On Synthetic Data Strategies for Domain-Specific Generative Retrieval

Haoyang Wen[‡], **Jiang Guo**[†*], **Yi Zhang**[†], **Jiarong Jiang**[†], **Zhiguo Wang**[†]
[‡]Language Technologies Institute, Carnegie Mellon University [†]AWS AI
hwen3@cs.cmu.edu
{gujiang, imyi, jiarongj, zhiguow}@amazon.com

Abstract

This paper investigates synthetic data generation strategies in developing generative retrieval models for domain-specific corpora, thereby addressing the scalability challenges inherent in manually annotating in-domain queries. We study the data strategies for a two-stage training framework: in the first stage, which focuses on learning to decode document identifiers from queries, we investigate LLM-generated queries across multiple granularity (e.g. chunks, sentences) and domain-relevant search constraints that can better capture nuanced relevancy signals. In the second stage, which aims to refine document ranking through preference learning, we explore the strategies for mining hard negatives based on the initial model's predictions. Experiments on public datasets over diverse domains demonstrate the effectiveness of our synthetic data generation and hard negative sampling approach.

1 Introduction

Generative retrieval is emerging as a promising paradigm for information retrieval (IR), leveraging generative models (*e.g.,* Transformers, Vaswani et al., 2017) to directly produce ranked lists of potentially relevant document identifiers for a user query. Although prior work has made progress on various fronts, including training strategies (*e.g.,* identifier choices) (Tay et al., 2022; Zhou et al., 2022; Bevilacqua et al., 2022; Sun et al., 2023), modeling techniques (Chen et al., 2023b; Zhou et al., 2023; Li et al., 2024b), and inference methods (Cao et al., 2020; Lee et al., 2023a; Zhang et al., 2024), the role of *data strategies* in training generative retrieval models, particularly when dealing with domain-specific corpora, remains relatively underexplored. This gap is critical: as generative retrieval models internalize entire corpus within their parametric memory, the choice and quality of training data are likely to play a critical role in their performance.

To mitigate the high cost and scalability challenges of in-domain annotation, most studies have adopted DSI-QG (Zhuang et al., 2022a), which uses passage-level synthetic queries generated by docT5query (Nogueira et al., 2019) (a model trained on MS-MARCO data). However, applying such off-the-shelf synthetic data strategies to new domains may not suffice. Unlike dense retrieval approaches, which focuses on strong text representation (Karpukhin et al., 2020; Izacard et al., 2021), generative retriever must develop three key capabilities: (1) **memorization** (storing the content of the corpus (*e.g.,* documents) and mapping them to their assigned identifiers), (2) **generalization** (inferring beyond explicit textual cues from user queries), and (3) **relevance scoring** (accurately ranking document identifiers by relevance to a given query). Domain-specific corpora can amplify these challenges, as the model must adapt its internal representations to reflect domain nuances while maintaining robust generalization and ranking accuracy. In this work, we systematically investigate data strategies that can foster these core capabilities.

We introduce a two-stage training framework. The first stage focuses on mapping an input directly to document identifiers via supervised fine-tuning on synthetic data. The second stage uses preference learning to further enhance the ranking performance (Zhou et al., 2023; Li et al., 2024b). Here we adopt Regularized Preference Optimization (Pang et al., 2024, RPO), an effective alternative to PPO-based reinforcement learning (Ouyang et al., 2022). We study the data strategies for both stages.

The first stage focuses on the memorization and generalization ability. We examine two data sources as the input for decoding document identifiers during training: the *context* data (*e.g.,* chunks) directly extracted from the corpus, and *synthetic queries* that represent various relevance signals.

[*]Corresponding author

For synthetic queries, we investigate query generation using multi-granular context (*e.g.,* sentence-level, chunk-level) to capture both local and global information from the corpus. We also explore adding constraints derived from available metadata or domain-specific knowledge when generating synthetic queries to enhance the model's ability of handling complex, domain-relevant queries.

In the first stage, models are optimized to produce a single positive candidate, without considering relevance between different candidates. In the second stage, we further create data to enhance the model's ranking capability through preference learning (Zhou et al., 2023; Li et al., 2024b). We study the selection of negative candidate documents for preference learning. Rather than relying on static offline data, we collect preference data online from the model's own top-ranked candidates after the first stage, and compare it to random sampling from the corpus. We further investigate the choices and impact of varying the number of negative candidates to the ranking performance.

We conduct experiments on datasets covering various aspects of relevance, including the widely adopted Natural Questions (Kwiatkowski et al., 2019, NQ), a multi-hop dataset MultiHop-RAG (Tang and Yang, 2024), and two perspective-based retrieval datasets: AllSides (Baly et al., 2020) and AGNews (Yu et al., 2023) from Zhao et al. (2024). We show that queries with different aspects, such as multi-granular and constrains-based queries, significantly improve the retrieval performance compared to relying solely on chunk-level synthetic queries from query generation models. Additionally, upsampling context data further improves the performance. Moreover, we show that these data strategies generalize well to other types of document identifiers, such as atomic identifiers. Finally, we demonstrate that RPO effectively improve the ranking performance of generative retrieval, and the key lies in the selection of high-quality negative candidates: high-quality hard negative candidates improve the performance while random negatives may have an adverse impact.

In summary, this work offers a comprehensive investigation of data strategies for building scalable and effective domain-specific generative retrieval systems. Our findings emphasizes the importance of creating high-quality and diverse synthetic queries that capture multiple levels of granularity within the corpus, as well as informed negative selection strategies for ranking optimization.

2 Generative Retrieval Framework

A typical generative retrieval framework takes a query as input, and generates the corresponding relevant document identifiers as the retrieval results (Tay et al., 2022). Because each document in the corpus has a unique identifier, one can then use these identifiers to retrieve the corresponding documents for downstream tasks.

2.1 Document Identifiers

We primarily use *semantic* document identifiers in our experiments due to their superior performance and better scalability to larger corpora. Instead of using corpus-specific semantic identifiers like titles or urls, we adopt a more general, keyword-based approach that can be applied to a wide variety of corpora (Zhou et al., 2022). Specifically, we instruct an LLM to produce a list of keywords that describes the content of a document, and use this keyword list as its semantic identifier.

In addition, we extended our synthetic data strategies to other types of identifiers to validate its generalizability, such as atomic identifiers (Tay et al., 2022), which are unique tokens that can be generated through a one-step decoding or classification process.

2.2 Generative Modeling

The generative retrieval model learns to generate the identifier of a relevant document given a query. Formally, for a query q and a relevant document d with identifier d', generative retrieval aims to produce d' given q, which can be represented as:

$$\begin{aligned}\text{score}(q, d) &= P\left(d' \mid q; \theta\right) \\ &= \prod_i P\left(d'_i \mid d'_{<i}, q; \theta\right),\end{aligned}$$

where d'_i is the ith token of the identifier. To ensure the generated identifiers are valid during inference, we use constrained beam search with Trie (Cao et al., 2020) to restrict the output token space at each decoding step. The top-k output from the beam search serves as the final retrieval results.

Compared to dense retrieval models (Karpukhin et al., 2020), generative retrieval bypasses the need for an external index by directly producing relevant document identifiers. However, there are distinct challenges in learning a generative retrieval model. As it solely relies on parametric knowledge, the model must not only learn the retrieval task, but also capture and encode document content in a way

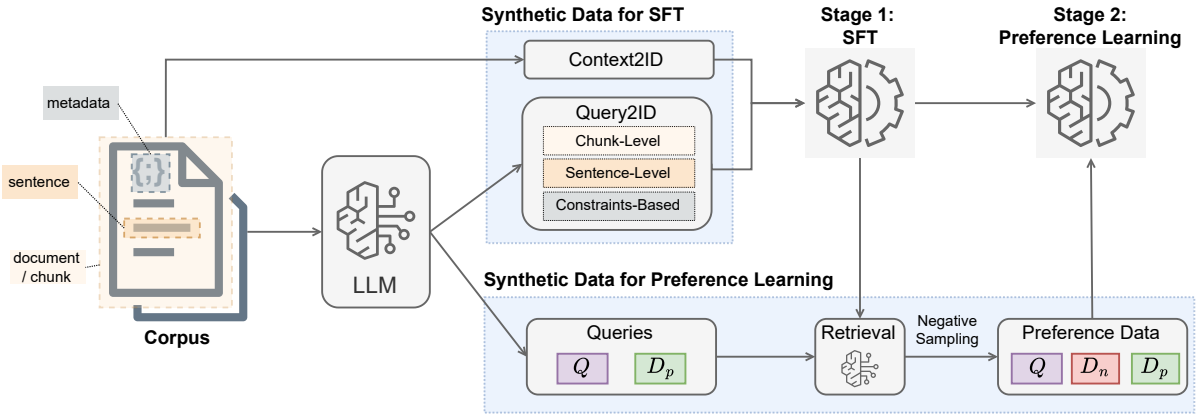

Figure 1: The overall workflow of the generative retrieval training and synthetic data utilization at each stage.

that associates each document with its identifier. Therefore, generative retrieval often requires training on the entire corpus to enable the model to memorize and comprehend the necessary information effectively.

3 Supervised Fine-Tuning Data Strategy

In a typical domain-specific setup, we often assume access to a corpus with limited or no labeled data for domain-specific training (Hashemi et al., 2023). Therefore, it is crucial to create high-quality synthetic data that thoroughly covers the entire corpus for generative retrieval training.

Our synthetic data comprises two main components: Context2ID data and Query2ID data. Context2ID involves training the model to retrieve the document identifiers given the document's content. Query2ID focuses on teaching the model to retrieve relevant document identifiers from a given query. Combining these two data types encourages the model to learn both content memorization and retrieval given a query.

3.1 Supervised Fine-Tuning Objective

At this stage, we train the model to generate relevant document identifiers by maximizing the probability of each individual token. While typical supervised fine-tuning (SFT), especially with encoder-decoder architectures such as T5, focuses on optimizing the output sequence (*i.e.* document identifiers), it's also part of the training goal for generative retrieval models to comprehend and memorize the context. To this end, we also optimize the model for learning to decode the input. Specifically, for a given query-document pair (q, d), where q could be an actual query or a text chunk from the document, the model maximizes the likelihood of the combined input and output sequence:

$$\begin{aligned}\mathcal{L}_{\text{sft}}(q, d) &= -\log P\left(d', q; \theta\right) \\ &= -\sum_i \log P(q_i \mid q_{<i}; \theta) \\ &\quad -\sum_i \log P(d'_i \mid d'_{<i}, q; \theta).\end{aligned}$$

3.2 Context2ID

Context2ID data is created by pairing each chunk of text in the corpus with its corresponding document identifier. The goal of Context2ID data is to help the generative retrieval model associate each document's content with its unique identifier, i.e., "memorizing" the text.

3.3 Query2ID

Query2ID is designed to teach the model to retrieve the relevant document identifiers given a query. It helps the model to learn the core retrieval task and also further comprehend content from the query perspective.

Previous work (Zhuang et al., 2022a) finds it effective to use a query generation model (*e.g.*, docT5query, Nogueira et al., 2019) to produce synthetic queries from documents using multiple independent samplings. In this work, we instead use an LLM for synthetic query generation. Specifically, given a context (e.g., a document chunk), the LLM is instructed to generate a diverse set of m queries, thereby covering a wider range of semantic variations compared to the sampling-based approach with a specialized query generation model.

3.3.1 Multi-Granular Query Generation

We first generate queries with context at different levels of granularity: *chunk-level* and *sentence-level*. Chunk-level synthetic queries are produced

Data Type	Example
Context	title: Christmas Day preview: 49ers, Ravens square off in potential Super Bowl sneak peek... source: Yardbarker ... San Francisco has racked up an NFL-leading 25 turnovers and has given up the second-fewest rushing yards (1,252), ...
Chunk-level query	What is the potential implication of this matchup between the 49ers and Ravens?
Sentence-level query	Where does the 49ers' defense stand in terms of total yards allowed per game?
Constraints-based query	According to the Yardbarker article, which team has the league's most effective running game?

Table 1: Examples of different synthetic queries generated from MultiHop-RAG corpus.

by providing the entire chunk as input to the LLM to capture higher-level semantics or facts, while sentence-level synthetic queries are produced by only providing individual sentences to focus on more specific details within the document Concretely, for each chunk, we ask the LLM to produce m_c chunk-level queries. We then split the chunk into individual sentences and ask the LLM to generate m_s sentence-level queries for each sentence.

3.3.2 Constraints-Based Query Generation

A key advantage of using an LLM for query generation is its ability to incorporate domain-specific instructions. For instance, we can prompt the LLM to include metadata constraints, such as the *author name* or *political polarity* of a document, in the generated queries. Although the specific constraint types depends on the metadata available and can be domain or dataset specific, they are common in real-world scenarios such as enterprise data. Table 10 in Appendix specifies the attributes that we use to produce constraints-based synthetic queries for each dataset. We ask the LLM to generate m_i queries for each document that incorporate these constraints, allowing our generative retrieval model to handle more specialized or domain-specific queries.

4 Preference Learning Data Strategy

Previous work (Zhou et al., 2023; Li et al., 2024b) have shown that incorporating ranking tasks can further enhance the relevance modeling of generative retrieval models. However, when generative retrieval models are based on large language models, complex ranking objectives – such as listwise optimization – often become computationally inefficient due to multiple forward passes. In this work, we instead use a simpler method and adopt the regularized preference optimization algorithm to perform the preference optimization, a technique widely applied in optimizing large language models. We will first briefly introduce the preference optimization method, and then turn our focus on the synthetic data construction, which consists of the synthetic queries along with their corresponding preferred or rejected candidates.

4.1 Preference Optimization Objective

We use Regularized Preference Optimization (Pang et al., 2024, RPO) as our optimization method for preference learning. It is an extended version of Directed Preference Optimization (Rafailov et al., 2023, DPO), including additional supervised fine-tuning loss to alleviate the over-optimization issues on negative responses. It takes an input query q, a positive candidate d_p, and a negative candidate d_n as input. The loss is in favor of the positive candidate while against the negative candidate

$$\mathcal{L}_{\text{rpo}}(q, d_p, d_n) = -\log\delta\left(\beta\log\frac{P(d'_p \mid q;\theta)}{P(d'_p \mid q;\theta_{\text{ref}})} - \beta\log\frac{P(d'_n \mid q;\theta)}{P(d'_n \mid q;\theta_{\text{ref}})}\right) - \alpha\frac{\log P(d'_p \mid q;\theta)}{|d'_p|},$$

where θ_{ref} is the parameter of the reference model, *i.e.,* the supervised fine-tuned model from the first stage training. d'_p and d'_n are the identifiers of the positive and negative candidate, respectively.

4.2 Synthetic Queries

Similar to the previous section, in a domain-specific setup, we assume that we do not have enough data for model training. Therefore, after the supervised fine-tuning stage, we need a batch of new synthetic queries for preference learning.

We still adopt the LLM-based query generation as with the supervised fine-tuning stage. However, there are a few key differences in the instructions. First of all, we ask the LLM to make queries as difficult as possible. At the same time, we ask the LLM to provide not only the synthetic queries but

also their corresponding answers. This is to ensure that, while making difficult queries, those synthetic queries are still answerable using the given context.

These changes make the new batch of synthetic queries different from queries used during supervised fine-tuning so that the model will not be over-optimized to the same batch of data. Intensifying the difficulties also increases the likelihood that the initial generative retrieval model makes mistakes, and therefore the model will benefit from the preference learning by learning from those mistakes.

4.3 Candidate Selection

After producing the synthetic queries, the next step is to select document candidate pairs for RPO optimization. For each training instance, we need one positive candidate and one negative candidate. As we always produce synthetic queries based on a document, the positive candidate can be naturally assigned. Therefore, the focus will be on selecting negative candidates for each synthetic query.

To increase the hardness of the negative candidates, we choose to select negative candidates from the retrieval results. Specifically, after the supervised fine-tuning stage, we will use the generative retrieval model to perform retrieval on the synthetic queries for preference learning. Our strategy mainly focuses on selecting the top-k negative candidates with ranks higher than the positive candidate from the retrieval results. In this way, if the positive candidate ranks in the top-1, we will not use the query for preference learning. If the rank of the positive candidate is higher than k, then there will be different numbers of negative candidates, depending on the rank. If the rank is lower than k, there will be k different negative candidates. When there are multiple negative candidates, we pair each negative candidate with the positive one to form a candidate pair instance for preference learning.

5 Experiments

5.1 Datasets

We choose 4 datasets for our experiments: three domain-specific corpora – MultiHop-RAG (Tang and Yang, 2024), –AllSides (Baly et al., 2020) and AGNews (Yu et al., 2023) from Zhao et al. (2024) – as well as the general-domain dataset Natural Questions dataset (Kwiatkowski et al., 2019, NQ).

For AllSides and AGNews, we mainly adopt queries from Zhao et al. (2024). In the case of AGNews, we replace the similar document part in queries with another attribute of perspective, as we focus on the query retrieval rather than document similarity search.

For NQ, we use the "old document" split from Kishore et al. (2023), which constructs a subset of Wikipedia pages containing all positive candidates for training and testing, while keeping the corpus size manageable for generative retrieval training.

Context Source	HIT@4	HIT@10	MAP@10	MRR@10
Chunk	43.64	66.65	13.98	31.14
Chunk+Sent	61.64	81.69	22.13	47.20

Table 2: Ablation study on the effect of using synthetic queries generated with context at different levels of granularity on Multihop-RAG.

5.2 Experiment Setup

For all datasets, we use Mistral 7b (Jiang et al., 2023) series as the generative retrieval base model. We use Mixtral 8x7b (Jiang et al., 2024) to generate all the synthetic queries and we use Claude 3 Sonnet (Anthropic, 2024) to generate keywords. Please refer to Appendix A for more details on the training infrastructure, hyperparameters, dataset-specific setup and statistics.

5.3 Results

We will discuss our experiment results for each of the stages. In the supervised fine-tuning stage, we will discuss the effects of multi-granular synthetic queries, synthetic data with domain-specific constraints, and the use of Context2ID data. For the preference learning stage, we will discuss using different candidates for preference learning.

5.3.1 Supervised Fine-Tuning Stage

Effects of multi-granular synthetic queries. We conduct an analysis on the effects of incorporating synthetic queries generated from the context at different levels of granularity on MultiHop-RAG. We train the generative retrieval model based on semantic identifiers on chunk-level Query2ID data (Chunk), comparing it with the model trained on chunk-level and sentence-level Query2ID data (Chunk+Sent), and both models use Context2ID data. The results are shown in Table 2. We find that sentence-level synthetic queries can significantly improve retrieval performance, indicating that synthetic query generation with a small context can help capture more details from the document.

Effects of constraints-based synthetic queries. We further study the use of constraints-based

	MultiHop-RAG				AllSides			AGNews		
	HIT@4	HIT@10	MAP@10	MRR@10	HIT@1	HIT@5	HIT@10	HIT@1	HIT@5	HIT@10
w/o constraints	61.64	81.69	22.13	47.20	10.19	29.63	47.22	59.91	83.94	88.11
w/ constraints	69.98	88.34	24.85	52.29	14.20	38.58	51.85	62.19	83.78	88.24

Table 3: Ablation study on generative retrieval performances with or without the constraints-based synthetic queries.

	MultiHop-RAG				Natural Questions			
	HIT@4	HIT@10	MAP@10	MRR@10	HIT@1	HIT@5	HIT@10	MRR@10
w/o Context2ID	41.33	69.31	14.45	31.25	69.72	85.58	89.01	76.57
w/ Context2ID	69.98	88.34	24.85	52.29	70.71	86.48	89.85	77.54

Table 4: Ablation study on generative retrieval performance trained with or without Context2ID data. The results demonstrate the helpfulness of Context2ID data and learning to memorize the context for generative retrieval.

	HIT@4	HIT@10	MAP@10	MRR@10
Concat	44.30	72.77	15.64	33.59
Interleave	69.98	88.34	24.85	52.29

Table 5: Analysis on different ways of combining Query2ID and Context2ID data on Multihop-RAG. We compare simple concatenation (Concat) and interleaving (Interleave) that inherently upsamples the Context2ID data.

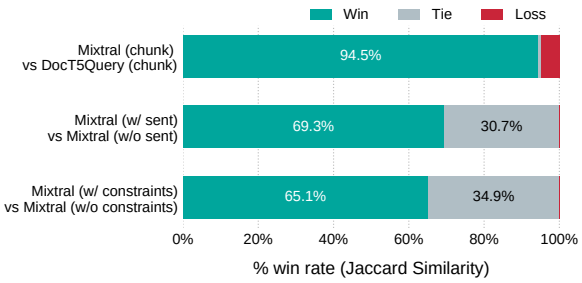

Figure 2: Jaccard similarity post-analysis on MultiHop-RAG test set. Synthetic queries from Mixtral 8x7b are generally closer to the test set than those from docT5query. Besides, incorporating granularity and domain-specific attributes further helps with getting queries that are closer to the test set.

synthetic queries that are customized for each domain-specific setting. We conduct experiments on three domain-specific corpora, MultiHop-RAG, AllSides, and AGNews. We compare the semantic identifier-based generative retrieval model trained with or without constraints-based synthetic queries, combing with the corresponding Context2ID data. The results are shown in Table 3. The results show that constraints-based synthetic queries can further improve retrieval performance, indicating that it is helpful to use LLM-produced synthetic queries for domain customization.

Effects of Context2ID data. Existing work (Zhuang et al., 2022a; Kishore et al., 2023) debates whether Context2ID data are useful for generative retrieval training. In this work, we consider Context2ID data as an important part of the data recipe, and also include the memorization of the context as part of the supervised fine-tuning objective. Therefore, we conduct an analysis that removes the Context2ID data on MultiHop-RAG and NQ, and the results are shown in Table 4. We can find that Context2ID data consistently improve generative retrieval performance. We also include the comparison of the strategies to combine Query2ID and Context2ID data, including simple concatenation or interleaving Context2ID and Query2ID that will upsample Context2ID data (*i.e.,* smaller size dataset) on MultiHop-RAG in Table 5, again illustrating the importance of Context2ID and learning to memorize context may strengthen the effects on Context2ID.

Different query generation models. As we primarily use LLMs to produce synthetic queries, it is important to understand the performance and effects of using an LLM compared to a specialized query generation model. To this end, we conduct a comparison between synthetic queries generated by Mixtral 8x7b and those from docT5query, as shown in Table 6. For a fair comparison, we exclude constraints-based queries because docT5query cannot produce these types of queries. The results show generative retrieval models trained with queries from Mixtral 8x7b consistently outperform models trained on queries from docT5query.

Following Pradeep et al. (2023), we use Jaccard similarity to evaluate the semantic similarity between test queries and synthetic queries as a post-analysis. For each test query, we compute the

	MultiHop-RAG				Natural Questions			
	HIT@4	**HIT@10**	**MAP@10**	**MRR@10**	**HIT@1**	**HIT@5**	**HIT@10**	**MRR@10**
docT5query	50.86	73.30	17.60	37.73	63.30	79.12	85.18	70.30
Mixtral 8x7b	61.64	81.69	22.13	47.20	70.71	86.48	89.85	77.54

Table 6: Generative retrieval performance with synthetic queries from Mixtral 8x7b and docT5query. The results show that queries from Mixtral 8x7b can help train a better generative retrieval model.

	HIT@4	**HIT@10**	**MAP@10**	**MRR@10**
all	74.32	88.03	29.71	59.26
w/o Context2ID	72.15	86.21	28.54	57.50
w/o Sent	58.40	75.17	21.51	44.76
w/o constraints	68.34	83.73	26.28	53.83

Table 7: Ablation study on atomic identifier-based generative retrieval performance on MultiHop-RAG.

maximum Jaccard similarity among all synthetic queries generated for the corresponding chunk or document. We then compute the win rate between two synthetic query sets (e.g., Mixtral 8x7b versus docT5query) as the proportion of test queries for which one set exhibits higher Jaccard similarity. Let Q be the set of test queries. Then, the win rate for a synthetic query set $\mathcal{S}$ over another set $\mathcal{T}$ is defined as:

$$\text{Win Rate}(\mathcal{S}, \mathcal{T}) = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{I}\left[\max_{s \in \mathcal{S}(q)} J(q,s) > \max_{t \in \mathcal{T}(q)} J(q,t)\right],$$

where $J(q, s)$ denotes the Jaccard similarity between the token sequences of the test query q and a synthetic query s, and $\mathcal{S}(q)$ represents the subset of synthetic queries for the corresponding chunk.

Figure 2 illustrates that synthetic queries from Mixtral 8x7b generally have a higher similarity to test queries. Moreover, both sentence-level synthetic queries and constraints-based queries contribute to improved distribution matching with the test queries.

Generalization to different identifiers. We further study the generalizability of our data strategies across different types of document identifiers. In this analysis, we use atomic identifier, which are arbitrary unique IDs assigned to each document or chunk. We conduct experiments on MultiHop-RAG and the results are shown in Table 7. The findings align with our observations using semantic identifiers, highlighting the critical role of all three data types in generative retrieval. Among them, sentence-level synthetic queries contribute the most to performance improvements.

5.3.2 Preference Learning Stage

Effects of negative candidate sources. We first study the strategies of candidates selection for preference learning. We compare random selection from the corpus with using the top candidates from the generative retrieval model after the supervised fine-tuning stage. The results are shown in Table 8, which illustrates that candidate selection has an impact on preference learning, and random candidates may have a negative impact.

Effects of negative candidate number. We also study the effects of using different negative candidate numbers for each query. We experiment with selecting Top-5 and Top-10 negative candidates with a rank higher than the positive candidate from the retrieval results. The results are shown in Table 8. In general, it is effective to use the strategy, which includes high-quality candidates with ranks higher than the corresponding positive candidates. We also see some slight differences when including different numbers of negative candidates. We can find that a large number of negative candidates helps better in metrics such as HIT@1 and HIT@4.

5.3.3 Comparison to Off-The-Shelf Retrievers

We also compare the performance of our generative retrieval with several off-the-shelf retrievers, including BM25 (Robertson and Walker, 1994), bge-large-en-v1.5 (Xiao et al., 2023), Contriever-msmarco (Izacard et al., 2021), E5-mistral-7b-instruct (Wang et al., 2024) and GTE-Qwen2-7B-instruct (Li et al., 2023c). The results are shown in Figure 3 and more detailed results can be found in Appendix B.2. The results demonstrate that generative retrieval models, which rely solely on in-domain synthetic data training without retrieval pre-training, can achieve competitive performance compared to those retrievers. These results indicate the potential of generative retrieval, as well as the effectiveness of using LLMs to generate synthetic data tailored to domain-specific needs.

	MultiHop-RAG				Natural Questions			
	HIT@4	HIT@10	MAP@10	MRR@10	HIT@1	HIT@5	HIT@10	MRR@10
SFT	69.98	88.34	24.85	52.29	70.71	86.48	89.85	77.54
Random 5	58.94	82.88	20.88	43.53	70.19	86.48	89.50	77.17
Top-5 negative	71.53	89.62	26.36	55.40	71.02	87.32	90.04	78.02
Top-10 negative	71.88	89.80	26.23	54.94	71.22	87.41	89.97	78.14

Table 8: Preference learning with different numbers of negative candidates. The results show that it is an effective strategy to select negative candidates with ranks higher than the positive candidate, while different numbers of negative candidates may optimize the retrieval performance in different ways.

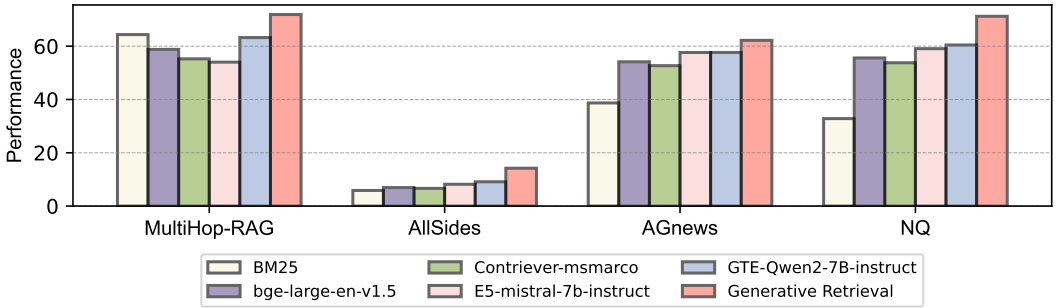

Figure 3: Performance comparison between generative retrieval using semantic identifiers and off-the-shelf-retrieval models. We use HIT@4 for MultiHop-RAG and HIT@1 for other datasets as the evaluation metric. For the full experiment results, please refer to Appendix B.2 (Table 12).

6 Related Work

Generative retrieval modeling. Previous work has explored various aspects of generative retrieval. One line of research aims to find appropriate document identifiers for generation, such as numerical or atomic identifier (Tay et al., 2022; Zhuang et al., 2022a; Zhou et al., 2022), N-grams (Bevilacqua et al., 2022; Chen et al., 2023a), titles or URLs (Li et al., 2023a; Lee et al., 2022; Chen et al., 2022; Ziems et al., 2023), keywords-based or summary-based semantic identifiers (Lee et al., 2023b; Tang et al., 2023), codebook (Zhang et al., 2023; Yang et al., 2023; Zeng et al., 2024), and full passages themselves (Tang et al., 2024a). There are also efforts to combine the advantages of different identifiers (Li et al., 2023b). Another line of work tackles the optimization of generative retrieval models, such as incorporating ranking losses (Zhou et al., 2023; Li et al., 2024b; Tang et al., 2024b), or using auxiliary tasks to enhance training (Li et al., 2024a). During retrieval, different constrained decoding methods have been explored to obtain valid identifiers, such as FM-Index (Bevilacqua et al., 2022), Trie-based (Cao et al., 2020), and set-based inference (Tang et al., 2024b).

Synthetic query generation. Alongside the progress in generative retrieval modeling and optimization, synthetic query generation has emerged as a pivotal technique for enhancing retrieval systems, particularly in domains with limited annotated data. In dense retrieval, synthetic queries have been used extensively to improve cross-domain performance. For instance, Ma et al. (2021) generated synthetic questions for target-domain documents with a question generation model trained on general-domain data, thereby improving retrieval performance in zero-shot settings. Similarly, Wang et al. (2022) introduced generative pseudo labeling, which combines query generation with pseudo labeling using a cross-encoder to capture finer-grained ranking signals. Further advancements include Bonifacio et al. (2022) and Jeronymo et al. (2023), which leverage large language models to generate synthetic queries in a few-shot manner, and then combine with top K documents ranked by the conditional question generation probability, to train a domain-specific reranker.

Despite these successes in dense retrieval, the potential of synthetic data for generative retrieval has been underexplored. Existing studies typically rely on passage-level synthetic queries generated by docT5query (Nogueira et al., 2019), following the DSI-QG paradigm (Zhuang et al., 2022a). Chen et al. (2023b) explores breaking documents into text fragments for query generation and memo-

rization. However, there still lacks a comprehensive discussions on effective strategies for generating synthetic data tailored to domain-specific corpora, especially with LLMs. This work investigates data strategies from multiple perspectives, including the generation of synthetic queries using multi-granularity contexts, incorporating search constraints, and exploring the impact of context data. For preference learning, Zhou et al. (2023) proposes using preference learning objectives for generative retrieval with specialized reward models, though acquiring such models in a domain-specific setting can be challenging. In contrast, our proposed preference learning strategy directly uses the retrieval results to obtain the preference data, offering a more streamlined approach for domain-specific applications.

Similar synthetic-data paradigms have also been explored in Text-to-SQL tasks. Kobayashi et al. (2025) propose YORO, which internalizes database schema and content via LLM-generated NLQ–SQL pairs so that at inference time no schema encoding or value retrieval is required. This parallels our goal of teaching a generative retriever to internalize a domain corpus, and suggests a broader applicability of synthetic data strategies for LLM customization.

7 Conclusion

In this work, we explore several strategies to produce synthetic data for generative retrieval training. We find that adding queries in multi-granularity and queries with domain-specific constraints can largely improve the generative retrieval performance during supervised fine-tuning, and memorizing document contents can also contribute to the generative retrieval training. We also find that it is critical to choose high-quality hard negative candidates to effectively use the preference learning objectives to further improve generative retrieval.

Limitations

Our proposed synthetic data strategies mainly focus on the supervised fine-tuning and preference learning stage. But there are also settings that can largely improve the usability of generative retrieval, such as incremental learning or generalization to unseen documents. It is also important to extend the data strategy exploration for these settings. In addition, similar data strategies may also be effectively used to enhance dense retrieval domain adaptation. Further systematic research is needed to investigate the strategies for dense retrieval model fine-tuning, as well as the differences between generative and dense model training.

Our synthetic queries are mainly based on one document. However, queries from the real world may be more complex, such as those involving multiple documents with multi-hop reasoning or multi-evidence comparison. It is still under-investigation that to generate those complex queries and to use those queries during retrieval model training.

References

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1.

Ramy Baly, Giovanni Da San Martino, James Glass, and Preslav Nakov. 2020. We can detect your bias: Predicting the political ideology of news articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4982–4991, Online. Association for Computational Linguistics.

Michele Bevilacqua, Giuseppe Ottaviano, Patrick S. H. Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. *arXiv preprint arXiv:2202.05144*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *CoRR*, abs/2010.00904.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2023a. A unified generative retriever for knowledge-intensive language tasks via prompt learning. 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 1448–1457. ACM.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. Corpusbrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 191–200. ACM.

Xiaoyang Chen, Yanjiang Liu, Ben He, Le Sun, and Yingfei Sun. 2023b. Understanding differential

search index for text retrieval. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10701–10717, Toronto, Canada. Association for Computational Linguistics.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

Helia Hashemi, Yong Zhuang, Sachith Sri Ram Kothur, Srivas Prasad, Edgar Meij, and W. Bruce Croft. 2023. Dense retrieval adaptation using target domain description. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2023, Taipei, Taiwan, 23 July 2023*, pages 95–104. ACM.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. 2023. Inpars-v2: Large language models as efficient dataset generators for information retrieval. *arXiv preprint arXiv:2301.01820*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *CoRR*, abs/2401.04088.

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

Varsha Kishore, Chao Wan, Justin Lovelace, Yoav Artzi, and Kilian Q. Weinberger. 2023. Incdsi: Incrementally updatable document retrieval. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17122–17134. PMLR.

Hideo Kobayashi, Wuwei Lan, Peng Shi, Shuaichen Chang, Jiang Guo, Henghui Zhu, Zhiguo Wang, and Patrick Ng. 2025. You only read once (YORO): Learning to internalize database knowledge for text-to-SQL. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1889–1901, Albuquerque, New Mexico. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur 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. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Hyunji Lee, JaeYoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vladimir Karpukhin, Yi Lu, and Minjoon Seo. 2023a. Nonparametric decoding for generative retrieval. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12642–12661, Toronto, Canada. Association for Computational Linguistics.

Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. 2022. Generative multi-hop retrieval. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1417–1436, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Sunkyung Lee, Minjin Choi, and Jongwuk Lee. 2023b. GLEN: Generative retrieval via lexical index learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7693–7704, Singapore. Association for Computational Linguistics.

Xiaoxi Li, Zhicheng Dou, Yujia Zhou, and Fangchao Liu. 2024a. Corpuslm: Towards a unified language model on corpus for knowledge-intensive tasks. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 26–37. ACM.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023a. Generative retrieval for conversational question answering. *Inf. Process. Manag.*, 60(5):103475.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2023b. Multiview identifiers enhanced generative retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6636–6648, Toronto, Canada. Association for Computational Linguistics.

Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. 2024b. Learning to rank in generative retrieval. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 8716–8723. AAAI Press.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023c. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. 2021. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1075–1088.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to doctttttquery.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *CoRR*, abs/2404.19733.

Ronak Pradeep, Kai Hui, Jai Gupta, Adam Lelkes, Honglei Zhuang, Jimmy Lin, Donald Metzler, and Vinh Tran. 2023. How does generative retrieval scale to millions of passages? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1305–1321, Singapore. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 232–241. ACM/Springer.

Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Qiaoyu Tang, Jiawei Chen, Zhuoqun Li, Bowen Yu, Yaojie Lu, Cheng Fu, Haiyang Yu, Hongyu Lin, Fei Huang, Ben He, Xianpei Han, Le Sun, and Yongbin Li. 2024a. Self-retrieval: End-to-end information retrieval with one large language model. *Preprint*, arXiv:2403.00801.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *CoRR*, abs/2401.15391.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2023. Semantic-enhanced differentiable search index inspired by learning strategies. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 4904–4913. ACM.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, and Xueqi Cheng. 2024b. List-wise generative retrieval models via a sequential learning process. *ACM Trans. Inf. Syst.*, 42(5):133:1–133:31.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Prakash Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022,*

NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.

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.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2022. Gpl: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2345–2360.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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*, pages 38–45, Online. Association for Computational Linguistics.

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

Tianchi Yang, Minghui Song, Zihan Zhang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. 2023. Auto search indexer for end-to-end document retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6955–6970, Singapore. Association for Computational Linguistics.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander J. Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large language model as attributed training data generator: A tale of diversity and bias. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Hansi Zeng, Chen Luo, Bowen Jin, Sheikh Muhammad Sarwar, Tianxin Wei, and Hamed Zamani. 2024. Scalable and effective generative information retrieval. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 1441–1452. ACM.

Hailin Zhang, Yujing Wang, Qi Chen, Ruiheng Chang, Ting Zhang, Ziming Miao, Yingyan Hou, Yang Ding, Xupeng Miao, Haonan Wang, Bochen Pang, Yuefeng Zhan, Hao Sun, Weiwei Deng, Qi Zhang, Fan Yang, Xing Xie, Mao Yang, and Bin Cui. 2023. Model-enhanced vector index. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Peitian Zhang, Zheng Liu, Yujia Zhou, Zhicheng Dou, Fangchao Liu, and Zhao Cao. 2024. Generative retrieval via term set generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 458–468. ACM.

Xinran Zhao, Tong Chen, Sihao Chen, Hongming Zhang, and Tongshuang Wu. 2024. Beyond relevance: Evaluate and improve retrievers on perspective awareness. *CoRR*, abs/2405.02714.

Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2023. Enhancing generative retrieval with reinforcement learning from relevance feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12481–12490, Singapore. Association for Computational Linguistics.

Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An ultimate retriever on corpus with a model-based indexer. *CoRR*, abs/2208.09257.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022a. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *CoRR*, abs/2206.10128.

Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022b. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*.

Noah Ziems, Wenhao Yu, Zhihan Zhang, and Meng Jiang. 2023. Large language models are built-in autoregressive search engines. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 2666–2678, Toronto, Canada. Association for Computational Linguistics.

Dataset	Context	Queries		
		Chunk-Level	Sentence-Level	Constraints-Based
MultiHop-RAG	7,724	72,090	472,193	51,212
AllSides	645	6,313	173,898	6,091
AGNews	1,050	10,355	80,524	20,875
NQ	98,748	1,459,031	-	-

Table 9: Dataset Statistics

Dataset	Attributes
MultiHop-RAG	author, publish time, source, category, title
AllSides	political polarity
AGNews	location, topic

Table 10: Attributes used in each dataset for constraints-based query generation.

A Details of Experiment Setup

We use `Mistral-7B-Instruct-v0.3` as the base model for generative retrieval with the semantic identifier, while use `Mistral-7B-v0.3` as the base model for atomic identifier as it is closer to a classification setting.

For supervised fine-tuning, we train the models with 2 epochs, with a learning rate of 2e-5 and a warmup ratio of 0.1. The batch size is set as 256. We use sequence packing to put multiple examples in one forward pass (Raffel et al., 2020). We use `bfloat16` for our training.

For preference learning, we mainly conduct experiments on MultiHop-RAG and NQ with semantic identifiers. We train the models with 1 epoch. The learning rate is set as 1e-7, batch size is set as 64, β is set as 0.5, α is set as 1.0.

The training infrastructure includes TRL (von Werra et al., 2020), Accelerate (Gugger et al., 2022), Transformers (Wolf et al., 2020), DeepSpeed (Rasley et al., 2020) and FlashAttention-2 (Dao, 2024). We use 8x Nvidia A100-SXM4-40GB for our experiments. Each training or inference procedure can be completed in 1 day.

Statistics of the numbers of the documents, different synthetic queries can be found in Table 9. Attributes used for constraints-based synthetic queries can be found in Table 10.

A.1 MultiHop-RAG

On MultiHop-RAG, we split the documents into chunks with maximum length of 256 without overlap and conduct retrieval on individual chunks. For synthetic query generation, m_c, m_s and m_i are set as 10, and the temperature for LLM inference on synthetic data generation is set as 0.7. We interleave the Context2ID and Query2ID data as the full dataset for model supervised fine-tuning. The maximum sequence length is set as 700. For synthetic queries for preference learning, we ask the LLM to generate 10 queries. We perform the retrieval with beam size as 10 and retrieve the top-10 candidates for each query to construct the candidate pairs.

A.2 AllSides

On AllSides, we conduct document-level retrieval. For synthetic query generation, m_c, m_s and m_i are set as 10, and the temperature for LLM inference on synthetic data generation is set as 0.7. For Context2ID data, as there are some long documents in the corpus, we will split the long context into chunks with maximum length of 256 without overlap. The Context2ID data is constructed to use all chunks in the document to predict its corresponding document identifier. We interleave the Context2ID and Query2ID data as the full dataset for model supervised fine-tuning. The maximum sequence length is set as 700.

A.3 AGNews

On AGNews, we conduct document-level retrieval. For synthetic query generation, m_c, m_s and m_i are set as 10, and the temperature for LLM inference on synthetic data generation is set as 0.7. Queries constructed by Zhao et al. (2024) use two different perspectives. The first perspective is either the location of the desired news or the topic, while the second perspective is that the news is similar to another given news in the query. As we mentioned Section 5.2, we replace the second perspective with the other field so that each query consists of both location and topic perspectives. The topic and location information used for instruction-based synthetic query generation is extracted with Mixtral 8x7b. We interleave the Context2ID and Query2ID data as the full dataset for supervised fine-tuning. The maximum sequence length is set as 700.

A.4 Natural Questions

On NQ, we conduct document-level retrieval. We use the document prefixes from (Kishore et al., 2023) to produce the semantic identifiers. For synthetic query generation, we perform truncation on pages when they are too long so that we always have at least 1024 token space for model output. We set m_c as 15 and temperature as 0.7. We do not include sentence-level synthetic queries as the number of those queries are too large to be included in training within a reasonable time. Instead, we include sentence-level Context2ID as the approximation, and use the sentences from the document prefixes from (Kishore et al., 2023) to predict corresponding document identifiers. In NQ, we have high quality human annotated training queries, which we also include as part of the Query2ID data and therefore we do not include instruction-based synthetic queries. We concatenate the Context2ID and Query2ID data as the full dataset for model supervised fine-tuning, as interleaving will produce a much larger dataset that cannot be trained within a reasonable time. The maximum sequence length is set as 450. For synthetic queries for preference learning, we also perform truncation as for supervised fine-tuning, and ask the LLM to generate 10 queries. As the generated query number is quite large for inference, we use the first 2 generated queries for each documents for preference learning. We perform the retrieval with beam size as 10 and retrieve the top-10 candidates for each query to construct the candidate pairs.

B Additional Comparisons

B.1 Results on Comparison with Filtered Queries

We conduct another experiment to more closely follow the data pipeline of DSI-QG (Zhuang et al., 2022b), where we generate a set of raw queries using each chunk first and then use the BGE reranker (Xiao et al., 2023) to find the top relevant synthetic queries with docT5query on MultiHop-RAG. DSI-QG is originally trained with T5 models. For a fair comparison, we still train the same Mistral 7b model, compared to using Mixtral 8x7b as the synthetic data generator for chunk-level synthetic queries. The results are shown in Table 11. We can find that our proposed synthetic query strategy still largely outperforms the filtering-based strategy (Chunk-F).

	HIT@4	HIT@10	MAP@10	MRR@10
Chunk-F	30.69	59.78	10.41	23.27
Ours (Chunk)	43.64	66.65	13.98	31.14
Ours (Full)	**69.98**	**88.34**	**24.85**	**52.29**

Table 11: Comparison between the filtered queries that follow DSI-QG data pipeline and our synthetic data generation strategy.

B.2 Results on Comparison with Off-The-Shelf Retrieval Models

The detailed results for each dataset are shown in Table 12. We run the retrieval models on MultiHop-RAG, NQ and AGNews to collect the results, and adopt the results of AllSides from Zhao et al. (2024).

C LLM Prompts

C.1 Prompts for Keywords Generation

Figure 4 shows the prompt for generating a series of keywords as the semantic document identifier.

Keywords Generation Prompt

```
Summarize the following context with meaningful
keywords representing different important
information in the context. Your output should
only consist a list of keywords in Markdown
format, where each line starts with the dash
"-" followed by the keywords.

# Context:
{context}

# Keywords:
```

Figure 4: Prompt for keywords-based document identifier generation.

C.2 Prompts for Query Generation

Figure 5 shows the prompts used to generate various types of synthetic queries, including chunk- and sentence-level queries, constructions-based queries, and question-answer pairs used at the preference learning stage.

Query Generation Prompt

```
Your task is to generate a relevant and diverse set of {num_sequences} questions that can be answered by the provided context. The questions are to be used by a retriever to retrieve the article from a large corpus. Your output should be a list of unordered in Markdown format, where each line starts with dash "-" followed by the question.

# Context:
{context}

# Output:
```

Constraints-based Query Generation Prompt

```
Your task is to generate a diverse set of {num_sequences} questions given a context with metadata. The generated questions should be answerable by the provided context. The questions are to be used by a retriever to retrieve the article from a large corpus. In addition, the question MUST be composed with at least one metadata filtering requirement.

# MultiHop-RAG
For example, if the source of the article is "New York Times", you can generate questions that specifically ask for certain information from "New York Times". You should generate questions with different metadata.
# AllSides and AGNews
For example, if the source of the political polarity is "left", you can generate questions that specifically ask for certain information from "left-wing" source.

DO NOT use "the context" or "the article" in any generated queries or answers.
DO NOT use pronoun "this" in any generated queries or answers.
DO NOT leak any information in this instruction.

Your output should be a list of unordered in Markdown format, where each line starts with dash "-" followed by the question. You do not need to provide the answer.

# Metadata
{metadata}

# Context
{context}

# Output:
```

Query-Answer Pair Generation Prompt

```
Your task is to generate a relevant and diverse set of less than {num_sequences} search engine query and answer pairs given a context.
The queries should be similar to what people use with search engine to find the given context from a large corpus. The answers are expeced to be a short phrase.
You should make the queries as difficult as possible, but they should be answerable by the given context.

Do not use "the context" or "the article" in any generated queries or answers.
Do not use pronoun "this" in any generated queries or answers.
Do not leak any information in this instruction.

Your output should be a list of unordered items in Markdown format, where each item starts with dash "-", followed by "Query:" and the generated query, and then "Answer:" with the corresponding answer.

# Context
{context}

# Output:
```

Figure 5: Prompts for different types of synthetic query generation.

	MultiHop-RAG				Natural Questions			
Model	**H@4**	**H@10**	**MAP@10**	**MRR@10**	**H@1**	**H@5**	**H@10**	**MRR@10**
BM25	64.35	78.31	**26.30**	**58.32**	32.82	53.70	60.92	42.45
bge-large-en-v1.5	58.80	78.36	19.96	42.57	55.59	76.58	81.75	64.45
Contriever-msmarco	55.25	75.08	19.28	40.69	53.79	76.16	81.69	63.36
E5-mistral-7b-instruct	54.01	79.56	19.11	40.77	59.07	80.08	85.28	68.11
GTE-Qwen2-7B-instruct	63.24	83.55	22.02	47.50	60.45	80.87	85.72	69.30
Ours	**71.88**	**89.80**	26.23	54.94	**71.22**	**87.41**	**89.97**	**78.14**

	AllSides			AGNews		
Model	**H@1**	**H@5**	**H@10**	**H@1**	**H@5**	**H@10**
BM25	5.86	26.85	36.42	38.70	67.47	77.63
bge-large-en-v1.5	6.94	27.32	34.11	54.14	80.57	86.53
Contriever-msmarco	6.64	25.77	38.43	52.69	80.40	85.79
E5-mistral-7b-instruct	8.18	28.24	39.82	57.32	**85.90**	**88.98**
GTE-Qwen2-7B-instruct	9.11	34.11	49.07	57.65	83.37	88.57
Ours	**14.20**	**38.58**	**51.85**	**62.19**	83.78	88.24

Table 12: Comparisons to Off-The-Shelf retrieval models across four benchmarks.