Optimizing Information Retrieval in RAG through Intelligent Reranking and Follow-Up Query Predictions

Kok Rhui Ong[1*] and Dr Wai Peng Wong[1†]

[1*]School of IT, Monash University Malaysia, Jalan Lagoon Selatan, Subang Jaya, 47500, Selangor, Malaysia.

*Corresponding author(s). E-mail(s): kokrhui.ong@gmail.com; Contributing authors: waipeng.wong@monash.edu;
†These authors contributed equally to this work.

Abstract

Reranking is a crucial process in Retrieval-Augmented Generation (RAG) systems as it significantly impacts the quality and the relevance of retrieved knowledge chunks. Conventional reranking models usually prioritize semantic similarity and matching accuracy between user queries and knowledge base embeddings. This causes them to often lack the ability to dynamically adapt to the evolving context of user interactions. In this paper, we propose a novel reranking framework designed to fill this gap and enhance retrieval in RAG systems by incorporating LLM-generated predicted follow-up queries coupled with the initial user query to better capture the evolving user intent. Our model leverages a fine-tuned weighting mechanism to balance the embeddings of the initial query and predicted follow-up queries, enabling context-aware reranking of knowledge chunks. The proposed approach tackles critical challenges, including enhancing personalization, scalability and ensuring relevance in scenarios where user queries are dynamic and context dependent. It addresses the need for adaptive retrieval mechanisms that can effectively handle evolving user intent and context to improve the quality of retrieved information. Evaluation on two benchmark datasets demonstrates that our reranking framework improves retrieval quality, effectively integrating user intent prediction to optimize the RAG process. Our results highlight the potential of embedding-driven, adaptive reranking models to advance the capabilities of RAG systems and pave the way for more intelligent information retrieval applications.

Keywords: Reranking, Information Retrieval, Retrieval Augmented Generation (RAG), Large Language Models (LLMs), Natural Language Processing (NLP)

1 Introduction

Reranking is a fundamental component of Retrieval-Augmented Generation (RAG) systems, playing a pivotal role in refining the relevance of knowledge chunks retrieved for a given query [1–3]. In traditional RAG pipelines, the initial ranking of retrieved knowledge chunks is based solely on the embedding similarity between the user query and the knowledge base, as illustrated in Figure 1. While this approach is effective in producing relevant results, it often overlooks the dynamic nature of user intent and the potential for context-aware enhancements [4, 5]. Addressing these limitations is critical to improving the adaptability and efficacy of RAG systems in real-world applications.

RAG systems face several critical challenges that underscore the need for effective reranking. One key challenge is the evolving nature of user intent, which can shift across interactions, particularly in information-intensive applications like digital libraries, scientific databases, and telemedicine platforms. For example, in a telemedicine application, an initial query about symptoms might lead to follow-up questions about potential treatments or medication interactions. Without reranking mechanisms that account for these evolving queries, the system might retrieve only general information about symptoms, neglecting the user's actual need for actionable treatment insights. This challenge highlights the necessity of reranking to ensure that retrieved knowledge remains contextually relevant across different stages of interaction.

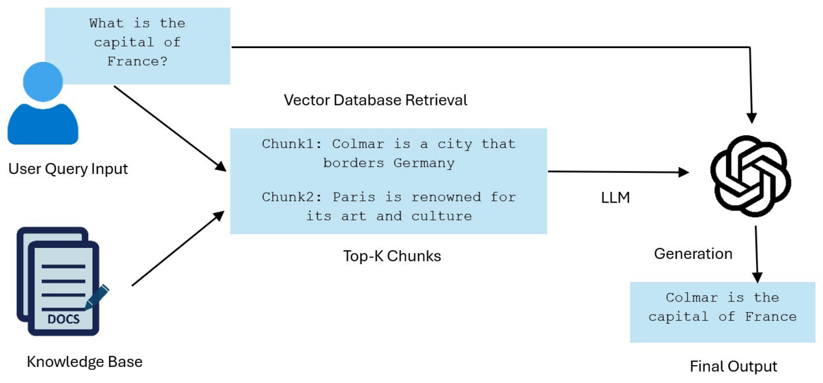
What is the capital of France?
User Query Input
Vector Database Retrieval
Chunk1: Colmar is a city that borders Germany
Chunk2: Paris is renowned for its art and culture
Top-K Chunks
LLM
Generation
DOCS
Knowledge Base
Colmar is the capital of France
Final Output

Fig. 1 A diagram of the basic RAG (Retrieval-Augmented Generation) pipeline without reranking of knowledge chunks.

Existing reranking approaches in RAG primarily focus on optimizing retrieval accuracy [6, 7]. However, this narrow focus fails to account for evolving user queries and nuanced contexts, limiting the potential for personalization and scalability [8–11]. In many cases, users' intent evolves as they interact with a system, resulting in follow-up queries that provide additional clarity or depth to their initial inquiry. Current reranking models are often unable to leverage this evolving context effectively, leading to suboptimal retrieval performance.

To address these limitations, we propose a novel reranking framework that enhances the retrieval process in RAG by incorporating predicted follow-up queries alongside the initial user query. Our approach leverages a fine-tuned weighting mechanism to balance the embeddings of the initial query and predicted follow-up queries, enabling a context-aware reranking of knowledge chunks. This framework not only improves retrieval accuracy but also enhances personalization and scalability, making it well-suited for dynamic user scenarios across diverse applications, including telemedicine, digital libraries, and research databases.

Our contributions in this work are threefold:

- We introduce a reranking model that integrates predicted follow-up queries to dynamically refine the retrieval process in RAG systems.
- We propose a fine-tuned weighting mechanism that balances embeddings from the initial and follow-up queries, enabling context-aware reranking.
- We evaluate our approach on two benchmark datasets, demonstrating significant improvements in retrieval quality compared to existing reranking methods.

In the following sections, we detail the architecture and methodology of our reranking framework, present experimental results, and discuss the implications of our findings for the development of next-generation RAG systems.

2 Related Work

Query Expansion and Fusion: Early research in query expansion has explored the integration of large language models (LLMs) to generate additional queries and enhance retrieval performance. The Generation-Augmented Retrieval (GAR) framework exemplifies this approach by leveraging pre-trained models to generate relevant contexts—such as potential answers and titles—that enrich the original query representation [12, 13]. Similarly, Generation-Augmented Query Expansion (GAQE) has been applied in code retrieval, where pre-trained models generate code snippets to improve retrieval quality [14]. Further refinements in retrieval-augmented generation (RAG) systems have introduced query rewriting techniques, query augmentation, decomposition, and disambiguation to improve query effectiveness [15, 16]. Nonetheless, standardized evaluation criteria for these techniques in knowledge retrieval systems remain underexplored.

Reranking Mechanisms in Retrieval-Augmented Generation: Reranking strategies have been extensively studied to optimize retrieval quality by refining the ranking of retrieved documents. A notable advancement in this area is the introduction of a list-aware reranking-truncation joint model, which concurrently performs reranking and document truncation to enhance retrieval for web search and RAG systems [17]. Moreover, a new semantic ranker has been integrated with generative query rewriting techniques, setting new standards in retrieval efficiency and relevance across multiple benchmarks [18]. Graph-based rerankers have also been proposed to connect related documents that might not be semantically strong [19]. While the idea of ad-hoc retrieval and reranking has been well established [20, 21], there is still room for improvement in how these methods integrate dynamic contextual cues.

LLM-Powered Query Refinement and User Interaction: LLMs have also been applied to refine user queries dynamically by analyzing conversational search logs [10, 22, 23]. One study introduced an LLM-powered classifier that identifies follow-up query patterns, thereby improving the understanding of user intent and informing reranking processes [24]. Additionally, LLMs have been directly employed for zero-shot document reranking [25, 26] besides query expansion [27, 28]. Although these approaches collectively push the boundaries of knowledge retrieval, further research is needed to seamlessly integrate query reformulation, expansion, and ranking into unified systems that address nuanced user needs.

Table 1 Comparison of This Work with Related Research

Ref.	**Primary Method**	**Reranking Score Inputs**			**Enhancement Strategy**
		Initial Query Sim.	Predicted Follow-ups	Geometric Penalty	
[12, 13]	Query Expansion	✓			Generated Answers/Titles
[14]	Query Expansion	✓			Generated Code Snippets
[15]	Query Rewriting	✓			Query Decomposition
[17]	Reranking	✓			Joint Truncation
[24]	Intent Analysis	✓			Historical Log Analysis
[19]	Reranking	✓			Graph-based Relations
This Work	**Reranking**	✓	✓	✓	**Predicted Follow-ups**

While the aforementioned studies have significantly advanced query and document handling in retrieval systems, our work distinguishes itself by introducing a novel reranking mechanism that proactively anticipates user intent. To better contextualize our contributions, Table 1 provides a comparative analysis of our framework against key related works.

The primary distinction of our approach lies in the reranking strategy. Unlike methods focused on pre-retrieval query enhancement, such as GAR [12, 13] and GAQE [14], which augment the initial query before retrieval, our framework operates post-retrieval. It refines from an already retrieved set of candidate chunks.

Furthermore, our method differs from other reranking and query analysis techniques. While some research analyzes historical conversational logs to understand follow-up patterns [24], our system dynamically predicts potential follow-up queries in real-time for the current query, making it adaptive to the immediate context rather than reliant on past data. Other rerankers may leverage graph-based relationships [19] or joint truncation models [17], but our model introduces a unique and comprehensive scoring function. Specifically, our reranker is novel in its synthesis of three distinct signals: (1) semantic similarity to the initial query, (2) aggregated semantic similarity to a set of predicted follow-up queries, and (3) a geometric penalty based on the Euclidean distance in the embedding space. This combination allows our framework to capture not only the explicit query but also the anticipated trajectory of a user's information needs, offering a more comprehensive and forward-looking approach to relevance ranking in RAG systems.

3 Framework

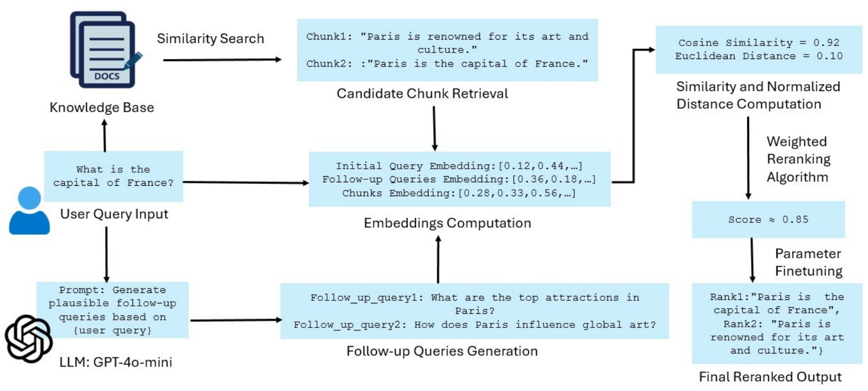
Similarity Search
Chunk1: "Paris is renowned for its art and culture."
Chunk2: :"Paris is the capital of France."
Knowledge Base
Candidate Chunk Retrieval
Cosine Similarity = 0.92
Euclidean Distance = 0.10
Similarity and Normalized Distance Computation
What is the capital of France?
User Query Input
Initial Query Embedding:[0.12,0.44,…]
Follow-up Queries Embedding:[0.36,0.18,…]
Chunks Embedding:[0.28,0.33,0.56,…]
Embeddings Computation
Weighted Reranking Algorithm
Score ≈ 0.85
Parameter Finetuning
Prompt: Generate plausible follow-up queries based on {user query}
LLM: GPT-4o-mini
Follow_up_query1: What are the top attractions in Paris?
Follow_up_query2: How does Paris influence global art?
Follow-up Queries Generation
Rank1:"Paris is the capital of France", Rank2: "Paris is renowned for its art and culture.")
Final Reranked Output

Fig. 2 A diagram of the proposed RAG (Retrieval-Augmented Generation) pipeline. A user query is enhanced by an LLM to generate follow-up queries. Embeddings are computed for all queries and used to retrieve candidate chunks from a knowledge base, which are then reranked using a weighted algorithm to produce the final output.

In the RAG process, the quality of retrieval is crucial for generating accurate and contextually relevant responses. The overall architecture of our proposed reranking framework is shown in Figure 2. This section details the methodology used for developing a reranker model to enhance the retrieval stage in Retrieval-Augmented Generation (RAG) processes. The framework introduces a novel approach to integrating user queries, predicted follow-up queries, and weighted reranking algorithms, aiming to optimize knowledge retrieval performance. This proposes a Retrieval-Augmented Generation (RAG) approach to improve the relevance and accuracy of retrieved documents in response to user queries.

3.1 Problem Formulation

Let Q represent a set of user queries and KB denote a knowledge base which contains chunks of content for retrieval. Given an initial query $q_{init} \in Q$, the model retrieves a candidate list of knowledge chunks:

$$KB_{cand} = \{c_1, c_2, ..., c_N\} \tag{1}$$

To enhance the relevance of the retrieved chunks, our reranking framework assigns a refined score to each chunk based on its alignment with the initial query and predicted follow-up queries $Q_{follow} = q_1, q_2, \ldots, q_M$. The reranked list KB_{ranked} is produced as:

$$KB_{ranked} = TopK_{c \in KB_{cand}} R(c, q_{init}, Q_{follow}) \tag{2}$$

where R is the reranking function that evaluates the relevance of each chunk c by leveraging embedding-based similarity measures, fine-tuned weights, and scoring mechanisms. This reranking framework ensures higher precision in retrieved knowledge, directly impacting the quality of downstream generative tasks.

The problem formulation can be defined as follows:

Given:

- q_{init} as the initial query issued by the user,
- Q_{follow} as the set of predicted follow-up queries that capture potential user intent beyond the initial query,
- $KBcand$ as the candidate set of document chunks retrieved from the knowledge base based on q_{init}
- C as the set of all candidate chunks in the knowledge base,

The goal is to develop a reranking function $R(c, q_{init}, Q_{follow})$ that outputs a ranked list of chunks KB_{ranked}, where each chunk is assigned a relevance score based on the following factors:

- Semantic similarity to the initial query q_{init}
- Semantic similarity to the follow-up queries Q_{follow}
- Geometric consideration (penalty based on the Euclidean distance from the initial query)

3.2 Methodology Overview

The proposed system aims to enhance query-document relevance ranking through a hybrid reranking framework that incorporates both semantic similarity and geometric considerations. The methodology is structured around three primary components:

- Query Prediction: The model predicts a set of plausible follow-up queries Q_{follow} based on the initial query q_{init}. This prediction leverages a large language model (LLM) using Chain of Thoughts (CoT) reasoning prompting method, ensuring that the predicted queries align with the user's expected follow-ups.
- Chunk Embedding: Chunks of knowledge (document segments) and the queries are embedded into a shared semantic space using a pre-trained embedding model. This step enables the system to compute the similarity between the initial query and the knowledge chunks in a high-dimensional space.
- Weighted Reranking Algorithm: The reranking process computes relevance scores for each chunk $c \in KB_{cand}$ based on the initial query q_{init} and the predicted follow-up queries Q_{follow}. The reranking algorithm integrates both semantic similarity and geometric penalties to produce a final ranked list of knowledge chunks.

3.3 Reranking Algorithm

The reranking process applies a weighted scoring function that combines the cosine similarity of the chunks to the initial query and follow-up queries, with a penalty based on the Euclidean distance. The scoring function is defined as:

$$R(c, q_{init}, Q_{follow}) = \alpha \cdot S(c, q_{init}) + \beta \cdot \sum_{q \in Q_{follow}} S(c, q) + \gamma \cdot D(c, q_{init}) \tag{3}$$

Where:

- α and β are the weights that adjust the contribution of the initial query and follow-up queries, respectively.
- $S(c, q)$ represents the cosine similarity between a chunk c and query q, which quantifies their semantic relevance.
- γ is a weight factor that penalizes the Euclidean distance $D(c, q_{init})$, where D is the Euclidean distance between the embedding of the chunk c and the embedding of the initial query q_{init}. This penalty is introduced to refine the ranking by considering the spatial separation in the embedding space.
- $D(c, q_{init})$ is computed as the Euclidean distance between the chunk c and the initial query q_{init} in the embedding space, providing an additional geometric consideration.

3.4 Detailed Process Flow

- **Initialization**: Given an initial query q_{init}, retrieve a candidate set of chunks KB_{cand} from the knowledge base. These chunks are selected based on their raw semantic similarity to q_{init}. To achieve this, the methodology employs a document retrieval system powered by Chroma, a vectorstore that indexes the documents as embeddings. Each document is represented as a vector, which is created by applying the OpenAI Embeddings model. The retrieval process utilizes a similarity search (based on k-nearest neighbors) to retrieve the top K relevant documents for each question.
- **Query Prediction** : A language model GPT-4o-mini, is used to predict a set of follow-up queries Q_{follow}, which are likely to be of interest based on the initial query. The follow-up queries are generated by utilizing Chain-of-Thought (CoT) reasoning method for prompting. These follow-up queries help capture user intent that is not directly addressed by the initial query. They aim to provide additional context or clarity, helping to enhance the retrieval accuracy for the initial query.
- **Similarity Computation**: After document retrieval, the methodology computes the similarity between the user query, follow-up queries, and the retrieved document chunks. Initially, the question and the retrieved chunks are embedded using the OpenAI Embeddings model. The Euclidean distance between the question and each chunk is computed. This distance is normalized using the sigmoid function to ensure that the distances are mapped to a range between 0 and 1. The normalized distance is then added to the metadata of the retrieved chunks, which is crucial for the subsequent reranking process. The normalized Euclidean distance between a query

q and a candidate chunk c can be defined as:

$$D(q,c) = \sigma\left(\sqrt{\sum_{i=1}^{d}(v_{q_i} - v_{c_i})^2}\right) \quad \text{where} \quad \sigma(x) = \frac{1}{1+e^{-x}} \tag{4}$$

- v_{q_i} and v_{c_i} are the i-th components of the query and candidate chunk embedding vectors, respectively.
- d is the dimensionality of the embedding space.
- The square root of the sum of squared differences computes the Euclidean distance, and the sigmoid function $\sigma(x)$ normalizes this distance to a value between 0 and 1.

• **Reranking of Retrieved Documents** To improve the quality of the retrieved documents, the methodology applies a reranking algorithm that combines several factors:

  - **Initial query relevance score**: Computed by cosine similarity between the question and chunk embeddings.
  - **Follow-up query relevance score**: Aggregated from the similarity between the follow-up queries and chunk embeddings. The cosine similarity between the embedding vectors of a query q and a candidate chunk c can be defined as:

$$S(q,c) = \frac{\mathbf{v}_q \cdot \mathbf{v}_c}{\|\mathbf{v}_q\|\,\|\mathbf{v}_c\|} \tag{5}$$

  Where:

    * v_q is the embedding vector for the query q
    * v_c is the embedding vector for the candidate chunk c
    * $\cdot$ denotes the dot product, and
    * $||\cdot||$ denotes the norm (magnitude) of a vector.

  - **Normalized distance**: A penalty term derived from the Euclidean distance between the question and chunk.

  The final relevance score for each chunk is computed using a weighted sum of the above factors, with hyperparameters α, β, and γ controlling their relative importance. The chunks are then reranked based on these final scores.

• **Scoring**: For each chunk c in K_{cand}, compute the composite relevance score using the following formula:

$$Score(c, q_{init}, Q_{follow}) = \alpha \cdot S(c, q_{init}) + \beta \cdot \sum_{q \in Q_{follow}} S(c,q) + \gamma \cdot D(c, q_{init}) \tag{6}$$

  Where:

  - $S(c, q_{init})$ is the cosine similarity between the chunk c and the initial query q_{init}

- $\sum_{q \in Q_{follow}} S(c, q)$ is the sum of cosine similarities between the chunk c and each predicted follow-up query.
- $D(c, q_{init})$ is the Euclidean distance between the chunk c and the initial query q_{init}, penalized by the weight γ
- **Normalization**: The computed relevance scores are normalized to ensure comparability across chunks, making the values suitable for ranking.
- **Selection**: After calculating the relevance scores for all candidate chunks, select the top-K chunks with the highest scores to form the final ranked list KB_{ranked}

3.5 Hyperparameter Fine-Tuning and Evaluation

To optimize the performance of the reranking system, the weights α, β and γ are fine-tuned using the MS MARCO training set. The training process minimizes the loss between predicted relevance scores and ground-truth relevance labels. A grid search is conducted over a range of values for these parameters. The search space is defined as follows:

- α: A factor that controls the influence of the initial query relevance score.
- β: A factor that controls the influence of the follow-up query relevance score.
- γ: A factor that penalizes the distance between the question and chunk.

Evaluation metrics such as Normalized Discounted Cumulative Gain (NDCG@K) and Recall@K are used to benchmark the effectiveness of the reranking model. These metrics assess the ability of the system to return the most relevant chunks in the top-K positions.

The optimal hyperparameters are determined by evaluating the performance of the system on a set of evaluation metrics and selecting the combination that yields the best results.

3.6 Evaluation Metrics

The evaluation of the reranked results is performed using several commonly used information retrieval metrics:

- **Mean Reciprocal Rank (MRR)**: The average inverse of the rank of the first relevant chunk retrieved across all user queries.
- **Precision@k**: The fraction of relevant items in the top k retrieved chunks.
- **Recall@k**: The fraction of relevant chunks that are retrieved in the top k results.
- **Normalized Discounted Cumulative Gain (NDCG@k)**: Measures the ranking quality considering the relevance of higher-ranked results.
- **Mean Average Precision (MAP)**: Computes the average precision score for all question.

The Mean Average Precision (MAP) in information retrieval can be expressed as:

$$MAP = \frac{1}{Q} \sum_{i=1}^{Q} \left(\frac{1}{n_i} \sum_{k=1}^{N_i} (P@k \cdot rel_i(k))\right) \tag{7}$$

Where:

- Q = total number of queries
- n_i = number of relevant documents for query i
- N_i = total retrieved documents for query i
- $P@k$ = precision at rank k
- $rel_i(k)$ = 1 if the document at rank k is relevant, 0 otherwise

3.7 Summary

The methodology leverages a combination of semantic similarity (via cosine similarity) and geometric consideration (via Euclidean distance) to improve the relevance of knowledge chunks retrieved from a knowledge base. The integration of predicted follow-up queries and the penalty for spatial distance provides a more comprehensive approach to query understanding, leading to more accurate and user-intent aligned rankings.

4 Results

4.1 MS MARCO Dataset

The MS MARCO (Microsoft MAchine Reading COmprehension) dataset is a large-scale benchmark for information retrieval and question answering, consisting of real-world, anonymised user queries from the Bing search engine. Its questions are genuine user queries, often containing colloquialisms and ambiguities, which are paired with web documents from which human annotators have generated answers. This data makes MS MARCO a challenging and standard benchmark for evaluating the performance of retrieval systems on passage-ranking tasks, where the goal is to find and rank the most relevant text passages for a given query.

Table 1 presents the performance results on the MS MARCO dataset, showcasing the effectiveness of our proposed Follow-up Query (FUQ) reranker. The FUQ method, both in its non–fine-tuned and fine-tuned variants, outperforms traditional retrieval models like BM25 and neural approaches like T5 [29], BERT [30, 31], and DPR Ranker [32]. These baseline models, which rely on static query representations and matching based on embeddings, tend to suffer from vocabulary mismatches and fail to capture the evolving nature of user intent. Additionally, while the Multi Query strategy shows competitive performance, methods like Parent Document, Contextual Compression, and Ensemble do not capture evolving user intent as effectively as our FUQ reranker. Overall, these results suggest that incorporating LLM-generated follow-up queries significantly enhances the contextual representation of the initial query, leading to more effective retrieval in dynamic environments.

4.2 AI Regulations in Healthcare Applications Dataset

The AI Regulations in Healthcare Applications dataset is a custom domain-specific collection developed for this study to assess retrieval performance on specialized content. It consists of text chunks encapsulating key principles and risks related to AI in healthcare, focusing on themes like risk assessment, clinical bias, system transparency, and data privacy. The questions are designed to be high-level and thematic, requiring the system to synthesize information from multiple disparate chunks to form a comprehensive response. This dataset provides a robust test for evaluating a model's ability to handle nuanced, domain-specific terminology and complex queries that demand a deeper contextual understanding.

Table 2 presents the performance of various retrieval and reranking methods on the AI Regulations in Healthcare Applications dataset. The FUQ reranker, particularly in its fine-tuned version, shows notable improvements in MRR and NDCG, indicating that it excels in ranking relevant documents towards the top. This enhancement is especially important for RAG systems that rely on the quality of the top-ranked knowledge chunks for downstream tasks. The fine-tuning of FUQ helps improve its ability to adapt to the domain-specific context, leading to better attention to user intent. However, while FUQ achieves solid results in MRR and NDCG, methods such as BM25, T5, and DPR show competitive results in Recall@k and Precision@k—metrics that highlight the systems' ability to retrieve a broad range of relevant documents. For example, BM25 and T5 outperform FUQ in terms of Precision@k, which is key for ensuring relevant documents are within the top retrieved set, while parent document methods show higher performance in Recall@k, indicating their capacity to recover more relevant documents from the knowledge base. These differences highlight the strengths and trade-offs of the FUQ reranker, which excels in optimizing retrieval precision and relevance ranking but may benefit from further refinement to improve recall and overall retrieval breadth.

Table 2 Retrieval Results of Top 3 relevant chunks on MS-MARCO Dataset

Retriever	**MRR**	**Precision@3**	**Recall@3**	**NDCG@3**	**MAP**
			Reranker Baselines		
FUQ (NF)	0.3637	0.1625	0.4625	0.3917	0.3767
FUQ (F)	**0.3799**	**0.1708**	**0.4875**	**0.4078**	**0.3856**
BM25	0.1575	0.0750	0.2125	0.1490	0.1598
DPR	0.3192	0.1500	0.4250	0.3368	0.3239
Rank GPT	0.2021	0.0833	0.2313	0.1805	0.1952
T5	0.2865	0.1250	0.3500	0.2718	0.2885
BERT	0.2462	0.1042	0.2938	0.2174	0.2466
			Advanced Query Strategies		
Multi Query	0.3474	0.1500	0.4313	0.3482	0.3490
Parent Document	0.2146	0.1021	0.2750	0.2082	0.2108
Contextual Compression	0.0462	0.0208	0.0625	0.0487	0.0450
Ensemble	0.3247	0.1375	0.3875	0.3105	0.3226

Table 3 Retrieval Results of Top 3 relevant chunks on AI Regulations in Healthcare Applications Dataset

Retriever	**MRR**	**Precision@3**	**Recall@3**	**NDCG@3**	**MAP**
		Reranker Baselines			
FUQ (NF)	0.7400	0.3800	0.3631	0.4556	0.2462
FUQ (F)	**0.8222**	**0.5000**	**0.4230**	**0.5930**	**0.2440**
BM25	0.5833	0.4600	0.4153	0.4748	0.2849
DPR	0.6200	0.3000	0.2767	0.3388	0.1543
Rank GPT	0.6167	0.4000	0.3841	0.4387	0.2277
T5	0.7400	0.4600	0.4047	0.4865	0.3055
BERT	0.4119	0.2600	0.2521	0.2799	0.1278
		Advanced Query Strategies			
Multi Query	0.6333	0.4200	0.4020	0.4694	0.2647
Parent Document	0.7067	0.3800	0.3687	0.4542	0.2418
Contextual Compression	6167	0.4067	0.3841	0.4399	0.2294
Ensemble	0.8000	0.4600	0.4037	0.5259	0.3237

5 Impact of K-Value on Reranking Performance

To understand the impact of the number of retrieved chunks on our reranker's effectiveness, we conducted an analysis by varying the value of k, the number of top candidates passed to the reranker. We evaluated the performance of both our fine-tuned and non-fine-tuned rerankers for $k \in \{1, 2, 3, 4, 5\}$. The goal was to identify the optimal number of chunks that balances computational cost with retrieval quality, as retrieving too few chunks might miss relevant information, while too many could introduce noise and dilute the relevance of the top-ranked results.

The performance was measured using Mean Hit Rate, Mean Recall @ K, and Mean Reciprocal Rank (MRR). The results, depicted in Figure 3 and Figure 4, reveal clear trends in how the value of k influences retrieval outcomes.

As shown in Figure 3, the Mean Hit Rate for the non-fine-tuned model increases dramatically to a peak of 0.5 at $k = 3$, after which it declines sharply. In contrast, the fine-tuned model shows a substantial performance gain from $k = 1$ to $k = 2$ and then maintains a high and stable hit rate for $k \geq 2$. A similar pattern is observed for Mean Recall @ K in Figure 4. The non-fine-tuned model again peaks at $k = 3$, while the fine-tuned model achieves strong recall at $k = 2$ and stabilizes, consistently outperforming the non-fine-tuned version for all $k > 1$.

Analysis of the Mean Reciprocal Rank (MRR) metric showed a comparable trend. The fine-tuned model's MRR peaked at $k = 2$ (0.3779) and remained consistently higher than the non-fine-tuned model, which peaked at $k = 3$ (0.3779) before declining.

These results collectively suggest that $k = 3$ is the optimal setting for our non-fine-tuned reranker. However, for the more robust fine-tuned model, a value of $k = 2$ or $k = 3$ provides a strong balance, achieving near-peak performance without the need

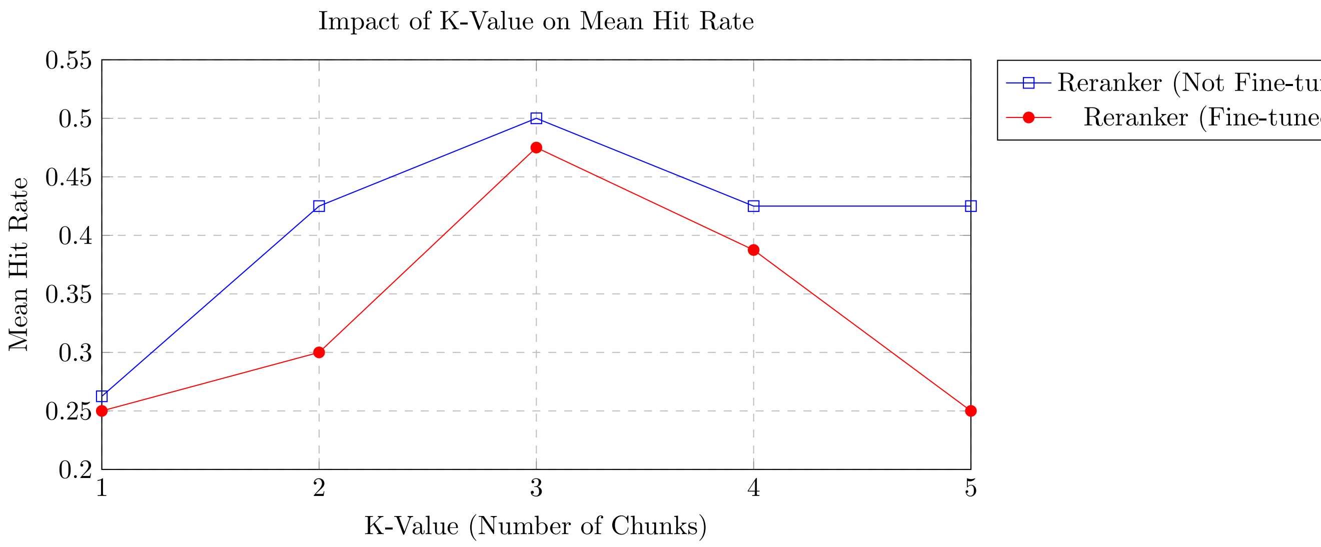

Fig. 3 Mean Hit Rate for different values of K. The fine-tuned model shows significant improvement and stability for K > 1, while the non-fine-tuned model's performance peaks sharply at K=3 before declining.

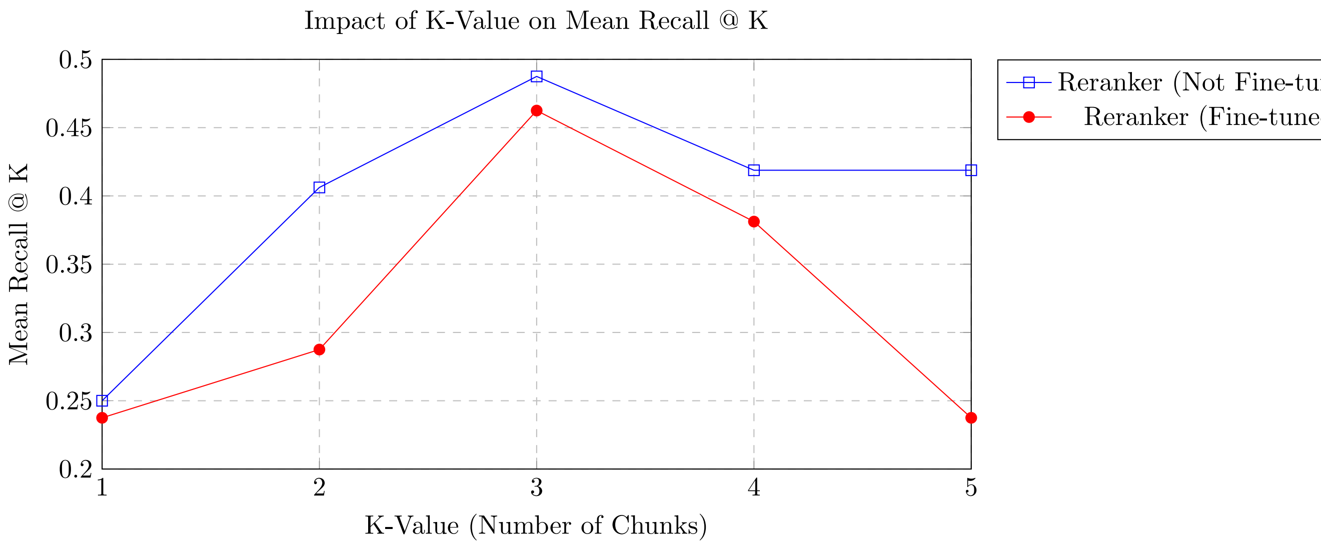

Fig. 4 Mean Recall @ K for different values of K. The trend mirrors the Mean Hit Rate, with the fine-tuned model providing superior recall and the non-fine-tuned model peaking at K=3.

to process additional, potentially noisy chunks. This highlights the efficiency of the fine-tuned reranker, as it can achieve superior results while considering a smaller set of candidate documents.

6 Limitations and Future Work

While our study demonstrates promising improvements in retrieval performance, there are several limitations that provide avenues for further research. First, our experiments were conducted on only two datasets—MS-MARCO and the AI Regulations in Healthcare Applications dataset—which may limit the immediate generalizability of our findings. Future studies could validate our approach across a broader range of domains. Second, our methodology currently relies on an initial retrieval step using vector similarity search with OpenAI embeddings, which is well-suited for sparse retrieval scenarios. It remains to be explored how our reranker might integrate with dense retrieval frameworks, such as dual encoder models, which address vocabulary mismatches differently. Third, our investigation focused on a specific set of prompt templates and fine-tuning strategies. Given the sensitivity of large language models to prompt design, exploring alternative formulations could yield further improvements. Finally, while the performance gains are encouraging, the computational cost of employing large language models in the reranking process is nontrivial. Future work could explore model distillation or more efficient architectures to facilitate practical deployment. Overall, these limitations open up exciting opportunities for extending and refining our approach in subsequent research.

7 Conclusion

In this paper, we presented a novel reranking framework for optimizing knowledge chunk ranking in Retrieval-Augmented Generation (RAG) systems by incorporating LLM-generated predicted follow-up queries. In contrast to conventional reranking methods that rely solely on the semantic similarity between the initial query and document embeddings, our approach dynamically integrates predicted follow-up queries to capture evolving user intent and contextual nuances. Our fine-tuned weighting mechanism effectively balances the contributions of the initial query and its follow-ups, leading to significant improvements in retrieval performance as demonstrated on both the MS-MARCO and AI Regulations in Healthcare Applications datasets. Notably, our results show enhanced hit rates, MRR, and other key metrics, underscoring the potential of embedding-driven reranking in delivering more relevant and context-aware retrieval outcomes. As large language models continue to advance, our work highlights promising avenues for further improvements, including refined prompt engineering, integration with dense retrieval architectures, and the development of more computationally efficient methods for practical deployment. Overall, our findings contribute to the advancement of adaptive RAG systems, paving the way for more personalized and robust information retrieval solutions.

Algorithm 1 LLM Follow-Up Query Generation

```
Require: $q_{\text{init}}$: Initial user query
 1: prompt_template: Few-Shot + CoT prompt template
 2: LLM: Chat-based language model
 3: parse_output: Function to parse the LLM's response
Ensure: $Q_{\text{follow}}$: List of predicted follow-up queries
 4: Format the prompt with $q_{\text{init}}$ using prompt_template
 5: for each query in $\{q_{\text{init}}\}$ do
 6:     Construct processing chain:

        final_chain ← user_query → prompt → LLM → OutputParser → parse_output

 7:     Invoke final_chain with $q_{\text{init}}$
 8:     Append the resulting follow-up queries to $Q_{\text{follow}}$
 9: end for
10: return $Q_{\text{follow}}$
```

Algorithm 2 Composite Score Computation for a Candidate Chunk

```
Require: $v_q$: Embedding vector of the initial query
 1: $v_c$: Embedding vector of the candidate chunk
 2: $Q_{\text{follow}}$: Set of follow-up queries
 3: $\alpha, \beta, \gamma$: Weighting hyperparameters
Ensure: $Score$: Composite relevance score for the candidate chunk
 4: $S_{\text{query}} \leftarrow \text{CosineSimilarity}(v_q, v_c)$
 5: $S_{\text{follow\_sum}} \leftarrow 0$
 6: for each follow-up query $q \in Q_{\text{follow}}$ do
 7:     $v_{q_{\text{follow}}} \leftarrow \text{ComputeEmbedding}(q)$
 8:     $S_{\text{follow\_sum}} \leftarrow S_{\text{follow\_sum}} + \text{CosineSimilarity}(v_{q_{\text{follow}}}, v_c)$
 9: end for
10: $S_{\text{follow\_avg}} \leftarrow \dfrac{S_{\text{follow\_sum}}}{|Q_{\text{follow}}|}$
11: $E_{\text{distance}} \leftarrow \text{EuclideanDistance}(v_q, v_c)$
12: $D_{\text{norm}} \leftarrow \text{Sigmoid}(E_{\text{distance}})$
13: $Score \leftarrow \alpha \cdot S_{\text{query}} + \beta \cdot S_{\text{follow\_avg}} + \gamma \cdot D_{\text{norm}}$
14: return $Score$
```

8 Algorithms, Program codes and Listings

Sample LLM Prompt Template for Follow-Up Query Generation

Context: You are an AI designed to assist in improving Retrieval-Augmented Generation (RAG) by generating follow-up questions.

Objective: Generate relevant follow-up questions that provide additional context or clarity to enhance retrieval accuracy.

Chain-of-Thought (CoT) Reasoning:
To accomplish this, follow these reasoning steps:
1. Break down the user's query into its key components (e.g., main topic, entities, ambiguities).
2. Identify additional details or perspectives (e.g., specific conditions or aspects).
3. Formulate two insightful follow-up questions.

Few-Shot Prompting:
Example:
User Query: "What are the effects of climate change?"
Follow-Up Questions:
"How does climate change specifically impact human health?"
"What are the economic consequences of climate change on developing countries?"

Supplementary information. Detailed code for evaluation of both datasets used can be provided upon request

Declarations

- Funding: No Financial Funding was involved.
- Competing interests: The authors have no relevant financial or non-financial interests to disclose.
- Author contribution: The conceptualization, methodology, analysis, and the writing of the original draft were performed by Kok Rhui Ong. The manuscript was reviewed and edited by Dr Wai Peng Wong. Both authors read and approved the final manuscript.
- Data availability: The MS MARCO dataset analysed during the current study is publicly available. The AI Regulations in Healthcare Applications dataset generated during the current study is available from the corresponding author on reasonable request.

References

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Proceedings of the 34th International Conference on Neural Information Processing Systems, 16 (2020) https://doi.org/10.5555/3495724.3496517

[2] Nogueira, R., Cho, K.: Passage re-ranking with bert. arXiv preprint arXiv:1901.04085 (2020) https://doi.org/10.48550/arXiv.1901.04085

[3] Nogueira, R., Yang, W., Cho, K., Lin, J.: Multi-stage document ranking with bert. arXiv preprint arXiv:1910.14424 (2019) https://doi.org/10.48550/arXiv.1910.14424

[4] Jiang, J.-Y., Wang, W.: Rin: Reformulation inference network for context-aware query suggestion. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 197–206. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3269206.3271808

[5] Trabelsi, M., Chen, Z., Davison, B.D., Heflin, J.: Neural ranking models for document retrieval. Information Retrieval Journal **24**(6), 400–444 (2021) https://doi.org/10.1007/s10791-021-09398-0

[6] Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W.B., Cheng, X.: A deep look into neural ranking models for information retrieval. Inf Process Manag **57**(6), 102067–102128 (2020) https://doi.org/10.1016/j.ipm.2019.102067

[7] Mitra, B., Craswell, N.: An introduction to neural information retrieval. Foundations and Trends® in Information Retrieval **13**(1), 1–126 (2018) https://doi.org/10.1561/1500000061

[8] Carpineto, C., Romano, G.: A survey of automatic query expansion in information retrieval. ACM Comput Surv **44**(1), 1–50 (2012) https://doi.org/10.1145/2071389.2071390

[9] Gienapp, L., Scells, H., Deckers, N., Bevendorff, J., Wang, S., Kiesel, J., Syed, S., Fröbe, M., Zuccon, G., Stein, B., Hagen, M., Potthast, M.: Evaluating generative ad hoc information retrieval. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), Washington, D.C., USA, pp. 1916–1929 (2024). https://doi.org/10.1145/3626772.3657849

[10] Schneider, P., Poelman, W., Rovatsos, M., Matthes, F.: Engineering conversational search systems: A review of applications, architectures, and functional components. In: Proceedings of the 6th Workshop on NLP for Conversational AI (NLP4ConvAI 2024), pp. 73–88. Association for Computational Linguistics, Bangkok, Thailand (2024). https://doi.org/10.48550/arXiv.2407.00997

[11] Eldin, S.S., Mohammed, A., Eldin, A.S., Hefny, H.: An enhanced opinion retrieval approach via implicit feature identification. J Intell Inf Syst **57**, 101–126 (2021) https://doi.org/10.1007/s10844-020-00622-9

[12] Chuang, Y.-S., Fang, W., Li, S.-W., Yih, W.-T., Glass, J.: Expand, rerank, and retrieve: Query reranking for open-domain question answering. In: Findings of the Association for Computational Linguistics: ACL 2023, pp. 12131–12147. Association for Computational Linguistics, Toronto, Canada (2023). https://doi.org/10.18653/v1/2023.findings-acl.768

[13] Li, M., Zhuang, H., Hui, K., Qin, Z., Lin, J., Jagerman, R., Wang, X., Bendersky, M.: Can query expansion improve generalization of strong cross-encoder rankers? In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, Washington D.C., USA, pp. 2321–2326 (2024). https://doi.org/10.1145/3626772.3657979

[14] Li, D., Shen, Y., Jin, R., Mao, Y., Wang, K., Chen, W.: Generation-augmented query expansion for code retrieval. arXiv preprint (2022) https://doi.org/10.48550/arXiv.2212.10692

[15] Chan, C.-M., Xu, C., Yuan, R., Luo, H., Xue, W., Guo, Y., Fu, J.: Rq-rag: Learning to refine queries for retrieval-augmented generation. In: Proceedings of the First Conference on Language Modeling, Philadelphia, Pennsylvania, United States (2024). https://doi.org/10.48550/arXiv.2404.00610

[16] Zhang, Q., Zheng, M., Chen, S., Liu, H., Fang, M.: Self data augmentation for open domain question answering. ACM Transactions on Information Systems **43**(2), 51 (2025) https://doi.org/10.1145/3707449

[17] Xu, S., Pang, L., Xu, J., Shen, H., Cheng, X.: List-aware reranking-truncation joint model for search and retrieval-augmented generation. In: Proceedings of the ACM Web Conference 2024, Singapore, pp. 1330–1340 (2024). https://doi.org/10.1145/3589334.3645336

[18] Berntson, A., Stoica Beck, A., Salvador Aguilera, A., Sunavala, F., Gisselbrecht, T., Chen, X.: Raising the bar for RAG excellence: Introducing generative query rewriting and new semantic ranker. Microsoft Tech Community Blog. Accessed: 16 February 2025 (2024). http://techcommunity.microsoft.com

[19] Dong, J., Fatemi, B., Perozzi, B., Yang, L.F., Tsitsulin, A.: Don't Forget to Connect! Improving RAG with Graph-based Reranking. arXiv preprint (2024). https://doi.org/10.48550/arXiv.2405.18414

[20] Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: Proceedings of the 25th ACM International Conference on Information and Knowledge Management, pp. 55–64. Association for Computing Machinery, Indianapolis, Indiana, USA (2016). https://doi.org/10.1145/2983323.2983769

[21] Xiong, C., Dai, Z., Callan, J., Liu, Z., Power, R.: End-to-end neural ad-hoc ranking with kernel pooling. In: Proceedings of the 40th International ACM SIGIR

Conference on Research and Development in Information Retrieval, pp. 55–64. Association for Computing Machinery, Tokyo, Japan (2017). https://doi.org/10.1145/3077136.3080809

[22] Sun, W., Yan, L., Ma, X., Wang, S., Ren, P., Chen, Z., Yin, D., Ren, Z.: Is chatgpt good at search? investigating large language models as re-ranking agents. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, pp. 14918–14937 (2023). https://doi.org/10.18653/v1/2023.emnlp-main.923

[23] Hauff, C., Kiseleva, J., Sanderson, M., Zamani, H., Zhang, Y.: Conversational search and recommendation: Introduction to the special issue. ACM Trans Inf Syst **39**, 1–6 (2021) https://doi.org/10.1145/3465272

[24] Kim, H., Choi, Y., Yang, T., Lee, H., Park, C., Lee, Y., Kim, J.Y., Kim, J.: Using LLMs to investigate correlations of conversational follow-up queries with user satisfaction. arXiv preprint (2024). https://doi.org/10.48550/arXiv.2407.13166

[25] Ma, X., Zhang, X., Pradeep, R., Lin, J.: Zero-Shot Listwise Document Reranking with a Large Language Model. arXiv preprint arXiv:2305.02156 (2023). https://doi.org/10.48550/arXiv.2305.02156

[26] Pradeep, R., Sharifymoghaddam, S., Lin, J.: RankVicuna: Zero-Shot Listwise Document Reranking with Open-Source Large Language Models. CoRR, `abs/2309.15088` (2023). https://doi.org/10.48550/ARXIV.2309.15088

[27] Wang, L., Yang, N., Wei, F.: Query2doc: Query expansion with large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, pp. 9414–9423 (2023). https://doi.org/10.18653/v1/2023.emnlp-main.585

[28] Jagerman, R., Zhuang, H., Qin, Z., Wang, X., Bendersky, M.: Query Expansion by Prompting Large Language Models. arXiv preprint arXiv:2305.03653 (2023). https://doi.org/10.48550/arXiv.2305.03653

[29] Zhuang, H., Qin, Z., Jagerman, R., Hui, K., Ma, J., Lu, J., Ni, J., Wang, X., Bendersky, M.: Rankt5: Fine-tuning t5 for text ranking with ranking losses. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, Taipei, Taiwan, pp. 2308–2313 (2023). https://doi.org/10.1145/3539618.3592047

[30] Zhuang, H., Qin, Z., Han, S., Wang, X., Bendersky, M., Najork, M.: Ensemble distillation for bert-based ranking models. In: Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval, pp. 131–136 (2021). https://doi.org/10.1145/3471158.3472238

[31] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep

bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, pp. 4171–4186 (2019). https://doi.org/10.18653/v1/N19-1423

[32] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.-t.: Dense passage retrieval for open-domain question answering. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 6769–6781. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.emnlp-main.550