GOLFer: Smaller LM-Generated Documents Hallucination Filter & Combiner for Query Expansion in Information Retrieval

Lingyuan Liu
City University of Hong Kong
ly.liu@my.cityu.edu.hk

Mengxiang Zhang*
The University of Hong Kong
mxzhang6@connect.hku.hk

Abstract

Large language models (LLMs)-based query expansion for information retrieval augments queries with generated hypothetical documents with LLMs. However, its performance relies heavily on the scale of the language models (LMs), necessitating larger, more advanced LLMs. This approach is costly, computationally intensive, and often has limited accessibility. To address these limitations, we introduce GOLFer - Smaller LMs-Generated Documents Hallucination Filter & Combiner - a novel method leveraging smaller open-source LMs for query expansion. GOLFer comprises two modules: a hallucination filter and a documents combiner. The former detects and removes non-factual and inconsistent sentences in generated documents, a common issue with smaller LMs, while the latter combines the filtered content with the query using a weight vector to balance their influence. We evaluate GOLFer alongside dominant LLM-based query expansion methods on three web search and ten low-resource datasets. Experimental results demonstrate that GOLFer consistently outperforms other methods using smaller LMs, and maintains competitive performance against methods using large-size LLMs, demonstrating its effectiveness. The code for our method is publicly available at https://github.com/liuliuyuan6/GOLFer.

1 Introduction

Information retrieval (IR) is crucial for extracting relevant information from large repositories, serving as a key component in modern search engines (Wang et al., 2019; Karpukhin et al., 2020). Query expansion, a key technique for enhancing IR performance, improves the precision and expressiveness of user queries (Azad and Deepak, 2019). Traditional methods use hand-built knowledge resources like WordNet and Thesaurus ((Pal et al., 2014; Gong et al., 2005) or external text collections (Roy et al., 2016; Diaz et al., 2016). However, these methods are limited by the quality of external data sources and show limited success on popular datasets (Azad and Deepak, 2019). More adaptable query expansion approaches are needed to meet diverse requirements across different contexts.

LLMs like GPT-4 (Ouyang et al., 2022) and LLaMA 3 (Dubey et al., 2024) have shown impressive abilities in generating fluent and realistic responses. Pre-trained on extensive corpora, these models excel in natural language understanding and generation. Ouyang et al. (2022) indicates that LLMs can be fine-tuned with minimal data to align with human intent, enabling them to generalize to diverse instructions in a zero-shot manner. This adaptability has spurred interest in using LLMs for query expansion in IR, where queries are often brief or ambiguous (Mitra and Craswell, 2017; Zhao et al., 2024). LLMs can generate hypothetical documents based on various prompts to enhance query expansion. For example, HyDE uses zero-shot instructions to generate a hypothetical document (Gao et al., 2022), and Query2Doc employs few-shot examples to create hypothetical-documents (Wang et al., 2023). These methods enhance the performance of retrievers such as Contriever and BM25 across a variety of tasks, including web search, question answering, and fact verification.

However, existing LLM-based query expansion methods face several critical challenges. Empirical experiments, such as those involving HyDE, indicate that the performance of LLM-based query expansion heavily depends on the scale of the LLM employed (Gao et al., 2022). Wang et al. (2023) suggest that smaller LMs tend to produce shorter outputs with more factual errors, posing a significant obstacle to building trustworthy systems. This tendency to hallucinate facts has been widely observed, where models can confidently generate fic-

*Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 153–162

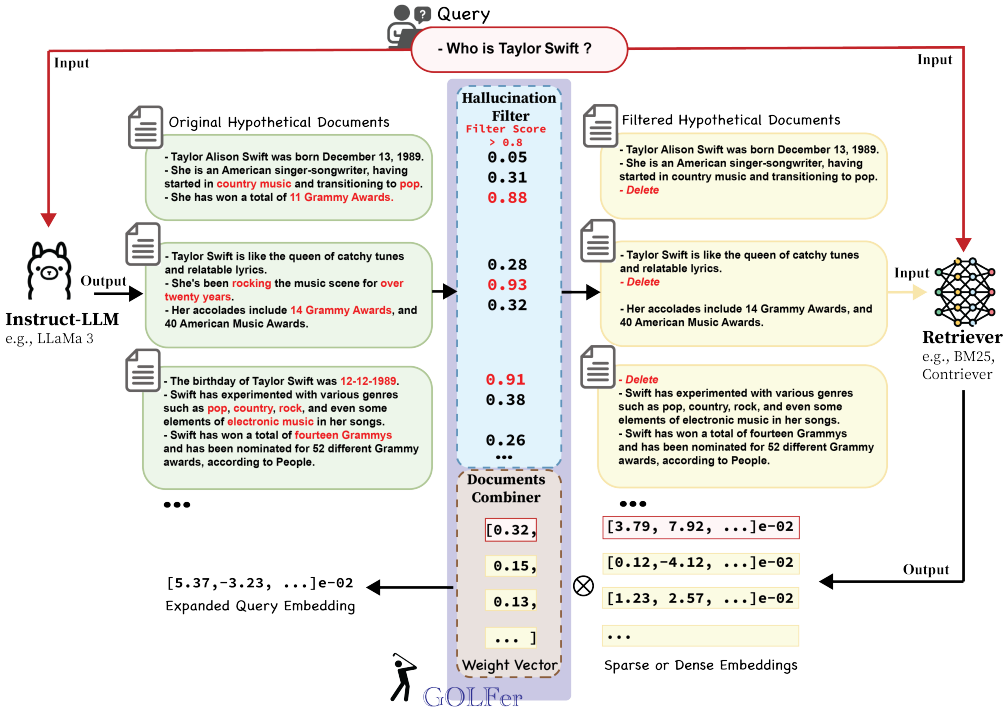

Figure 1: Overview of GOLFer. Given a query, GOLFer generates n passages using an Instruct-smaller LM, which are then processed through a hallucination filter to produce filtered hypothetical documents. These filtered documents are combined with the original query using a weight vector by the documents combiner module to create the expanded query embedding for retrieval.

titious information. Consequently, current LLM-based query expansion methods often advocate for using large-scale, advanced LLMs like GPT-3.5 (175B) and GPT-4 to mitigate these inaccuracies and enhance query expansion performance in IR. However, in practice, employing such large-scale models is costly, computationally intensive, and regionally restricted. For instance, the API costs for GPT-3.5 and GPT-4 are $1.50 and $2.50 per million tokens, respectively [*], making widespread use financially prohibitive for small and private institutions. Additionally, quota limits often restrict the practical application of these models, and their token-by-token autoregressive decoding can significantly reduce retrieval efficiency, potentially taking over 2000 ms to generate hypothetical documents (Wang et al., 2023). Furthermore, access to large-scale LLMs like GPT is restricted in certain regions, such as China, further limiting their applicability. These factors collectively constrain the practical deployment of existing LLM-based query expansion methods.

To overcome these limitations, we introduce a new method, GOLFer, i.e., smaller LMs-generated documents hallucination filter & combiner for query expansion in IR (See Fig. 1). Unlike existing LLM-based query expansion methods, our method focuses on smaller open-source LMs, such as LLaMA-3-8B-Instruct, to assist in query expansion, thereby mitigating some of the limitations associated with using large-scale models. However, smaller LMs often introduce hallucinations, such as non-factual content and inconsistent contexts, when generating hypothetical documents. If these flawed documents are directly utilized by retrievers, they can introduce irrelevant, noisy, or erroneous data, jeopardizing the quality of the expanded query embeddings and thus affecting query expansion performance in IR. To address this issue, GOLFer is designed to detect and filter out non-factual and inconsistent sentences from the hypothetical documents, ensuring that only relevant and accurate information is incorporated. Additionally, to balance the influence of the query and hypothetical documents, our method combines these filtered documents with the original query based on factuality by setting a weight vector. The inner product of the weight vector and the embedding vectors for the original query and filtered documents forms a new, refined query for IR. GOLFer is a lightweight, versatile query expansion method that can be integrated into any Transformer-based LM without

[*]Refer to https://openai.com/api/pricing/.

requiring additional training, fine-tuning, or prompt engineering.

We comprehensively evaluate GOLFer using LLaMA-3-8B-Instruct as the smaller LM, alongside three types of retrievers: sparse retriever (e.g., BM25), dense retriever (e.g., ANCE (Xiong et al., 2020)), and advanced dense retriever (e.g., Aggretriever_v1 (Lin et al., 2023)). Under consistent experimental conditions, we compare GOLFer with existing LLM-based query expansion methods, including HyDE (Gao et al., 2022) and query2doc (Wang et al., 2023), across three web search datasets from MS MARCO (Bajaj et al., 2016) and ten low-resource datasets from the BEIR benchmark (Thakur et al., 2021). The results demonstrate that GOLFer outperforms other methods using smaller LMs on all datasets across nearly all evaluation metrics, highlighting its effectiveness. Notably, GOLFer remains competitive even against methods utilizing large-scale LLMs like GPT-3.5 (175B). In summary, the contributions of our paper are as follows:

- We propose a novel smaller LMs-driven query expansion method: GOLFer. It can i) detect and filter out non-factual and inconsistent sentences in the original hypothetical documents generated by smaller LMs, and ii) combine these filtered hypothetical documents with the original query by a weight vector to form a expanded query for IR, enhancing factuality.
- We evaluate GOLFer alongside dominant query expansion methods on MS MARCO dev, TREC DL19 and DL20 and nine low-resource datasets form BEIR. Experimental results demonstrate the efficacy of our method.

2 Related Work

2.1 LLM-based Query Expansion

Query expansion improves retrieval systems by broadening query terms to include synonyms or related concepts, enhancing document matching. Leveraging LLMs' generative capabilities, some studies generate hypothetical documents for query expansion. For instance, HyDE uses an instruction-following LLM to create a hypothetical document (Gao et al., 2022), which is then encoded into an embedding vector for retrieval. Similarly, Query2Doc generates hypothetical documents through few-shot prompting of LLMs (Wang et al., 2023), combining them with the original query to boost retrieval performance. Additionally, various prompting strategies to generate hypothetical documents for query expansion, including zero-shot, few-shot, and Chain-of-Thought, have been explored (Jagerman et al., 2023), with Chain-of-Thought particularly effective. These advances suggest that knowledge distillation from LLMs can transfer their capabilities to smaller models, advancing LLM-based query expansion.

2.2 Hallucination Detection in LLM-Generated Documents

LLM-generated documents often suffer from hallucinations, producing nonsensical or inaccurate text (Raunak et al., 2021), which degrades system performance (Welleck et al., 2019). Recent research has focused on identifying these hallucinations through three primary approaches: white-box, grey-box, and black-box methods. White-box methods leverage LLM internal states to assess response factuality (Azaria and Mitchell, 2023), requiring labeled data for supervised training. Grey-box methods evaluate factuality using output distributions, employing intrinsic uncertainty metrics to identify uncertain segments (Yuan et al., 2021; Fu et al., 2023). Black-box methods, like SelfCheckGPT (Manakul et al., 2023), fact-check responses by comparing multiple sampled outputs for consistency. Each approach addresses hallucinations in different contexts, enhancing the reliability of LLM-generated content.

3 Methodology

In this section, we provide a detailed explanation of the GOLFer method. GOLFer is composed of two main components: the hallucination filter and the documents combiner, as shown in Fig. 1. Within the GOLFer method, n passages generated by an Instruct-smaller LLM from a given user query are treated as original hypothetical documents for query expansion. Initially, these documents are processed through the hallucination filter to produce filtered hypothetical documents. Subsequently, the expanded query embedding for query expansion is obtained by combining embedding vectors with a weight vector, which is determined by the documents combiner module in the GOLFer method. The configuration of the weight vector varies between sparse and dense retrieval methods. The hallucination filter is detailed in section 3.1, while the documents combiner is discussed in section 3.2.

Notation Suppose n passages for query expansion are generated by an instruction-following smaller LM, such as LLaMa-3-8B-Instruct, given a user query q. Let d^i refer to the i-th smaller LM-generated document, where $i \in \{1, 2, ..., n\}$. Each document d^i contains m_i sentences, denoted by s^i_j for the j-th sentence in document d^i, where $j \in \{1, 2, ..., m_i\}$. Furthermore, each sentence s^i_j consists of o^i_j tokens, with $t^i_j(l)$ representing the l-th token in the j-th sentence of the i-th smaller LM-generated document, where $l \in \{1, 2, ..., o^i_j\}$.

3.1 Hallucination Filter

The hallucination filter module can evaluate the degree of hallucination for each sentence in a smaller LM-generated passages based on consistency and factuality, and then, can filter these sentences out based on their hallucination degree.

Hallucination degree is based on the idea that factual sentences generated by a smaller LLM tend to contain tokens with higher likelihood and lower entropy, whereas hallucinated sentences are characterized by tokens with flat probability distributions and high uncertainty. What's more, each token has a different influence on the subsequent context. Thus, we define the factuality score of the hypothesis documents by evaluates the uncertainty of tokens and their impact on subsequent tokens. Our method begins by quantifying the uncertainty of each token, $t^i_j(l)$. This is achieved by recording the entropy of the token's probability distribution across the vocabulary. For any token $t^i_j(l)$, the entropy $\mathcal{H}^i_{jl}$ is computed as follows:

$$\mathcal{H}^i_{j_l} = -\sum_{\tilde{v} \in \mathcal{V}} p^i_{j_l}(\tilde{v}) \log p^i_{j_l}(\tilde{v}), \tag{1}$$

where $p^i_{j_l}(\tilde{v})$ denotes the probability of generating the token $\tilde{v}$ over all tokens in the vocabulary $\mathcal{V}$ at position l of the j-th sentence in document i.

In addition to uncertainty, GOLFer leverages the self-attention mechanism inherent in Transformer-based LLMs to assign weights to tokens, reflecting their impact on the subsequent context. Specifically, for any given token $t^i_j(l)$, we quantify its influence by recording the average attention value $Avg(\mathcal{A}^i_{jl})$, which captures the average attention from all following tokens. The attention scores are taken from the last Transformer layer of the smaller LM. The attention value $\mathcal{A}^i_{jl,v}$ between two tokens $t^i_j(l)$ and $t^i_j(v)$ for any $l < v$ is computed as follows:

$$\mathcal{A}^i_{j_l,v} = \text{softmax}\left(\frac{Q^i_{j_l} {K^i_{j_v}}^\top}{\sqrt{d_k}}\right), \tag{2}$$

where $Q^i_{j_l}$ represents the query vector of token $t^i_j(l)$, $K^i_{j_v}$ is the key vector of token $t^i_j(v)$, and d_k denotes the dimensionality of the key vector. The softmax function is applied to the dot product of $Q^i_{j_l}$ and $K^i_{j_v}$, normalized by the square root of d_k. The average attention value $Avg(\mathcal{A}^i_{jl})$ for token $t^i_j(l)$ is then identified by averaging $\mathcal{A}^i_{jl,v}$ for all $v > l$:

$$Avg(\mathcal{A}^i_{jl}) = \frac{\sum_{v=l+1}^{o^i_j} \mathcal{A}^i_{jl,v}}{o^i_j - l}. \tag{3}$$

Combining uncertainty and significance, GOLFer computes a comprehensive factuality score for each token $t^i_j(l)$. Specifically, the factuality score $\mathscr{F}^i_{jl}$ is calculated by multiplying the entropy $\mathcal{H}^i_{jl}$ of the token by its average attention value $Avg(\mathcal{A}^i_{jl})$:

$$\mathscr{F}^i_{jl} = \mathcal{H}^i_{jl} \cdot Avg(\mathcal{A}^i_{jl}). \tag{4}$$

This token-level factuality score serves as the basis for evaluating the overall factuality of a sentence. The sentence-level factuality score $\mathscr{F}(s^i_j)$ is then derived by averaging the factuality scores of all tokens within the sentence:

$$\mathscr{F}(s^i_j) = \frac{\sum_{l=1}^{o^i_j} \mathscr{F}^i_{jl}}{o^i_j}. \tag{5}$$

And the fundamental idea behind detecting hallucination in terms of consistency is rooted in the premise that if a smaller LLM possesses genuine knowledge of a concept, its sampled responses will likely be similar and factually consistent. Conversely, hallucinated facts often lead to divergent and contradictory responses when multiple outputs are drawn from the same query. By comparing multiple responses generated from the same query, we can assess information consistency and determine the factuality of statements (Manakul et al., 2023). Natural Language Inference (NLI) has been utilized to measure faithfulness in hallucination detection (Manakul et al., 2023), demonstrating competitive performance. Inspired by this approach, we employ a fine-tuned NLI classifier, DeBERTa-v3-large (He et al., 2021), to compute the NLI contradiction score for each sentence across different documents. Only the logits associated with the entailment and contradiction classes are considered,

and the NLI contradiction score $\mathscr{C}$ for the sentence s_j^i in document $d^{k \neq i}$ is computed as follows:

$$\mathscr{C}(s_j^i, d^{k \neq i}) = \frac{\exp(\omega_c)}{\exp(\omega_c) + \exp(\omega_e)}, \quad (6)$$

where $\exp(\omega_c)$ and $\exp(\omega_e)$ are the logits of the contradiction and entailment classes, respectively. In GOLFer, the consistency score for a sentence s_j^i is the mean value of its NLI contradiction scores across all documents that do not contain s_j^i. This can be formulated as:

$$\mathscr{C}(s_j^i) = \frac{\sum_{k \in \{1,2,\ldots,n\} \setminus \{i\}} \mathscr{C}(s_j^i, d^k)}{n-1}. \quad (7)$$

We calculate the filter score $\mathscr{H}(s_j^i)$ as follows:

$$\mathscr{H}(s_j^i) = \mathscr{F}(s_j^i) \cdot \mathscr{C}(s_j^i). \quad (8)$$

If the filter score $\mathscr{H}(s_j^i)$ is beyond than a certain number, we will delete it since it could be highly hallucinatory. We have empirically found that 0.8 is a generally good value and do not tune it on a dataset basis.

3.2 Documents Combiner

The documents combiner module can merge the original query with filtered hypothetical documents to form an expanded query for IR. This combination process is tailored based on the type of retrieval—sparse or dense—and the generation confidence of the documents. The subsequent sections elaborate on the specific operations for sparse and dense retrievers, respectively.

Sparse Retrieval In the case of sparse retrieval, we enhance the query term weights by repeating the query 20 times when combining 5 hypothesis documents. This repetition aims to balance the relative weights of the query and the hypothetical documents before merging them. The expanded query q^+ for sparse retrieval is then formulated as follows:

$$q^+ = 20 \cdot q + \sum_{i=1}^{n} d^i, \quad (9)$$

where $n = 5$. This formulation ensures an effective balance between the original query and the augmented content for improved retrieval performance.

Dense Retrieval For dense retrieval, the document combiner module takes into account the generation confidence of smaller LM-generated hypothetical documents. These documents, with varying degrees of generation confidence, exhibit different levels of factual information and relevance patterns concerning the real documents we aim to retrieve. Consequently, we posit that hypothetical documents with higher generation confidence should contribute more significantly to the query expansion process for IR.

Specifically, we estimate the generation confidence, $\varpi(d^i)$ of a filtered document by averaging the generation probabilities, p_{jl}^i, of each token within the filtered documents. This can be expressed as follows:

$$\varpi(d^i) = \frac{\sum_{j=1}^{m_i} \sum_{l=1}^{o_j^i} p_{jl}^i}{\sum_{j=1}^{m_i} o_j^i}. \quad (10)$$

Subsequently, we encode both the generated documents and the original query into embedding vectors using the dense retriever, denoted as $f(\cdot)$. The expanded query embedding for IR, V_{q^+}, is then formulated by combining the embedding vectors for the filtered hypothetical documents and the original query with their corresponding weights. This can be expressed as follows:

$$V_{q^+} = \beta \cdot f(q) + \frac{1-\beta}{\sum_{i=1}^{n} \varpi(d^i)} \cdot \sum_{i=1}^{n} \varpi(d^i) f(d^i), \quad (11)$$

where β represents the contribution rate of the original query in forming the expanded query for IR. And we find that $\beta = 0.6$ is an effective values for combing 5 hypothesis documents , which we do not tune on a dataset-specific basis. For dense retrieval, the inner product is computed between V_{q+} and the set of all document vectors, and the most similar documents are subsequently retrieved.

4 Experiments

4.1 Setup

Implementation We implement Meta-LLaMA-3-8B-Instruct (Dubey et al., 2024), a smaller open-source LM, to generate hypothetical documents for given queries. We sample documents with a temperature setting of 0.6, top-p of 0.9 and max tokens of 128 for open-ended generation. Retrieval experiments are conducted using the Pyserini toolkit (Lin et al., 2021).

Datasets and Evaluations We evaluate our method using two types of datasets relevant to information retrieval tasks. The first type includes web search datasets: MS MARCO dev (Bajaj et al., 2016), TREC-DL-2019 (Craswell et al.,

Model	Fine-tuning	MS MARCO dev		TREC DL 19			TREC DL 20		
		MRR@10	R@1k	MAP	nDCG@10	R@1K	MAP	nDCG@10	R@1K
Sparse retrieval									
BM25 (Robertson et al., 2009)	×	18.4	85.7	30.1	50.6	75.0	28.6	48.0	78.6
+query2doc (Wang et al., 2023)	×	19.0	86.9	36.3	53.9	78.7	38.4	56.1	83.6
+GOLFer	×	**19.9**$^{+1.5}$	**88.5**$^{+2.8}$	**39.1**$^{+9.0}$	**59.5** $^{+9.0}$	**83.0**$^{+7.9}$	**45.5**$^{+17.0}$	**62.9**$^{+14.9}$	**86.1**$^{+7.4}$
Dense retrieval w/o distillation									
ANCE (Xiong et al., 2020)	√	33.0	95.9	37.1	64.5	75.5	40.8	64.6	77.6
+HyDE (Gao et al., 2022)	√	33.2	96.3	**45.3**	68.2	**80.5**	44.2	67.8	81.6
+query2doc (Wang et al., 2023)	√	32.9	96.0	45.0	69.9	77.8	44.6	67.5	**82.0**
+GOLFer	√	**33.3**$^{+0.3}$	**96.4**$^{+0.5}$	44.1$^{+7.0}$	**71.3**$^{+6.8}$	79.1$^{+3.6}$	**44.7**$^{+3.9}$	**67.9**$^{+3.3}$	81.2$^{+3.6}$
Dense retrieval w/ distillation									
Colbert_v2 (Santhanam et al., 2021)	√	34.4	96.7	41.0	68.4	81.3	45.2	69.3	83.9
+HyDE (Gao et al., 2022)	√	34.1	96.7	47.5	72.0	83.7	46.8	69.5	84.4
+query2doc (Wang et al., 2023)	√	33.9	96.6	46.0	71.3	81.3	47.4	69.8	85.6
+GOLFer	√	34.5$^{+0.1}$	97.1$^{+0.4}$	48.2$^{+7.3}$	73.4$^{+4.9}$	84.6$^{+3.3}$	47.4$^{+2.2}$	70.0$^{+0.7}$	85.7$^{+1.8}$
Aggretriever_v1 (Lin et al., 2023)	√	34.1	96.0	43.0	68.2	80.2	43.3	67.3	83.5
+HyDE (Gao et al., 2022)	√	33.8	96.3	47.3	68.9	85.7	44.8	68.0	84.4
+query2doc (Wang et al., 2023)	√	34.0	96.2	47.5	69.5	81.9	45.5	68.1	87.0
+GOLFer	√	34.3$^{+0.2}$	96.9$^{+0.8}$	48.3$^{+5.3}$	70.3$^{+2.1}$	85.3$^{+5.1}$	45.3$^{+2.0}$	67.9$^{+0.7}$	86.6$^{+3.1}$
Aggretriever_v2 (Lin et al., 2023)	√	36.2	97.4	43.5	68.4	80.8	47.1	69.7	85.6
+HyDE (Gao et al., 2022)	√	36.1	97.5	48.3	**73.5**	85.4	**49.2**	72.0	86.6
+query2doc (Wang et al., 2023)	√	36.1	97.4	46.4	72.1	81.8	47.5	71.1	88.0
+GOLFer	√	**36.3**$^{+0.1}$	**97.8**$^{+0.4}$	**48.4**$^{+4.9}$	73.0$^{+4.7}$	**86.4**$^{+5.6}$	49.0$^{+1.9}$	**72.2**$^{+2.5}$	**88.3**$^{+2.8}$

Table 1: Results for web search on MS MARCO dev and DL19/20. Best performing systems are marked **bold**. All the hypothetical documents used in this table are generated by LLaMA-3-8B-Instruct.

Dataset	BM25	BM25+G.	BM25+Q.	Cont.	Cont.+G.	Cont.+H.	Cont.+Q.
	nDCG@10						
NQ	30.5	**47.6**	42.3	49.8	**51.2**	51.1	50.8
FiQA-2018	23.6	**23.9**	23.6	24.5	**26.5**	24.5	21.3
TREC-COVID	59.5	**69.9**	72.1	27.1	**57.4**	53.1	51.5
Signal-1M	33.0	**36.5**	34.7	27.8	**29.9**	29.1	29.4
TREC-NEWS	39.5	**50.5**	49.1	34.8	**41.1**	40.1	38.7
Robust04	40.7	**46.8**	43.1	47.3	**47.7**	47.5	47.4
Touche 2020	44.2	**45.8**	45.3	20.4	21.1	20.7	**21.8**
CQADupStack	30.2	**31.4**	30.0	**34.5**	34.4	34.2	33.9
DBPedia	31.3	**34.1**	32.8	41.3	**46.5**	42.3	43.2
SciFact	67.9	**70.8**	70.3	67.7	**69.4**	67.9	66.2
	Recall@100						
NQ	76.0	**89.7**	85.3	82.1	**83.1**	82.9	82.8
FiQA-2018	53.9	**56.9**	56.0	56.2	**61.8**	59.3	56.8
TREC-COVID	49.8	**56.7**	51.5	17.2	**32.0**	30.4	30.4
Signal-1M	37.0	**39.8**	38.1	32.2	**34.4**	33.6	33.1
TREC-NEWS	44.7	**52.8**	51.5	42.3	**49.7**	46.1	43.2
Robust04	37.5	**38.3**	37.7	39.2	**42.6**	40.3	41.1
Touche 2020	53.8	56.7	**56.8**	44.2	**46.0**	44.4	45.8
CQADupStack	60.6	**61.7**	60.5	**66.3**	59.7	59.1	58.7
DBPedia	39.8	**47.0**	42.1	54.1	**58.1**	46.2	45.1
SciFact	92.5	**95.4**	95.1	92.6	**96.6**	96.1	94.1

Table 2: Results for Low resource tasks from BEIR. Best performing systems are marked **bold**. G. represents GOLFer, Q. represents query2doc, and H. represents HyDE, and Cont. represents Contriever that are fine-tuned on MS MARCO training data. All the hypothetical documents used in this table are generated by LLaMA-3-8B-Instruct.

2020), and 2020 (Craswell et al., 2021). The second type consists of low-resource datasets from the BEIR benchmark (Thakur et al., 2021), such as NQ, FiQA-2018, TREC-COVID, Signal-1M, TREC-NEWS, Robust04, Touche2020, CQADupStack, DBPedia, and Scifact. We use the following evaluation metrics: MAP, $nDCG@10$, and $Recall@1k$ for TREC DL 2019 and 2020, $MRR@10$ and $Recall@1k$ for MS-MARCO datasets, and $nDCG@10$ and $Recall@100$ for the BEIR datasets. We employ distinct instructions for each dataset, maintaining a similar structure but varying quantifiers to control the form of the generated hypothetical documents. These instructions are detailed in Appendix A.1.

Compared Systems In our experiments, for the web search task, we use BM25 (Robertson et al., 2009) as the baseline for sparse retrieval and ANCE (Xiong et al., 2020), fine-tuned on MS-MARCO datasets, as the baseline for dense retrieval. Additionally, we consider three advanced dense retrievers enhanced by distillation and pre-training techniques: Colbert_v2 (Santhanam et al., 2021), Aggretriever_v1 trained with distillation (Lin et al., 2023), and Aggretriever_v2 trained with distillation and pre-training (Gao and Callan, 2021). For the low-resource retrieval task, BM25 is again used as the baseline for sparse retrieval, while Contriever (Izacard et al., 2021) serves as the baseline for dense retrieval. Retrievers within GOLFer share the same embedding spaces as these baselines, with

the primary difference being in how the query vector is constructed.

This setup allows us to effectively assess the impact of GOLFer. Furthermore, we compare our method with two other query expansion approaches: HyDE, designed for dense retrieval systems, and query2doc, applicable to both sparse and dense retrieval systems. HyDE and query2doc use the same original hypothesis documents as we do.

4.2 Web Search

The results, summarized in Table 1, present the performances of various retrieval models enhanced by GOLFer. For sparse retrieval, GOLFer consistently outperforms the query2doc approach across all metrics, demonstrating its superior effectiveness in improving retrieval performance. In dense retrieval without distillation, ANCE combined with GOLFer shows significant improvements across most metrics compared to other methods. For dense retrieval with distillation, GOLFer enhances the performance of Colbert_v2, Aggretriever_v1 and v2, with notable gains in metrics such as $nDCG@10$ and $R@1k$. This consistent improvement across both sparse and dense retrieval models highlights the robustness and reliability of GOLFer in enhancing retrieval systems.

4.3 Low Resource Retrieval

The performance for low-resource tasks is summarized in Table 2. For sparse retrieval using BM25, GOLFer generally outperforms query2doc across nearly all datasets. Notably, GOLFer improves $nDCG@10$ and $Recall@100$ significantly on datasets such as NQ, TREC-COVID, and TREC-NEWS. The only exception is a slight underperformance in $Recall@100$ on the Touche2020 dataset, where the difference is minimal (0.4 difference). This consistent performance highlights the robustness of GOLFer in enhancing sparse retrieval tasks. For dense retrieval using Contriever, GOLFer consistently surpasses other query expansion approaches across all low-resource datasets and metrics. Specifically, it shows substantial improvements in $nDCG@10$ and $Recall@100$ on datasets like NQ, FiQA-2018, and DBPedia. These results demonstrate the effectiveness of GOLFer in enhancing query expansion performance, significantly contributing to the improvement of retrieval tasks.

5 Analysis

Ablation Study To better understand the utility of GOLFer, we use Aggretriever_v2 as a backbone model to conduct various experiments on the TREC DL 19/20 datasets, analyzing the impact and effectiveness of each component within this architecture as follows:

Necessity of Individual Components: We establish two variants to investigate the necessity of each component: a) **w/ Filter Only**: Our proposed framework with only the hallucination filter module. b) **w/ Combiner Only**: Our proposed framework with only the document combiner module.

Model	TREC DL 19		
	MAP	nDCG@10	R@1K
Aggretriever_coCondenser	43.5	68.4	80.8
w/ filter only	47.3	68.6	86.0
w/ combiner only	47.9	72.1	85.9
w/ filter + combiner	48.4	73.0	86.4
	TREC DL 20		
Aggretriever_coCondenser	47.1	69.7	85.6
w/ filter only	45.7	64.6	87.4
w/ combiner only	48.9	71.9	87.9
w/ filter + combiner	49.0	72.2	88.3

Table 3: Ablation results of GOLFer on TREC DL 19/20

From Tabs 3, the following conclusions can be drawn: a) The performance of GOLFer on the TREC DL 19/20 datasets surpasses these variants lacking components, affirming the effectiveness and necessity of both the hallucination filter module and the document combiner module. The hallucination filter module reduces the degree of hallucination in smaller LM-generated passages, while the document combiner module balances the influence of the original query and the hypothetical document. These modules function independently yet complement each other, amplifying the performance of smaller LM-based query expansion. b) Among the different variants, the variant w/ Combiner Only shows high performance, highlighting the critical role of balancing the influence of the original query and the hypothetical document in enhancing query expansion. By further incorporating the hallucination filter module, irrelevant or erroneous information generated by smaller LMs is reduced, thus enhancing the overall performance of the GOLFer framework.

Compare to Large size Generative Models In this experiment, we explore the potential of GOLFer using a smaller LM by comparing it with existing dominant query expansion methods with LLMs. Previous studies have shown that the scale

	GPT-4o	LLaMA-3 (8B) w/ Contriever
	w/ HyDE	w/ GOLFer
Scifact	69.2	**69.4**
TREC-NEWS	**44**	41.1
FiQA	27.6	**28.1**
DBPedia	**37.1**	35.7

Table 4: Results for effect of different combination of instruction LMs and query expansion approaches. Hypothesis documents for GOLFer are generated using LLaMA-3-8B-Instruct, while those for Hyde are generated by GPT-4o. Best systems are marked **bold.**

of the generative LLM significantly impacts the quality of query expansion (Wang et al., 2023; Gao et al., 2022). We compared our performance to HyDE in BEIR datasets. It is important to note that the hypothesis documents for GOLFer are generated using LLaMA-3-8B-Instruct, while those for Hyde are generated by GPT-4o.

As shown in Table 4, GOLFer with LLaMA-3-8B outperforms HyDE with GPT-4o on the SciFact and FiQA datasets in terms of nDCG@10, although it falls behind on the TREC-NEWS and DBPedia datasets. Those results show that GOLFer with smaller LMs is competitive with, and sometimes outperforms, other query expansion methods with LLMs across various low-resource retrieval tasks. GOLFer is potential as a viable alternative to LLM-based query expansion methods in information retrieval.

Generalizability We applied GOLFer to another LM, Deepseek-r1-distill-qwen-7b, across DL19/20 datasets. The results of DL19/DL20 are shown in the table, proving the generalizability of GOLfer.

6 Conclusion

In this work, we introduce GOLFer, a novel method designed to leverage smaller open-source LMs for query expansion, aiming to enhance both sparse and dense retrieval systems. The core idea is to distill the smaller LM outputs through effective hallucination detection and mitigation techniques. GOLFer identifies and filters out non-factual and inconsistent sentences in smaller LM-Generated documents, ensuring that only reliable documents are used as hypothetical documents for query expansion. The expanded query embeddings for information retrieval are then obtained by computing the dot product of the embedding vectors of the filtered hypothetical documents and the original query with a weight vector. Experimental evaluations demonstrate that the effectiveness of GOLFer in filtering and combining smaller LM-Generated texts contributes significantly to the improvement of query expansion performance in information retrieval.

7 Limitations

We acknowledge several limitations in this paper. One significant limitation is the dependency on the self-attention mechanism of Transformer-based LLMs for evaluating factuality scores within the hallucination detection module. Although self-attention scores are available for all open-source LLMs, our method cannot be applied directly to certain APIs that do not offer access to these scores. Consequently, our future work will focus on developing alternative approaches to address this limitation.

References

Hiteshwar Kumar Azad and Akshay Deepak. 2019. Query expansion techniques for information retrieval: a survey. *Information Processing & Management*, 56(5):1698–1735.

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when it's lying. *arXiv preprint arXiv:2304.13734*.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. *Preprint*, arXiv:2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*.

Zhiguo Gong, Chan Wa Cheang, and U Leong Hou. 2005. Web query expansion by wordnet. In *Database and Expert Systems Applications: 16th International Conference, DEXA 2005, Copenhagen, Denmark, August 22-26, 2005. Proceedings 16*, pages 166–175. Springer.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2(3).

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

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. *arXiv preprint arXiv:2004.04906*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2356–2362.

Sheng-Chieh Lin, Minghan Li, and Jimmy Lin. 2023. Aggretriever: A simple approach to aggregate textual representations for robust dense passage retrieval. *Transactions of the Association for Computational Linguistics*, 11:436–452.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Dipasree Pal, Mandar Mitra, and Kalyankumar Datta. 2014. Improving query expansion using wordnet. *Journal of the Association for Information Science and Technology*, 65(12):2469–2478.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. *arXiv preprint arXiv:2104.06683*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *CoRR*, abs/2112.01488.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Heyuan Wang, Ziyi Wu, and Junyu Chen. 2019. Multi-turn response selection in retrieval-based chatbots with iterated attentive convolution matching network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1081–1090.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4):1–60.

A Appendix

A.1 Instructions

TREC DL19 Instruction message = *"Please write a passage to answer the question. [question_text]"*.

TREC DL20 Instruction message = *"Please write a passage to answer the question. [question_text]"*.

MS MARCO dev Instruction message = *"Please write a passage to answer the question. [question_text]"*.

NQ Instruction message = *"Please write a passage to answer the question. [question_text]"*.

FiQA-2018 Instruction message = *"Please write a financial article passage to answer the question. [question_text]"*.

TREC_COVID Instruction message = *"Please write a scientific paper passage to answer the question. [question_text]"*.

Signal-1m Instruction message = *"Please write a passage to answer the question. [question_text]"*.

TREC_NEWS Instruction message = *"Please write a news passage about the topic. [question_text]"*.

Robsut04 Instruction message = *"Please write a news passage about the topic. [question_text]"*.

Touche2020 Instruction message = *"Please write a counter argument for the passage. [question_text]"*.

CQADupStack Instruction message = *"Please write a passage to answer the question. [question_text]"*.

DBPedia Instruction message = *"Please write a passage to answer the question. [question_text]"*.

SciFact Instruction message = *"Please write a scientific paper passage to support/refute the claim. [question_text]"*.