oracle passages to be ranked at the forefront by the preceding retriever, so that the succeeding one can reach out to oracle passages. To this end, we propose to distill retrieval knowledge from the succeeding to the preceding, termed local-to-global (L2G) distillation[8]. While distilling, an issue emerges: the granularity of retrieval units between sequentially neighboring stages is different, hindering distillation. Given that, we propose aggregating retrieval scores from local to global. By doing so, we can fill the granularity gap between two neighboring stages. Formally, we aggregate retrieval scores as:

$$S_d^{post \to pre} = \alpha \times \max_{p \in d} \left\{ S_p^{post} \right\} + (1-\alpha) \times \operatorname*{mean}_{p \in d} \left\{ S_p^{post} \right\}, \quad (4)$$

where $\alpha \in [0, 1]$ controls the strength of '**max**' and '**mean**' parts. When $\alpha = 1$, the score focuses on the most important passage in the document; when $\alpha = 0$, the score focuses on the average importance level of all passages in the document. We annotate documents with Top-$\mathcal{K}$ aggregated scores as positive. Inspired by (Yu et al., 2023), we annotate the ones $\mathcal{D}^{h+}$ that **h**it the ground truth as positive to make the training process more robust:

$$\mathcal{D}^+ = \mathcal{D}^{h+} \cup \underset{S_d^{post \to pre}}{\text{Top-}\mathcal{K}} \ \mathcal{D}_m, \qquad \mathcal{D}^- = \mathcal{D}_m \backslash \mathcal{D}^+, \quad (5)$$

where the remaining documents (*i.e.,* $\mathcal{D}_m \backslash \mathcal{D}^+$) serves as negative ones. Following ANCE (Xiong et al., 2021), we adopt the pairwise Bayesian Personalized Ranking (BPR) loss (Rendle et al., 2009), enforcing the match score between the query and positive documents to be higher than negative ones:

$$\mathcal{L} = \sum_q \sum_{d^+ \in \mathcal{D}^+} \sum_{d^- \in \mathcal{D}^-} -\log \sigma(S_{d^+}^{pre} - S_{d^-}^{pre}), \quad (6)$$

where $\sigma(\cdot)$ denotes the sigmoid function; $S_{d^+}^{pre}$ and $S_{d^-}^{pre}$ denote $f(q, d^+)$ and $f(q, d^-)$, respectively.

4 Experiment

In this section, we conduct extensive experiments on two QA benchmark datasets to answer the following Research Questions (**RQs**): **RQ1:** How does progressive retrieval perform *w.r.t.* Answer Recall when compared to flat one? **RQ2:** What are the benefits of performing FUNNELRAG in Question Answering? **RQ3:** How do different settings influence the effectiveness of progressive retrieval?

4.1 Experimental Settings

Datasets. We experiment on two QA datasets: *i.e.,* Natural Question (NQ) (Kwiatkowski et al., 2019) and Trivia QA (TQA) (Joshi et al., 2017). Appendix B.1 provides the statistics of the datasets.

Metrics. Following (Lewis et al., 2020) and (Jiang et al., 2024), we employ Answer Recall (AR) and Exact Match (EM) to measure the performance of retrieval and generation, respectively. Besides, we use time cost as the metric of retrieval efficiency.

Retrievers. In progressive retrieval, we adopt BM25 (Robertson and Zaragoza, 2009) for retrieval, employ bge-reranker-v2-m3 (Chen et al., 2024) as our pre-ranker, and leverage FiD (Izacard and Grave, 2021b) to perform post-ranking. As for flat retrieval, we use bge-large-en-v1.5 (Xiao et al., 2023), the SOTA embedding model, to retrieve passages, and use bge-reranker-v2-m3 to rerank.

Generators. We choose two representative open-source LLMs: Llama3-8B-Instruct (Dubey et al., 2024) and Qwen2-7B-Instruct (Yang et al., 2024), for our QA evaluation[9]. More details related to experimental settings can be found in Appendix B.

4.2 Retrieval Performance (RQ1)

Table 2, 10, 11, and 12 shows the retrieval results on NQ and TQA datasets. As BM25's retrieval speed is breakneck, we mark its time cost as 0.00. From the results, we mainly have the following observations: **(1)** By adjusting the collaboration between large-to-small quantities, coarse-to-fine granularity, and simple-to-complex retrievers, progressive retrieval can find a gain-cost balance point. Specifically, the time cost of progressive retrieval is considerably lower than that of

[8] In this work, we only distill retrieval knowledge from the post-ranking to pre-ranking stage, as we use sparse retrieval (BM25 (Robertson and Zaragoza, 2009)) in the retrieval stage.

[9] We use Llama3-8B and Qwen2-7B to represent Llama3-8B-Instruct and Qwen2-7B-Instruct, respectively, for brevity.

Datasets	Retrieval Paradigm	Retrieval Stages			Time Cost (T)	Answer Recall (AR)
		Retrieval	Pre/Re-ranking	Post-ranking		
NQ	Flat Retrieval	21M → 4p	N/A	N/A	4.90 (4.90+N/A+N/A)	72.91
		21M → 400p	400p → 4p	N/A	5.25 (4.90+0.35+N/A)	75.90
	Progressive Retrieval	600K → 4c	N/A	N/A	0.00 (0.00+N/A+N/A)	71.69
		600K → 20c	20c $\xrightarrow{\sim 180d}$ 4d	N/A	0.49 (0.00+0.49+N/A)	74.57
		600K → 80c	80c $\xrightarrow{\sim 740d}$ 8d	8d $\xrightarrow{\sim 50p}$ 4p	2.97 (0.00+2.20+0.77)	75.43
TQA	Flat Retrieval	21M → 4p	N/A	N/A	5.02 (5.02+N/A+N/A)	75.22
		21M → 400p	400p → 4p	N/A	5.41 (5.02+0.39+N/A)	81.29
	Progressive Retrieval	600K → 4c	N/A	N/A	0.00 (0.00+N/A+N/A)	78.94
		600K → 20c	20c $\xrightarrow{\sim 200d}$ 4d	N/A	0.60 (0.00+0.60+N/A)	80.00
		600K → 80c	80c $\xrightarrow{\sim 800d}$ 12d	12d $\xrightarrow{\sim 65p}$ 4p	3.47 (0.00+2.52+0.95)	80.69

Table 2: Retrieval performance comparison *w.r.t.* time cost and answer recall on NQ and TQA datasets, where 'c', 'd', and 'p' denote clustered documents, documents, and passages, respectively. The value on the arrow (→) indicates the number of fine-grained candidates after segmenting coarse-grained ones. The number of time cost is in seconds. We provide detailed hardware and software configurations for experiments on time cost in Appendix B.4.

Generators	@K	NQ		TQA	
		Flat	Progressive	Flat	Progressive
Llama3-8B	@1	43.88	**51.27** (16.84%)	67.54	**70.77** (4.78%)
	@2	46.18	**51.50** (11.52%)	66.84	**67.66** (1.23%)
	@3	**49.31**	49.06 (-0.51%)	66.10	**67.89** (2.71%)
	@4	**51.44**	48.86 (-5.02%)	67.10	**68.47** (2.04%)
Qwen2-7B	@1	48.86	**53.43** (9.35%)	72.01	**72.39** (0.53%)
	@2	53.32	**54.27** (1.78%)	**73.69**	73.17 (-0.71%)
	@3	54.63	**55.07** (0.81%)	**74.60**	73.73 (-1.17%)
	@4	55.46	**55.48** (0.04%)	**74.97**	74.37 (-0.80%)
Average Performance		50.39	**52.37** (3.93%)	70.36	**71.06** (0.99%)

Table 3: Generation performance in terms of different retrieval paradigms. The **bold** indicates the best results.

the flat one (reduced by about 40%), while the AR is comparable to (or even better than) the flat one. The main reason for this results is that progressive retrieval combines the advantage of different retrievers but circumvents their disadvantage[10]. **(2)** By retrieving in a funnel manner (*e.g.,* 600K $\xrightarrow{\text{Retrieval}}$ 80c $\xrightarrow{\text{Pre-ranking}}$ 8d $\xrightarrow{\text{Post-ranking}}$ 4p), progressive retrieval enables load balancing. It assigns simple but computationally demanding tasks to low-capacity retrievers, *e.g.,* retrieving 80 clustered documents from 600K candidates. With almost no loss of golden units, the number of candidates dropped 7500x, largely increasing the signal-to-noise ratio. On the other hand, it assigns hard but computationally undemanding tasks to high-capacity retrievers, *e.g.,* retrieving 4p clustered documents from 8d candidates. More retrieval performance comparisons can be seen in appendix C.4.

[10]Progressive retrieval combines simple retrievers' fast speed with complex ones' high precision, while it avoids simple retrievers' low precision and complex ones' slow speed.

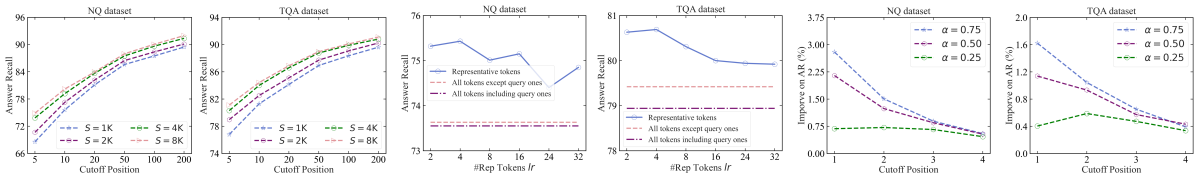
4.3 Generation Performance (RQ2)

Table 3 shows the generation performance *w.r.t.* flat and progressive retrieval. From the results, we observe performance with progressive retrieval outperforms that with flat one in 13 out of 18 cases, indicating the superiority of supplementing LLMs with FUNNELRAG. Specifically, when the cutoff position is small, progressive retrieval considerably outperforms flat one, such as the improvement of 16.87% and 9.35% on NQ dataset in terms of @1. When the cutoff position is high, progressive retrieval still outperforms flat one in half of cases, even if the AR of progressive one is slightly lower than flat one. The reason is that passages retrieved by FUNNELRAG have higher contextual integrity, refer to Appendix C.3 for more details. Besides, we find Llama3-8B's performance may deteriorate when provided with more passages, whereas Qwen2-7B does not. The main reason is the difference in the ability of them to handle long contexts.

4.4 Study of FUNNELRAG (RQ3)

In this section, we move on to studying different settings in the proposed FUNNELRAG framework:

(1) AR *w.r.t.* different granularity: As shown in Figure 5(a), we experiment to evaluate the answer recall *w.r.t.* different granularity of clustered documents, where we tune the max cluster size S within the range of {1K, 2K, 4K, 8K}. It can be observed that setting S as 8K does not bring much performance benefits compared to 4K. In contrast, setting S as a small value (*e.g.,* 1K) considerably degrades the performance. In view of this, we set



(a) AR *w.r.t.* different granularity. (b) Impact of #rep tokens (in Top-4). (c) Effect of local-to-global distillation.

Figure 5: Model performance *w.r.t.* (a) different granularity of clustered documents, (b) different number of representative tokens, and (c) L2G distillation. #Rep tokens is the abbr of "the number of representative tokens".

the max cluster size S as 4K tokens in the work.

(2) Impact of representative tokens: As shown in Figure 5(b) and 6, we experiment to investigate #rep tokens lr, where we search lr in the range of {2, 4, 8, 16, 24, 32}. We also aggregate FiD cross-attention scores across all tokens with/without query ones. We observe that the performance of aggregating rep tokens significantly outperforms that of aggregating all ones. On the other hand, it gets a peak value when selecting a few rep tokens, such as 4 in most cases. The results fully validate the necessity of selecting rep tokens. More related details can be seen in Appendix C.2.

(3) Effect of local-to-global distillation: As shown in Figure 5(c), we experiment to explore the effect of local-to-global distillation, we tune α in the range of {0.25, 0.5, 0.75}. From the results, we observe that a higher value α (*e.g.,* 0.75) usually leads to better performance and relatively high improvement on the top-ranked position (*e.g.,* Top-1). The improvement is relatively trivial when α is small (*e.g.,* 0.25). When the cutoff position is large (*e.g.,* Top-4), different values of α bring similar performance improvements. In conclusion, we suggest tuning α within the range of [0.5, 1.0).

5 Related Works

Information Retrieval (IR). IR models contain two categories: (1) Sparse retrieval, which computes relevance scores via lexical similarity (Salton and Buckley, 1988); (2) Dense retrieval, which captures semantic similarity within dense space (Izacard et al., 2022). To further improve retrieval performance, the re-ranking module is proposed to re-estimate relevance scores using an extra model (Zhuang et al., 2023; Nogueira et al., 2020; Hwang et al., 2024). Recently, LLMs have shown remarkable capability in re-ranking without further fine-tuning (Liang et al., 2023; Sun et al., 2023; Pradeep et al., 2023). Despite effectiveness, they have not fully disclosed coarse-to-fine granularity evolution.

Retrieval-Augmented Generation. Retrieving knowledge from external sources to supplement LLMs' generation has been proven effective (Lewis et al., 2020; Guu et al., 2020; Mialon et al., 2023; Izacard et al., 2023) in knowledge-intensive tasks (Petroni et al., 2021). Specifically, RAG incorporates retrieval modules to provide external non-parametric knowledge into generation modules to remedy their incomplete, incorrect, or outdated internal parametric knowledge (Karpukhin et al., 2020; Min et al., 2019). Previous works jointly train the retrieval and generation modules (Borgeaud et al., 2022; Lewis et al., 2020; Guu et al., 2020). With the rise of LLMs (OpenAI, 2023), most works directly serve them as generation modules without tuning due to their strong emergent abilities (Xu et al., 2024; Jeong et al., 2024; Cheng et al., 2024). Inspired by RAG's strong practicality, a wide range of domains have developed specific RAG frameworks to perform their tasks, including computer vision (Xie et al., 2023; Sharifymoghaddam et al., 2024), knowledge graph (Yu et al., 2022; Edge et al., 2024), and speech (Xue et al., 2024; Wang et al., 2024). Albeit studied for ages, the balance between the effectiveness and efficiency of retrievers is less explored.

6 Conclusion

In this paper, we propose a coarse-to-fine progressive retrieval paradigm for RAG, FUNNELRAG, to enable load balancing and improve retrieval performance. The FUNNELRAG framework efficiently retrieves useful passages by seamlessly collaborating different granularity (clustered document → passage), quantity (21M → 8d), and capacity (BM25 → FiD). Applying our framework reduces time cost by nearly 40%, while the retrieval accuracy is comparable. In conclusion, our work sheds light on collaboration between complex and simple retrieval modules, and extensive experiments fully demonstrate its feasibility as well as usefulness.

Limitations

Despite our innovations and improvements, we must acknowledge certain limitations in our work:

- **Heuristic Metric.** The answer recall (AR), which is the main retrieval metric adopted in the experiments, might overestimate the usefulness of the retrieved information because it mechanically measures whether the retrieved information contains the answer string, even if the retrieved information does not convey accurate semantics. Evaluating metrics for RAG remains an open field that still needs further investigation.

- **Hand-crafted Labor.** Although our method is able to balance effectiveness and efficiency well, some hand-crafted labors are required to fulfill this goal. Specifically, to enable load balancing and improve retrieval accuracy, some hyperparameters needed to be tuned, such as the max cluster size S, and the data flow between each stage required to carefully collaborate, such as the quantity of the retrieved units in each stage.

- **Compatibility Issue.** It is worth mentioning that FUNNELRAG is a model-agnostic framework and can flexibly adapt to different retrievers. Therefore, the performance of FUNNELRAG may be influenced by the characteristics of the retrievers. On the other hand, users can customize the retrievers to accommodate their specific scenarios. This work experiment on recently released retrievers and the results fully demonstrate the superiority of FUNNELRAG.

References

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, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *CoRR*, abs/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. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *CoRR*, abs/2402.03216.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. 2024. xrag: Extreme context compression for retrieval-augmented generation with one token. *CoRR*, abs/2405.13792.

Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 191–198. ACM.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Boyi Deng, Wenjie Wang, Fengbin Zhu, Qifan Wang, and Fuli Feng. 2024. Cram: Credibility-aware attention modification in llms for combating misinformation in RAG. *CoRR*, abs/2406.11497.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan

Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Tevatron: An efficient and flexible toolkit for dense retrieval. *CoRR*, abs/2203.05765.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

Mihajlo Grbovic and Haibin Cheng. 2018. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 311–320. ACM.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Taeho Hwang, Soyeong Jeong, Sukmin Cho, Seungyoon Han, and Jong C. Park. 2024. DSLR: document refinement with sentence-level re-ranking and reconstruction to enhance retrieval-augmented generation. *CoRR*, abs/2407.03627.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Gautier Izacard and Edouard Grave. 2021a. Distilling knowledge from reader to retriever for question answering. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Gautier Izacard and Edouard Grave. 2021b. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. Association for Computational Linguistics.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 7036–7050. Association for Computational Linguistics.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. 2024. Longrag: Enhancing retrieval-augmented generation with long-context llms. *CoRR*, abs/2406.15319.

JinaAI. 2024. Late chunking in long-context embedding models.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Greg Kamradt. 2023. Needle in a haystack - pressure testing llms.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan,

Sébastien M. R. Arnold, Vincent Perot, Siddharth Dalmia, Hexiang Hu, Xudong Lin, Panupong Pasupat, Aida Amini, Jeremy R. Cole, Sebastian Riedel, Iftekhar Naim, Ming-Wei Chang, and Kelvin Guu. 2024. Can long-context language models subsume retrieval, rag, sql, and more? *CoRR*, abs/2406.13121.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6086–6096. Association for Computational Linguistics.

Patrick S. H. 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2023. Holistic evaluation of language models. *Trans. Mach. Learn. Res.*, 2023.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. RA-DIT: retrieval-augmented dual instruction tuning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *Preprint*, arXiv:2407.03618.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey. *Trans. Mach. Learn. Res.*, 2023.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Knowledge guided text retrieval and reading for open domain question answering. *CoRR*, abs/1911.03868.

Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 708–718. Association for Computational Linguistics.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2523–2544. Association for Computational Linguistics.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *CoRR*, abs/2309.15088.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.*, 24(5):513–523.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended RAG: improving RAG (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. *CoRR*, abs/2404.07220.

Claude E. Shannon. 2001. A mathematical theory of communication. *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55.

Sahel Sharifymoghaddam, Shivani Upadhyay, Wenhu Chen, and Jimmy Lin. 2024. Unirag: Universal retrieval augmentation for multi-modal large language models. *CoRR*, abs/2405.10311.

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. *CoRR*, abs/2301.12652.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 8371–8384. Association for Computational Linguistics.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Mingqiu Wang, Izhak Shafran, Hagen Soltau, Wei Han, Yuan Cao, Dian Yu, and Laurent El Shafey. 2024. Retrieval augmented end-to-end spoken dialog models. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*, pages 12056–12060. IEEE.

Zhe Wang, Liqin Zhao, Biye Jiang, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2020. COLD: towards the next generation of pre-ranking system. *CoRR*, abs/2007.16122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *CoRR*, abs/2309.07597.

Chen-Wei Xie, Siyang Sun, Xiong Xiong, Yun Zheng, Deli Zhao, and Jingren Zhou. 2023. RA-CLIP: retrieval augmented contrastive language-image pre-training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 19265–19274. IEEE.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Zhipeng Xu, Zhenghao Liu, Yibin Liu, Chenyan Xiong, Yukun Yan, Shuo Wang, Shi Yu, Zhiyuan Liu, and Ge Yu. 2024. Activerag: Revealing the treasures of knowledge via active learning. *CoRR*, abs/2402.13547.

Jinlong Xue, Yayue Deng, Yingming Gao, and Ya Li. 2024. Retrieval augmented generation in prompt-based text-to-speech synthesis with context-aware contrastive language-audio pretraining. *CoRR*, abs/2406.03714.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang,

Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2022. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 4961–4974. Association for Computational Linguistics.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2421–2436. Association for Computational Linguistics.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. Rankt5: Fine-tuning T5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2308–2313. ACM.

A More Details Related to Methods

A.1 Cross-Attention

Our model is built upon FiD (Izacard and Grave, 2021b), whose architecture is a sequence-to-sequence model that consists of an encoder and a decoder. The encoder encodes each query-passage pair separately. The output representations of the encoder are concatenated (Equ (2)) along the sequence dimension to form a global representation $\mathbf{H} = \|_{m=1}^{M} \mathbf{H}_{q,p_m^f}$. Then, the decoder autoregressively attends this representation, which alternates self-attention, cross-attention, and feed-forward modules. In the decoder, only the cross-attention module explicitly takes the global representation $\mathbf{H}$ of the encoder as the input. Let $\mathbf{X}$ denote the output of the previous self-attention layer of the decoder, the cross-attention operation consists of the following steps. First, three linear transformations are applied to produce queries $\mathbf{Q}$, keys $\mathbf{K}$, and values $\mathbf{V}$, which can be formulated as follows:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \mathbf{K} = \mathbf{H}\mathbf{W}^K, \mathbf{V} = \mathbf{H}\mathbf{W}^V. \quad (7)$$

Then, the cross-attention scores are computed using the dot product between queries $\mathbf{Q}$ and keys $\mathbf{K}$. Taking the query at position i, $\mathbf{Q}_i$, and the key at position j, $\mathbf{K}_j$ for example, the cross-attention score and its normalized one can be computed as:

$$\alpha_{i,j} = \mathbf{Q}_i \mathbf{K}_j, \quad \tilde{\alpha}_{i,j} = \frac{\exp(\alpha_{i,j})}{\sum_k \exp(\alpha_{i,k})}. \quad (8)$$

A new representation is derived from a sum of the values, weighted by the normalized cross-attention scores, going through a final linear transformation:

$$\mathbf{O}_i = \mathbf{W}_O \sum_j \tilde{\alpha}_{i,j} \mathbf{V}_j. \quad (9)$$

We omit the layer and head indices for brevity in the above formula, refer to (Vaswani et al., 2017) for more details on the structure of Transformers.

A.2 Representative Token Selection

Enlightened by the token redundancy in hidden states (Dai et al., 2020; Goyal et al., 2020), we select several representative tokens per layer and head, to estimate the relative importance of each passage more accurately. Specifically, we select lr tokens that have the highest cross-attention scores from the passage to represent it. For the i-th layer and j-th head, we select the representative tokens according to their cross-attention scores and obtain an index list $\mathbf{r}$ that contains the index of lr representative tokens. Then, we can average FiD cross-attention scores over representative tokens by looking up the index list $\mathbf{r}$, referring to Equ (3).

Dataset	#Train	#Dev	#Test
NQ (Kwiatkowski et al., 2019)	79.1k	8.7k	3.6k
TQA (Joshi et al., 2017)	78.7k	8.8k	11.3k

Table 4: Statistics of the datasets.

B More Details Related to Experiments

B.1 Datasets

We experiment on NQ (Kwiatkowski et al., 2019) and TQA (Joshi et al., 2017) datasets. Following (Lee et al., 2019), we discard answers with more than 5 tokens, as answers with many tokens often resemble extractive snippets instead of canonical ones. Table 4 shows the statistics of the datasets.

B.2 LLMs for QA

- **Llama3-8B-Instruct**[11] is the developed version of Llama2 (Touvron et al., 2023). Compared to Llama2, its training data volume has increased by seven times, showing higher reasoning and multilingual capabilities. For question answering, we use the Llama3-8B-Instruct, which is specifically optimized for the instruction-following ability.

- **Qwen2-7B-Instruct**[12] is the developed version of Qwen (Bai et al., 2023). Except for Chinese and English, its training data has also added high-quality data related to 27 languages and it achieves significant improvement in code and math capabilities. We also use the instruction-following version for the same reason aforesaid.

B.3 Prompts

We provide the prompt that is used for retrieval-augmented question answering (§4.3) in Table 5.

B.4 Hardware and Software Configurations

Hardware Configurations. The experiments are conducted on a Linux server equipped with an AMD EPYC 7742 64-Core Processor, 1000GB RAM, and 2 NVIDIA A100-SXM4-40GB GPUs.

[11] https://github.com/meta-llama/llama3
[12] https://github.com/QwenLM/Qwen2

Retrieval-Augmented Question Answering Prompt

[Instruction]
Given the [‘context’, ‘question’], answer the question based on the context.
[Completion]
[context]: Nobel Prize in Physics The Nobel Prize in Physics () is a yearly award given by the Royal Swedish Academy of Sciences .....
[question]: Who got the first nobel prize in physics?
[answer]:

Table 5: The prompt for retrieval-augmented QA. For intuition, we select a real example from the NQ dataset.

Software Configurations. We run retrieval and generation experiments on the above hardware. We conduct experiments with the FlagEmbedding[13] and huggingface transformers toolkit (Wolf et al., 2020). For dense retrieval, we leverage the open-source retrieval toolkit Tevatron (Gao et al., 2022). On the other hand, we adopt BM25S (Lù, 2024) for sparse retrieval. Besides, we adopt T5-large[14] as the backbone of FiD (§3.1.3). The detailed setting of the software environment is provided in Table 6.

Configuration	Value
Tevatron	V2
bm25s	0.1.10
faiss-cpu	1.8.0.post1
FlagEmbedding	1.2.11
transformers	4.44.2

Table 6: Detailed settings of the software environment.

C More In-depth Studies

C.1 Study of Attention Aggregation Schema

Table 7 shows the results with different aggregation schemes. Particularly, we consider **(1)** taking the average over all the layers, all the heads, and all the representative tokens; **(2)** taking the max over the layers instead of the average; **(3)** taking the max over the heads instead of the average; **(4)** taking the max over input tokens instead of the average; **(5)** taking the mean over the last 6 layers instead of all the layers. In all of the above variants, we do not consider query tokens when aggregating. We observe that: (1) Aggregation schemes with representative tokens selection (*i.e.,* (i), (iv), and (v)) outperform those without representative tokens selection by a large margin, which indicates the necessity of selecting representative tokens. (2) Comparing (i) and (iv), we find that setting #rep tokens to a small value (*e.g.,* 1) will hurt the retrieval performance, which suggests setting #rep tokens to moderate values (*e.g.,* 4). (3) Comparing (i) and (v), only considering part of the decoder layers may bring a little performance degradation. The main reason is that certain clues captured by the other layers have been neglected. (4) Comparing (ii) and (iii), taking the max over the heads generally performs better than that over the layers. The main reason is taking the max over the heads may find the influential head that plays a key role in identifying clues (Deng et al., 2024), while taking the max over the layers will miss some key clues captured by the other layers. In conclusion, we suggest setting #rep tokens to moderate values, using all the layers, and trying to identify the influential heads.

C.2 Ablation of Query Tokens

In Table 8, we conduct experiments to verify the effectiveness of ablating query tokens when estimating the relative importance of each passage. Averaging the FiD cross-attention score of all tokens in $q \oplus p^f$ mentions “**w/ query**” and averaging that of all tokens in p^f mentions “**w/o query**”. Here, we do not consider selecting representative tokens so that we can verify the effectiveness of ablating query tokens straightforwardly. The results show that ablating query tokens consistently improves answer recall across different cutoff positions. Specifically, We find that the marginal benefit appears as the cutoff position increases. For example, on the TQA dataset, **w/o query** achieves 5.17% improvement over **w/ query** at the Top-1 position, while it is 0.61% at the Top-4 position. It can be concluded that ablating query tokens can significantly

[13] https://github.com/FlagOpen/FlagEmbedding
[14] https://huggingface.co/google-t5/t5-large

Method	NQ					TQA				
	@1	@2	@3	@4	Avg	@1	@2	@3	@4	Avg
(i) $\text{mean}_{i,j,k}\alpha_{0,i,j,\mathbf{r}[k]}$	**63.74**	**71.36**	**73.88**	75.43	**71.10**	74.57	**78.31**	79.83	80.69	78.35
(ii) $\text{mean}_{j,k}\text{max}_i\alpha_{0,i,j,k}$	57.06	68.01	71.91	73.85	67.71	66.98	75.44	78.02	79.56	75.00
(iii) $\text{mean}_{i,k}\text{max}_j\alpha_{0,i,j,k}$	57.65	68.14	71.86	73.77	67.86	68.32	75.70	78.12	79.47	75.40
(iv) $\text{mean}_{i,j}\text{max}_k\alpha_{0,i,j,k}$	63.32	70.69	73.52	75.12	70.66	74.10	78.11	79.70	80.59	78.13
(v) $\text{mean}_{-6\leq i\leq -1,j,k}\alpha_{0,i,j,\mathbf{r}[k]}$	63.68	71.16	73.82	**75.51**	71.04	**74.64**	78.26	**79.93**	**80.89**	**78.43**

Table 7: Performance comparison of attention aggregation schemes on NQ and TQA datasets, where the index i corresponds to layers of the decoder, j corresponds to heads, and k corresponds to input tokens of a given passage.

Datasets	@K	w/ query	w/o query	%Improv.
NQ	@1	56.79	**57.65**	1.51%
	@2	67.51	**67.84**	0.49%
	@3	71.80	**71.88**	0.11%
	@4	73.55	**73.63**	0.11%
TQA	@1	65.33	**68.71**	5.17%
	@2	74.67	**75.86**	1.59%
	@3	77.49	**78.27**	1.01%
	@4	78.94	**79.42**	0.61%

Table 8: AR comparison with and without query tokens.

Dataset	Flat Retrieval	Progressive Retrieval
NQ	1.330	**1.259**
TQA	1.760	**1.494**

Table 9: Contextual integrity measurement on passages. The lower the value, the better the contextual integrity.

improve answer recall in the top-ranked positions, however, the gain decreases as the number of passages retrieved increases. The main reason is that as the number of passages retrieved increases, it becomes more essential than ablating query tokens.

C.3 Contextual Integrity Analysis

Generally, a higher contextual integrity is in favor of most downstream tasks. To measure contextual integrity, we compute information entropy (Shannon, 2001) on passages' source in terms of Top-4 passages for each query, where we use the title of passages as the identifier. Table 9 shows the measurement results *w.r.t.* flat and progressive retrieval. From the results, we find that progressive retrieval achieves lower information entropy (*i.e.,* higher contextual integrity) than flat one. Specifically, the improvement of progressive retrieval over flat one is 5.34% and 15.11% in terms of NQ and TQA datasets, respectively. The results fully verify that the proposed FUNNELRAG can largely enhance the contextual integrity just as we discussed in §1.

C.4 Retrieval Performance Comparison

Table 10-12 shows the retrieval performance on NQ and TQA datasets in terms of the cutoff position of 1, 2, and 3. The results show similar trends as Table 2. Specifically, progressive retrieval usually achieves better performance while using less time cost. For example, on the NQ dataset, progressive retrieval uses the time cost of 2.97s and obtains the **AR** of 63.73, in terms of Top-1, while flat one takes 5.25s and only achieves the **AR** of 56.09. The results verify the superiority of progressive retrieval in balancing effectiveness and efficiency.

C.5 Case Study

Table 13 shows a data-flow example of FUNNELRAG from the NQ dataset. It consists of three stages: Retrieval, Pre-ranking, and Post-ranking. From the retrieval to the post-ranking stage, the granularity evolves from coarse-grained clustered documents to fine-grained passages. Besides, the number of candidates decreases step by step (*e.g.,* $600\text{K} \xrightarrow{\text{Retrieval}} 80\text{c} \xrightarrow{\text{Pre-ranking}} 8\text{d} \xrightarrow{\text{Post-ranking}} 4\text{p}$). More importantly, we find that the proportion of candidates that contains the answer string gradually increases, which indicates that the signal-to-noise ratio is greatly improved through progressive retrieval. Furthermore, with more advanced models, the pre-ranking and the post-ranking stage can perceive clues in the question, *i.e.,* "**the fifth season**". Though the model becomes more complex, the number of candidates is limited, so the computational cost is small. As a result, "**America's Got Talent (season 5)**" has been ranked first. The above analyses fully verify the effectiveness and reasonability of the proposed retrieval paradigm for RAG, *i.e.,* FUNNELRAG, whose main design ideas are large-to-small quantities, coarse-to-fine granularity, as well as simple-to-complex models.

Datasets	Retrieval Paradigm	Retrieval Stages			Time Cost (T)	Answer Recall (AR)
		Retrieval	Pre/Re-ranking	Post-ranking		
NQ	Flat Retrieval	21M → 1p	N/A	N/A	4.90 (4.90+N/A+N/A)	51.16
		21M → 400p	400p → 1p	N/A	5.25 (4.90+0.35+N/A)	56.09
	Progressive Retrieval	600K → 1c	N/A	N/A	0.00 (0.00+N/A+N/A)	52.74
		600K → 20c	20c $\xrightarrow{\sim 180d}$ 1d	N/A	0.49 (0.00+0.49+N/A)	63.46
		600K → 80c	80c $\xrightarrow{\sim 740d}$ 8d	8d $\xrightarrow{\sim 50p}$ 1p	2.97 (0.00+2.20+0.77)	63.74
TQA	Flat Retrieval	21M → 1p	N/A	N/A	5.02 (5.02+N/A+N/A)	60.29
		21M → 400p	400p → 1p	N/A	5.41 (5.02+0.39+N/A)	71.37
	Progressive Retrieval	600K → 1c	N/A	N/A	0.00 (0.00+N/A+N/A)	66.48
		600K → 20c	20c $\xrightarrow{\sim 200d}$ 1d	N/A	0.60 (0.00+0.60+N/A)	72.59
		600K → 80c	80c $\xrightarrow{\sim 800d}$ 12d	12d $\xrightarrow{\sim 65p}$ 1p	3.47 (0.00+2.52+0.95)	74.57

Table 10: Retrieval performance comparison (in Top-1) *w.r.t.* time cost and answer recall on NQ and TQA datasets.

Datasets	Retrieval Paradigm	Retrieval Stages			Time Cost (T)	Answer Recall (AR)
		Retrieval	Pre/Re-ranking	Post-ranking		
NQ	Flat Retrieval	21M → 2p	N/A	N/A	4.90 (4.90+N/A+N/A)	64.21
		21M → 400p	400p → 2p	N/A	5.25 (4.90+0.35+N/A)	67.34
	Progressive Retrieval	600K → 2c	N/A	N/A	0.00 (0.00+N/A+N/A)	63.57
		600K → 20c	20c $\xrightarrow{\sim 180d}$ 2d	N/A	0.49 (0.00+0.49+N/A)	70.17
		600K → 80c	80c $\xrightarrow{\sim 740d}$ 8d	8d $\xrightarrow{\sim 50p}$ 2p	2.97 (0.00+2.20+0.77)	71.36
TQA	Flat Retrieval	21M → 2p	N/A	N/A	5.02 (5.02+N/A+N/A)	68.98
		21M → 400p	400p → 2p	N/A	5.41 (5.02+0.39+N/A)	77.29
	Progressive Retrieval	600K → 2c	N/A	N/A	0.00 (0.00+N/A+N/A)	73.58
		600K → 20c	20c $\xrightarrow{\sim 200d}$ 2d	N/A	0.60 (0.00+0.60+N/A)	77.13
		600K → 80c	80c $\xrightarrow{\sim 800d}$ 12d	12d $\xrightarrow{\sim 65p}$ 2p	3.47 (0.00+2.52+0.95)	78.31

Table 11: Retrieval performance comparison (in Top-2) *w.r.t.* time cost and answer recall on NQ and TQA datasets.

Datasets	Retrieval Paradigm	Retrieval Stages			Time Cost (T)	Answer Recall (AR)
		Retrieval	Pre/Re-ranking	Post-ranking		
NQ	Flat Retrieval	21M → 3p	N/A	N/A	4.90 (4.90+N/A+N/A)	69.64
		21M → 400p	400p → 3p	N/A	5.25 (4.90+0.35+N/A)	72.83
	Progressive Retrieval	600K → 3c	N/A	N/A	0.00 (0.00+N/A+N/A)	68.98
		600K → 20c	20c $\xrightarrow{\sim 180d}$ 3d	N/A	0.49 (0.00+0.49+N/A)	72.71
		600K → 80c	80c $\xrightarrow{\sim 740d}$ 8d	8d $\xrightarrow{\sim 50p}$ 3p	2.97 (0.00+2.20+0.77)	73.88
TQA	Flat Retrieval	21M → 3p	N/A	N/A	5.02 (5.02+N/A+N/A)	73.11
		21M → 400p	400p → 3p	N/A	5.41 (5.02+0.39+N/A)	79.88
	Progressive Retrieval	600K → 3c	N/A	N/A	0.00 (0.00+N/A+N/A)	76.96
		600K → 20c	20c $\xrightarrow{\sim 200d}$ 3d	N/A	0.60 (0.00+0.60+N/A)	79.04
		600K → 80c	80c $\xrightarrow{\sim 800d}$ 12d	12d $\xrightarrow{\sim 65p}$ 3p	3.47 (0.00+2.52+0.95)	79.83

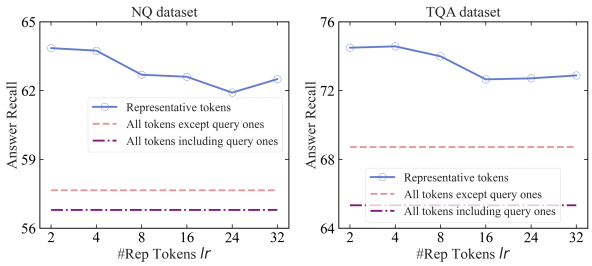
Table 12: Retrieval performance comparison (in Top-3) *w.r.t.* time cost and answer recall on NQ and TQA datasets.

D Cluster Document Algorithm

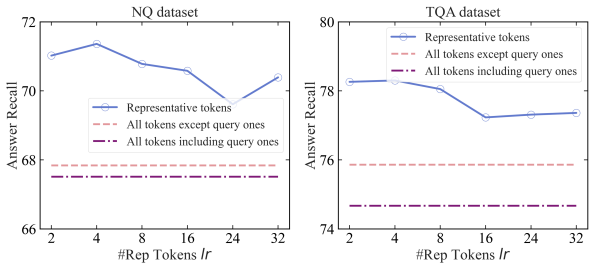
Algorithm 1 shows the document clustering algorithm used in the retrieval stage (§3.1.1). Simply put, documents with highly relevant will be clustered together, where each cluster is a list of documents related to each other. And, the clustered documents serve as the coarse-grained retrieval units.

Question: Who won the fifth season of America's Got Talent? **Answer:** ["Soul singer Michael Grimm", "Michael Grimm"]
(1) Retrieval Stage (clustered documents): (..., Miss Kansas USA, America's Got Talent (season 9), Hurricane (Nick Fradiani album), ...); (..., Preacher Lawson, **America's Got Talent**, Jodi Miller, ...); (..., Kite line, **Michael Grimm (album)**, **America's Got Talent (season 5)**, **Michael Grimm (musician)**, Connor Doran, ...); (America's Got Talent (season 4), Kevin Skinner, The Spiritual Harmonizers, ...) .....
(2) Pre-ranking Stage (documents): **America's Got Talent (season 5)**; **Michael Grimm (musician)**; America's Got Talent (season 12); America's Got Talent (season 13); **America's Got Talent**; Got Talent; America's Got Talent (season 9) .....
(3) Post-ranking Stage (passages): **America's Got Talent (season 5)**; **Michael Grimm (musician)**; America's Got Talent; **America's Got Talent (season 5)**.

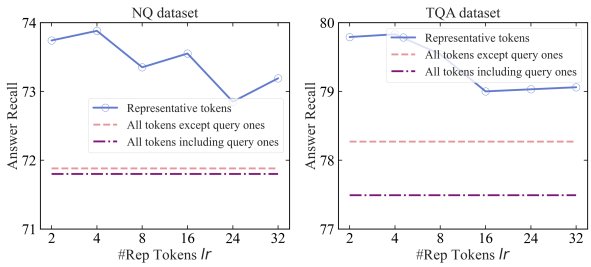
Table 13: An example of FUNNELRAG's data flow from NQ dataset, where the retrieved information that contains the answer string is marked in **green**. Note that we only present the title of the retrieved information for brevity.



(a) Impact of #rep tokens (in Top-1).



(b) Impact of #rep tokens (in Top-2).



(c) Impact of #rep tokens (in Top-3).

Figure 6: Answer recall comparison (in Top-1, Top-2, and Top-3) of FUNNELRAG in terms of different numbers of representative tokens. #Rep tokens is the abbreviation of "the number of representative tokens".

Algorithm 1 Cluster Document Algorithm

Input: $\mathcal{D}$ (documents), S (max cluster size), $\text{adj}[d]$ (related documents for each d)
Output: $\mathcal{C}$ (set of clusters)

```
Sort D by their local cluster coefficient;
Initialize an empty set of clusters C ← ∅;
for each document d in D do
    Add {d} to C;
end for
for each document d in D do
    Remove {d} from C;
    R ← FIND_RELATED_CLUSTER(d, C);
    Create a new cluster c_new = {d};
    Sort R by their closeness centrality to d;
    for each cluster c in R do
        if |c_new| + |c| ≤ S then
            c_new ← c_new ∪ c;
            Remove c from C;
        end if
    end for
    Add c_new to C;
end for
return C;

function FIND_RELATED_CLUSTER(d, C)
    Initialize the set of related clusters R ← ∅;
    for each related document d' in adj[d] do
        for each cluster c in C do
            if d' ∈ c then
                R ← R ∪ {c};
            end if
        end for
    end for
    return The set of related clusters R;
end function
```