Improving Retrieval Mechanism in Retrieval-Augmented Generation Architecture

Homayoun Afshari
Politecnico di Torino
Turin, Italy
homayoun.afshari@studenti.polito.it

Hossein Khodadadi
Politecnico di Torino
Turin, Italy
hossein.khodadadi@studenti.polito.it

Arash Daneshvar
Politecnico di Torino
Turin, Italy
arash.daneshvar@studenti.polito.it

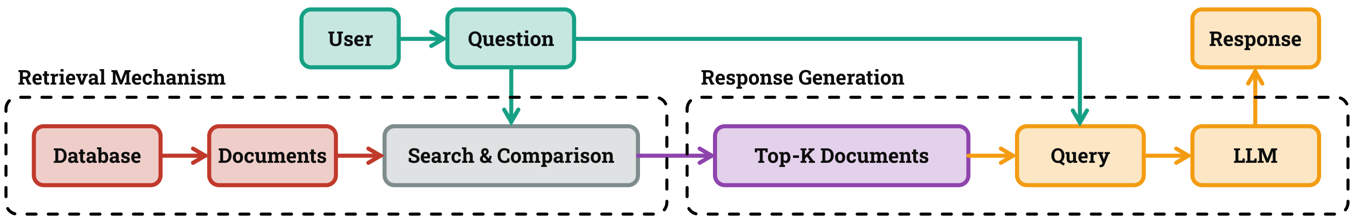


Figure 1: A high level demonstration of RAG architecture.

ABSTRACT

This research investigates advancements in document retrieval mechanisms within Retrieval-Augmented Generation (RAG) systems, with a focus on interplay between syntactic and semantic search methods. Using the MS-Marco and Hotpot-QA datasets, the study evaluates baseline approaches—including Named Entity Recognition (NER), BM25, and semantic search with embedding models—and introduces a novel methodology to enhance retrieval performance. The proposed approach employs a learnable mapping that optimizes question embeddings by aligning them with relevant document clusters through contrastive learning. The results demonstrate that the proposed method successfully refines semantic search in terms of different performance metrics. This work also highlights the importance of balancing semantic understanding with syntactic precision and offers insights for hybrid retrieval architectures in future research. The codebase for this study is available on GitHub via this link.

KEYWORDS

Retrieval-Augmented Generation, Contrastive Learning, Semantic Search, Syntactic Search, Document Embeddings

1 INTRODUCTION

Information Retrieval (IR) plays a crucial role in answering user questions with relevance [6]. To address this task, as illustrated in Figure 1, the recent trend has introduced the concept of Retrieval-Augmented Generation (RAG) [5]. These systems comprise a pipeline where the question is processed through a *Retrieval Mechanism* to identify the most relevant documents stored in a database [14]. These documents are then aggregated with the question to construct an enhanced query, which is subsequently fed into a Large Language Model (LLM) for *Response Generation* [14]. The RAG architecture was originally designed to help LLMs generate more informed responses, but the retrieval mechanism serves as a primary role in such architecture, as the quality of the final response depends heavily on the relevance of the retrieved documents [5].

The retrieval mechanism, as illustrated in Figure 1, is composed of two major concepts, *Search* and *Comparison*. The first concept typically follows one of two major approaches: *Exhaustive Search*, which involves traditional search methods such as brute-force or analyzing patterns within the user's question [6]; and *Vector Database* (VectorDB), which stores vector representations of documents to leverage Approximate Nearest Neighbor (ANN) algorithms for efficient search through them [11]. This efficiency can be achieved by organizing representations within optimized data structures, such as graphs, hash tables, or trees, to minimize retrieval time complexity [5, 10, 15]. The second concept in the retrieval mechanism involves two broad categories: *Syntactic Comparison*, which focuses on matching keywords and phrases based on their lexical and syntactic properties; and *Semantic Comparison*, which leverages semantic representations and relationships between words and concepts to identify relevant documents [11].

Despite the promising advancements in RAG, particularly regarding the retrieval mechanism, several challenges continue to hinder their reliability. To the best of our knowledge, the most common issues in this field can be considered as follows:

- **Incomplete Retrieval**: The system fails to accurately retrieve the most relevant documents, resulting in suboptimal context for downstream tasks [11].
- **Redundant Retrieval**: While relevant documents are retrieved, the system also retrieves a huge number of irrelevant ones, which can negatively affect its quality [27].
- **Lack of Well-Defined Evaluation Metrics**: The absence of clearly defined performance metrics makes it challenging to systematically evaluate and benchmark the effectiveness and efficiency of retrieval mechanisms.
- **Exceeding Context Window Limitations**: Retrieved documents may sometimes exceed the input context window size of the LLMs, which can limit their ability to process the entire context effectively and provide accurate responses.

Recognizing these challenges, our study aims to address some of the limitations and make an effort to advance the field of retrieval mechanisms. Our main contributions can be summarized as follows:

- **Baseline Formation**: We establish robust baselines by systematically implementing and evaluating state-of-the-art retrieval mechanisms. The baselines serve as a foundation for benchmarking the proposed methodology.
- **Innovative Techniques**: We propose a novel methodology to address the challenge of incomplete retrieval and improve the system's ability to retrieve the most relevant documents.
- **Quality Metrics**: We define and implement robust evaluation metrics to systematically assess the quality and relevance of retrieved documents, enabling reliable and consistent performance measurement.

The remainder of this study is organized as follows. In section 2, we review the state-of-the-art in the retrieval mechanism by categorizing existing approaches into syntactic and semantic strategies. Then, in section 3, we outline the design and implementation of the retrieval mechanism, including the task definition, baseline establishment, and a novel approach developed to enhance its performance. Afterwards, in section 4, we describe the experimental setup, including the datasets, implementation techniques, the evaluation metrics, and analysis of the proposed method in comparison with the baselines. Finally, in Section 5, we summarize the key findings of this study, discuss their implications, and propose potential directions for future research.

2 RELATED WORKS

This study centers around the comparison concept within the retrieval mechanism of the RAG architectures. Therefore, while the search strategies, such as exhaustive search and VectorDB approaches, have been extensively explored, this section delves into the existing body of research on comparison strategies employed within the retrieval mechanism. It is worth noting that the reference [8] provides a comprehensive review on the the core ideas and architectures of the search concept.

2.1 Syntactic Approaches

As mentioned, syntactic approaches focus on structural comparisons. Across different techniques in this category, Vector Space Models (VSM) are foundational, as they enable partial matching and ranked results [21]. These techniques start by representing documents and questions as vectors in a high-dimensional space. Then, their relevancy is determined through calculating measures like cosine similarity, often enhanced by weighting schemes such as Term-Frequency-Inverse-Document-Frequency (TF-IDF) to emphasize important terms [17].

Beyond VSM, more sophisticated syntactic methods delve deeper into the structural relationships within text [24]. Techniques such as dependency parsing and syntactic graph analysis extract features like grammatical relationships that enable comparisons based on structural similarity [6]. For instance, tree edit distance and subtree matching can be employed to assess the similarity between different sentence structures [6]. However, these approaches are outperformed by probabilistic methods such as as Best-Matching 25 (BM25) that improve upon TF-IDF by introducing term saturation and document length normalization to balance term frequency and document properties for relevance scoring [21]. This method remains a widely used approach in IR [20].

Additionally, as another set of approaches, Named Entity Recognition (NER) also tries to further refine the syntactic approaches by focusing on entities such as names, dates, and locations to improve domain-specific search accuracy, which is particularly effective in tasks such as question answering [22]. Similarly, keyword/topic extraction identifies salient terms and concepts within documents, which can be used as indexing terms [17] or for question expansion [17], which helps mitigate the vocabulary mismatch. These category of approaches are further improved by combining syntactic parsing with the use of lexical resources such as WordNet [6]. This integration bridges the limitations of traditional NER, keyword-matching, or topic-matching methods by creating more linguistically informed systems, which can improve precision by analyzing the underlying syntactic structure rather than solely relying on entities, keywords or topics [13].

2.2 Semantic Approaches

Semantic methods aim to capture deeper meanings and contextual relationships within texts to enable more accurate and meaningful retrieval. Pre-trained Language Models (PLMs) have transformed semantic retrieval by generating dense, context-aware embeddings for both questions and documents. Techniques such as dual-encoder models, contrastive learning, and efficient indexing significantly enhance scalability and retrieval effectiveness compared to traditional sparse methods [27].

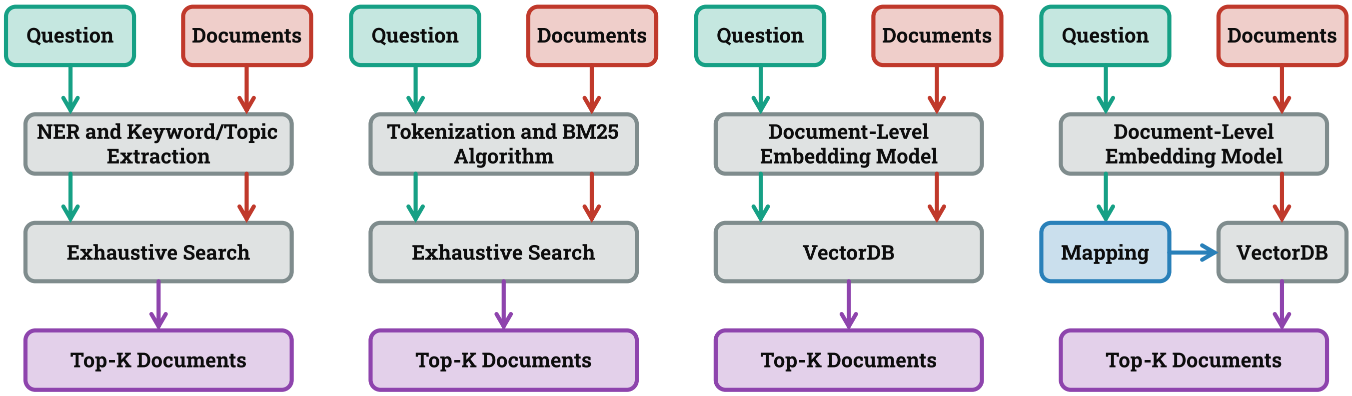
Dense Passage Retrieval (DPR) serves as a prime example of leveraging dense, context-aware embeddings for semantic retrieval [11]. Designed for open-domain question answering, DPR utilizes a dual-encoder architecture where two BERT-based encoders independently process queries and documents to generate dense embeddings. By optimizing the dot product similarity between question and document embeddings, it enhances retrieval accuracy through the use of positive and hard negative pairs during training [11].

Another innovative approach, consists of a two-stage retrieval method combining traditional and semantic methods to enhance performance. Initially, a traditional retrieval model like BM25 efficiently filters and retrieves a set of candidate documents. These candidates are then re-ranked using a BERT-based model that evaluates their semantic relevance to the question. This sequential process highlights how integrating syntactic retrieval techniques with deep learning models can significantly improve retrieval accuracy while maintaining computational efficiency [19].

Contextualized Late Interaction over BERT (ColBERT) exemplifies another approach that combines dense semantic embeddings from BERT with efficient late interaction for precise matching. This method achieves scalability and accuracy in large-scale retrieval tasks [12]. Similarly, Sparse Lexical and Expansion Model for Information Retrieval (SPLADE) integrates sparse lexical representations with dense contextualized embeddings. By leveraging transformer-based models, SPLADE balances interpretability, efficiency, and semantic richness [7].

3 METHODOLOGY

According to what we discussed in previous sections, once a user interacts with a RAG system by posing a question, like q, the system accesses its pre-stored database, like $\mathbb{D}$, which contains a collection



(a) B1: Syntactic Search I (b) B2: Syntactic Search II (c) B3: Semantic Search (d) Proposed Method

Figure 2: Different strategies used in retrieval mechanism.

of documents that may or may not be relevant to q. In an ideal situation, we expect the retrieval mechanism to retrieve $\mathbb{G}(q) \subseteq \mathbb{D}$, referred to as the *ground truth*, representing the optimal set of information about the question. In practice, however, the retrieval mechanism retrieves an ordered set of documents, like $\mathbb{R}_K(q) \subseteq \mathbb{D}$, where K documents are ranked by their *relevancy* to the question. Both the definition of the relevancy and the value of K, referred to as the *retrieval capacity*, are based on how the retrieval mechanism is designed to align the retrieved documents with the user's question. We will explore these concepts in greater details later.

3.1 Task Definition

The first step in our methodology during this project is to define the main task. In nutshell, we can formulate it as designing a retrieval mechanism to obtain an optimal $\mathbb{R}_K(q)$ such that:

$$\mathbb{G}(q) \subseteq \mathbb{R}_K(q), \tag{1a}$$

$$\mathbb{R}_K(q) \subseteq \mathbb{G}(q), \tag{1b}$$

where the conditions respectively ensure that all the relevant documents are retrieved and no irrelevant one is included. Nevertheless, since, as demonstrated in Figure 1, the subsequent block after the retrieval mechanism in a RAG system is a response-generating LLM, our focus in this project is primarily satisfying condition (1a). In other words, our primary goal is to avoid missing critical information at the cost of including some additional non-relevant ones and relying on the ability of LLMs to handle those redundant documents. Accordingly, we can define $\mathbb{R}_K(q)$ as follows:

$$\mathbb{R}_K(q) = \left\{ d_i \;\middle|\; d_i \in \mathbb{D} \wedge |\mathbb{R}_K| = K \right\}, \tag{2}$$

where, d represents a document chosen from $\mathbb{D}$ and i denotes its ordering position, which is also defined as follows:

$$\forall i_1, i_2 \in \{1, 2, \cdots, K\} : i_1 < i_2 \iff r\left(q, d_{i_1}\right) > r\left(q, d_{i_2}\right), \tag{3}$$

where r is the function used to measure relevancy. It is worth noting that we always have $r(\cdot) \in [0, 1]$.

3.2 Baseline Formation

The next step in our methodology is to establish a baseline for analyzing the task outlined in subsection 3.1. For this purpose, while the literature consistently demonstrates that semantic strategies outperform syntactic ones, we aim to maintain a general perspective by considering both categories as potential baselines. Therefore, we select state-of-the-art methods from each category and implement them systematically. Ultimately, the strongest method will serve as the foundation for our proposed approach.

The first baseline is *Syntactic Search I* (B1). In this strategy, as illustrated in Figure 2a, NER is combined with keyword/topic extraction to perform a character-level search within $\mathbb{D}$ for a given q. B1 leverages the concept of key-value pairs, which is originally derived from NER, and generalizes it to include keywords and topics by treating them as additional entities. In other words, B1 views keywords and topics also as entity types, with their names being *keyword* and *topic* respectively. Consequently, B1 starts with augmenting each document in $\mathbb{D}$ by extracting its key-value pairs in the form of (name, entity), (keyword, a possible keyword), or (topic, a possible topic). Then, the same process is applied to q, which enables comparing the values of each pair in the question with that of the documents at character-level to calculate a relevancy score for each document through an exhaustive search. Accordingly, we can define the relevancy function as follows:

$$\begin{aligned} r_{B1}(q, d) &= s_{ch}\left(p_q, p_d\right) \text{ given that} \\ p_x &= \left\{ (k, v) \;\middle|\; (k, v) \text{ is a key-value pair in } x \right\}, \end{aligned} \tag{4}$$

which can be written as:

$$\begin{aligned} s_{ch}\left(p_q, p_d\right) &= \frac{\left|\mathbb{K}_q \cap \mathbb{K}_d\right|}{\left|\mathbb{K}_q\right|} \operatorname*{agg}_{\substack{k \in \mathbb{K}_q \cap \mathbb{K}_d \\ (k, v_q) \in p_q \\ (k, v_d) \in p_d}} \left\{ c_{ch}\left(v_q, v_d\right) \right\} \text{ given that} \\ \mathbb{K}_x &= \left\{ k \;\middle|\; k \text{ is a key in } x \right\}, \end{aligned} \tag{5}$$

where c_{ch} is the function used to compare values at character-level and agg is the operator that aggregates the comparisons between different pairs of values of q and d.

The second baseline is *Syntactic Search II* (B2), which, as depicted in Figure 2b, utilizes the BM25 algorithm for token-level syntactic search. Similar to the first strategy, B2 also start with augmenting $\mathbb{D}$, but here, it only employs tokenization. Then, once the question q is also tokenized in the same manner, the BM25 algorithm computes a relevancy score for each document with respect to the question and retrieves the top-ranked documents through an exhaustive search. In this strategy, the relevancy function is defined as follows:

$$\begin{aligned} r_{B2}\left(q,d\right) &= s_{bm25}\left(t_q,t_d\right) \text{ given that} \\ t_x &= tok\left(x\right), \end{aligned} \tag{6}$$

where tok is the tokenizer function, and therefore, we have:

$$s_{bm25}\left(t_q,t_d\right) = c_{bm25}\left(t_q,t_d\right), \tag{7}$$

where c_{bm25} is the function that applies BM25 algorithm.

The final baseline is *Semantic Search* (B3), where embedding models and VectorDBs are used to perform a document-level semantic search, as illustrated in Figure 2c. In B3, $\mathbb{D}$ is again augmented using a pre-trained document-level embedding model to encode each document into an embedding vector. These document embeddings are then indexed in a VectorDB to enable efficient similarity search, which, after applying the embedding on q as well, retrieves documents that are semantically most relevant to the question. For B3, we can define the relevancy function as follows:

$$\begin{aligned} r_{B3}\left(q,d\right) &= s_{emb}\left(u_q,u_d\right) \text{ given that} \\ u_x &= enc\left(x\right), \end{aligned} \tag{8}$$

where enc is the encoder function, and therefore, we have:

$$s_{emb}\left(u_q,u_d\right) = c_{emb}\left(u_q,u_d\right), \tag{9}$$

where c_{emb} is that function that compares embeddings.

3.3 Proposed Method

With the baseline established, the final step in our methodology is developing an enhanced retrieval mechanism. Building upon the semantic search strategy outlined in the baseline, our proposal is to improve the alignment of the question embedding with that of its relevant documents. The motivation behind this approach stems from two key observations in the literature. First, semantic retrieval methods consistently outperform syntactic ones in capturing nuanced relationships between questions and documents. Consequently, there is limited value in further exploring syntactic approaches, as their limitations create a barrier to achieve optimal performance in most IR tasks [27]. Secondly, as illustrated in Figure 3, semantic methods inherently operate by identifying the cluster of embeddings in the embedding space that correspond to the most relevant documents for a given question, i.e., locating $\mathbb{G}\left(q\right)$. Therefore, our proposal is to capitalize on this clustering property by actively steering the question embedding toward the centroid of that *relevant cluster* using learnable parameters. Accordingly, the relevancy function for our method is defined as:

$$\begin{aligned} r_P(q,d) &= s_{emd}\left(u_q^*,u_d\right) \text{ given that} \\ u_q^* &= Wu_q+b, \end{aligned} \tag{10}$$

where W is a learnable weight matrix that applies a linear transformation to the question embedding, and b is a bias vector that shifts the mapped embedding. Together, these parameters enable the mapper to achieve the described goal.

To enable the learning process of the proposed mapper, we need to define a loss function. Referring to Figure 3, our proposed method tries to map the question embedding closer to the embeddings of its relevant documents, referred to as the *positives*, which are the members of $\mathbb{G}\left(q\right)$, and farther from the irrelevant ones, referred to as the *negatives*, which are the members of $\mathbb{D}\setminus\mathbb{G}\left(q\right)$. Consequently, our loss function adopts a contrastive formulation:

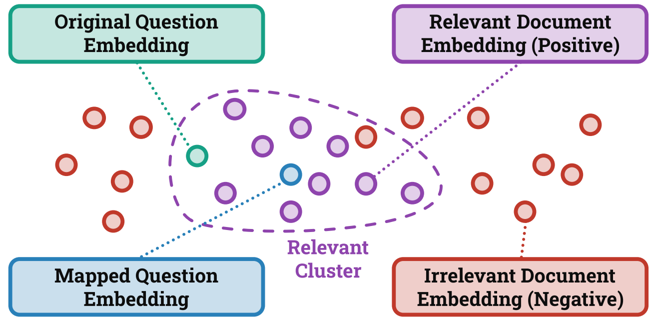


Figure 3: A conceptual demonstration of how the proposed method is built on top of B3 to push the original question embedding (green circle) from the boundary of the relevant cluster toward its center (blue circle).

$$l(q) = \left\|u_q^*-u_q^+\right\|_P + \min\left\{0,\left(\Gamma-\left\|u_q^*-u_q^-\right\|_P\right)\right\}, \tag{11}$$

where P and Γ are both hyper-parameters. The first one denotes the order of the norm operation, which is incorporated to provide the mapper with a sense of distance in the embedding space. The second one is a margin, which is used to properly separate the question embedding from the negatives. In addition, u_q^+ and u_q^- represent the aggregations of positives and negatives respectively, which are computed as follows:

$$u_q^+ = \frac{1}{\left|\mathbb{S}_q^+\right|}\sum_{d\in\mathbb{S}_q^+} u_d \wedge u_q^- = \frac{1}{\left|\mathbb{S}_q^-\right|}\sum_{d\in\mathbb{S}_q^-} u_d, \tag{12}$$

where $\mathbb{S}_q^+ \subseteq \mathbb{G}\left(q\right)$ and $\mathbb{S}_q^- \subseteq \mathbb{D}\setminus\mathbb{G}\left(q\right)$ are the set of positives and negatives associated with q. The selection of these sets depends on a predefined sampling strategy to ensure the mapper is trained effectively. We will further discuss these concepts later.

4 EXPERIMENTS

In this section, we outline the experimental process carried out during the project. For this purpose, we utilize two datasets, MicroSoft Machine reading comprehension (MS-Marco [18]) and Hotpot Question-Answering (HotpotQA [25]), which are specifically designed for developing and evaluating IR systems.

4.1 Evaluation Metrics

To assess the effectiveness of the methods employed in this study, we utilize three key metrics. Let $\mathbb{Q}$ denote the set of all the possible questions that we want to evaluate. The first two metrics are commonly used in IR tasks and are defined as follows:

$$R_k = \frac{1}{|\mathbb{Q}|}\sum_{q\in\mathbb{Q}} \frac{\left|\mathbb{R}_k\left(q\right)\cap\mathbb{G}\left(q\right)\right|}{\left|\mathbb{G}\left(q\right)\right|} \tag{13a}$$

$$M_K = \frac{1}{|\mathbb{Q}|}\sum_{q\in\mathbb{Q}} \min_{\forall i=1,2,\cdots,K}\left\{\frac{1}{i}\;\middle|\;\mathbb{R}_i\left(q\right)\subseteq\mathbb{G}\left(q\right)\right\} \tag{13b}$$

where R_k, referred to as the *Recall@k*, measures the proportion of the relevant documents retrieved within the top k results. Also, M_K, referred to as the *Mean Reciprocal Rank* (MRR), evaluates the rank of the first relevant document across all K retrieved ones [26]. In addition to these metrics, we also introduce a new one to provide a more comprehensive assessment of system performance by addressing its various aspects. This metric is defined as follows:

$$M_K^P = \frac{1}{|\mathbb{Q}|} \sum_{q \in \mathbb{Q}} \min_{\forall i=1,2,\cdots,K} \left\{ \frac{|\mathbb{G}(q)|}{i} \;\middle|\; \mathbb{G}(q) \subseteq \mathbb{R}_i(q) \right\} \tag{14}$$

where M_K^P, referred to as the *Pessimistic MRR*, measures the rank at which all relevant documents are included within the K retrieved ones. Unlike the traditional MRR, Optimistic MRR emphasizes the rank position where the entire set of relevant documents is retrieved, which tries to quantify how capable the retrieval mechanism is to capture all pertinent information within a specified cutoff.

4.2 Baseline Implementation

As discussed in section 1, B1 incorporates NER and keyword/topic extraction. For these techniques, we use *spaCy* and *Llama3-8b*, respectively. The first tool, spaCy, is a Python package designed for various Natural Language Processing (NLP) tasks, including NER [4]. The second tool, is an open-source LLM that we employed for keyword/topic extraction through carefully designed prompt engineering [16]. Additionally, we implement the c_{ch} function using two string similarity measures: *Levenshtein Distance*, which calculates the minimum number of single-character edits required to transform one string into another, and *Jaro-Winkler Distance*, which gives higher similarity to strings with matching prefixes [3]. Moreover, for the agg operator, we use minimization, averaging, and maximization techniques, and for the exhaustion search, we implement a simple *For-Loop* structure.

Regarding B2, the *tok* function is implemented by tokenization through *Split-by-Space* or *Lemmatization* using *spaCy* [1]. We also use *rank-bm25*, which implements the BM25 algorithm through the BM25Okapi method [2], alongside an exhaustive search approach based on a for-loop structure again. As for B3, the *enc* function leverages pre-trained foundation models available on HuggingFace, which include *all-mpnet-base-v2*, *multi-qa-mpnet-base-dot-v1*, and *all-distilroberta-v1*, implemented using the *SentenceTransformer* Python package. Also, in this strategy, we utilize Facebook AI Similarity Search (FAISS) and Scalable Nearest Neighbors (ScaNN) as our VectorDB [9, 23], which implement the c_{emb} function through *Cosine Similarity*, *Inner Product*, or *Euclidean Distance*.

4.3 Mapper Implementation

To implement the proposed method, we use *PyTorch* and establish a GPU-based training and evaluation framework utilizing the *torch* Python package. The method introduces several hyper-parameters, including the batch size (B), learning rate (σ), norm order (P), margin (Γ), the mode of forming positive and negative sets ($\mathbb{S}q^+$ and $\mathbb{S}q^-$), the preferred total number of positives and negatives (T) in those sets, and the preferred proportion of positives (ρ). Among these hyper-parameters, the mode of forming $\mathbb{S}q^+$ and $\mathbb{S}q^-$ is the most important. It can take "random", "far-far", "far-close", "close-far", or "close-close" as values. The "random" mode selects embeddings randomly, but for the other modes, the two parts in the name (e.g., "far" and "close" in "far-close") specify how the positives and negatives are respectively selected according to their distance from the question embedding in the embedding space. For this purpose, "close" indicates that embeddings are chosen based on their ranking in B3, where higher-ranked embeddings are preferred, and "far" means embeddings are selected based on their lower rank. For example, in "far-close" mode, positives and negatives are respectively chosen from lower-ranked and higher-ranked document embeddings. We will discuss the effects of these selection modes later.

4.4 Result

We began our tests by setting the retrieval capacity (K) to 50, a relatively strict value compared to the typical value of 100 commonly used in the literature [5]. Based on our initial experiments, we determined the best configurations for c_{ch}, agg, *tok*, *enc*, VectorDB, and c_{emb} to be Jaro-Winkler distance, maximization, lemmatization, all-mpnet-base-v2, FAISS, and cosine similarity. Consequently, Table 1 summarizes the baseline results across both datasets, created by testing only 100 queries selected from the entire datasets. As shown, B3 outperforms the alternatives on the MS-Marco dataset, establishing it as the optimal choice for our project's main baseline. On the other hand, B2 performs better on the Hotpot-QA dataset, suggesting that semantic methods may not always be the best approach depending on the dataset. However, since our primary focus is on MS-Marco, this result is secondary to our main objectives.

Table 1: Comparison of the baselines.

Dataset	**Strategy**	M_K	M_K^P	R_1	R_5	R_{10}
MS-Marco	B1	0.91	0.37	0.11	0.49	0.69
	B2	0.96	0.45	0.12	0.53	0.76
	B3	**1.00**	**0.90**	**0.13**	**0.64**	**0.97**
Hotpot-QA	B1	0.83	0.04	0.08	0.25	0.36
	B2	**1.00**	**0.62**	**0.10**	**0.48**	**0.84**
	B3	0.98	0.24	0.10	0.39	0.61

The next step involved tuning the hyper-parameter of the mapper. For this purpose, we utilized Weights & Biases and searched a total of 1643 different configurations. In this process, we conducted a Bayesian method with the objective of maximizing M_K^P on the validation set of 1000 queries chosen from the MS-Marco dataset. As our search space, we assumed $B \in \{512, 768\}$, $\sigma \in [0.0005, 0.0010]$, $P \in \{2, 3, 5, 8\}$, $\Gamma \in [0.2, 0.4]$, mode can take "random", "far-far", "far-close", "close-far", or "close-close", $T \in \{2, 3, 5, 8\}$, and $\rho \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$. The results of this tuning process are summarized in Table 2. Further details are available in this link.

As shown in Table 2, the most distinctive hyper-parameters are the mode and the proportion of positives (ρ). These results reveal that the mapper exhibits a strong preference for setting $\rho = 0.75$, which ensures that the question embedding is positioned closer to the positives in most cases. Furthermore, the mapper tends to select the farthest positives and negatives as its anchors in its

Table 2: Result of hyper-parameter tuning.

#	B	σ	P	Γ	*mode*	T	ρ	M_K^P
1	**512**	**0.0006598**	**3**	**0.2068**	**f-f**	**2**	**0.75**	**0.8171**
2	512	0.0006493	2	0.2259	f-f	2	1.00	0.8169
3	512	0.0007584	5	0.2250	f-f	3	0.75	0.8167
4	768	0.0005064	2	0.2470	f-c	3	0.75	0.8165
5	512	0.0005845	3	0.2129	f-f	2	0.5	0.8163

mapping process. By prioritizing the farthest positives, the mapper ensures that the question embedding is drawn closer to all the positives. Additionally, selecting the farthest negatives allows for a larger margin between positives and negatives, which enhances generalizability, i.e., it helps the mapper form a clearer boundary between positives and negatives to reduce possible overlaps and improve robustness against unseen question embeddings.

Table 3: Comparison of B3 and mapper.

Dataset	**Strategy**	M_K	M_K^P	R_1	R_5	R_{10}
MS-Marco	B3	0.98	0.77	0.12	0.61	0.91
	Mapper	**0.99**	**0.84**	**0.13**	**0.61**	**0.95**
Hotpot-QA	**B3**	**0.95**	**0.12**	**0.10**	**0.31**	**0.43**
	Mapper	0.82	0.10	0.08	0.26	0.38

Our final evaluation involved testing the mapper on a set of 1000 queries selected from both the MS-Marco and Hotpot-QA datasets. The results, summarized in Table 3, reveal a clear trend. For the MS-Marco dataset, our proposed method consistently outperforms B3 across all the performance metrics, demonstrating the mapper's effectiveness in refining semantic embeddings to enhance retrieval mechanisms. However, for the Hotpot-QA dataset, the mapper underperforms compared to B3, which can be attributed to two factors. First, this evaluation served as a domain adaptation test with no prior fine-tuning. Secondly, as seen in Table 1, the Hotpot-QA dataset inherently favors syntactic methods, which makes it a challenging candidate for the so-called test.

5 CONCLUSION

In this study, we investigated various retrieval mechanisms in RAG architecture. We established multiple baselines and introduced a novel approach to optimize the embedding space for tasks emphasizing semantic understanding. This approach involved a learnable mapping to align question embeddings with their relevant documents using contrastive loss. Our results confirm that the proposed method achieved its objective. Additionally, we highlighted the significance of dataset-specific tuning and the limitations of a one-size-fits-all strategy in IR tasks. This insight points to a promising direction for future work in this field: the development of hybrid methods that integrate both semantic and syntactic approaches.

REFERENCES

[1] Vimala Balakrishnan and Ethel Lloyd-Yemoh. 2014. Stemming and lemmatization: A comparison of retrieval performances. (2014).

[2] Dorian Stuart Brown. Feb 16 2022. Rank-BM25: A two line search engine. https://pypi.org/project/rank-bm25/ Last accessed 24 January 2025.

[3] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg, et al. 2003. A Comparison of String Distance Metrics for Name-Matching Tasks.. In *IIWeb*, Vol. 3. 73–78.

[4] explosion.ai. Jan 14 2025. spaCy: Industrial-strength NLP. https://pypi.org/project/spacy/ Last accessed 24 January 2025.

[5] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.

[6] Antonio Ferrandez. 2011. Lexical and syntactic knowledge for information retrieval. *Information processing & management* 47, 5 (2011), 692–705.

[7] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[8] Yikun Han, Chunjiang Liu, and Pengfei Wang. 2023. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703* (2023).

[9] Jeff Johnson Hervé Jegou, Matthijs Douze. 29 March 2017. FAISS: A Library for Efficient Similarity Search. https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/ Last accessed 24 January 2025.

[10] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[11] 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* (2020).

[12] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 39–48.

[13] Vitaly Klyuev and Vladimir Oleshchuk. 2007. Semantic retrieval of text documents. In *7th IEEE International Conference on Computer and Information Technology (CIT 2007)*. IEEE, 189–193.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[15] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[16] Meta. April 18 2024. meta-llama/Meta-Llama-3-8B. https://huggingface.co/meta-llama/Meta-Llama-3-8B Last accessed 24 January 2025.

[17] Mandar Mitra and BB Chaudhuri. 2000. Information retrieval from documents: A survey. *Information retrieval* 2 (2000), 141–163.

[18] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset. (2016).

[19] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[20] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[21] Francis A Ruambo and Mrindoko R Nicholaus. 2019. Towards enhancing information retrieval systems: A brief survey of strategies and challenges. In *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 1–8.

[22] Amit Singh. 2012. Entity based q&a retrieval. In *Proceedings of the 2012 Joint conference on empirical methods in natural language processing and computational natural language learning*. 1266–1277.

[23] Philip Sun. 28 July 2020. Announcing ScaNN: Efficient Vector Similarity Search. https://research.google/blog/announcing-scann-efficient-vector-similarity-search/ Last accessed 24 January 2025.

[24] Bernd Teufel and Stephanie Schmidt. 1988. Full text retrieval based on syntactic similarities. *Information Systems* 13, 1 (1988), 65–70.

[25] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).

[26] Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: BERT and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*. 1154–1156.

[27] 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 (2024), 1–60.