# Performance evaluation and improvement using text-based information retrieval technique

Ken Yeh
University of Twente
Enschede, Netherlands
p.yeh@student.utwente.nl

Hsin-Hui Huang
University of Twente
Enschede, Netherlands
h.huang-4@student.utwente.nl

Peixuan Wu
University of Twente
Enschede, Netherlands
p.wu-2@student@utwente.nl

## Abstract

This report investigates the effectiveness of advanced techniques in text-based information retrieval (IR), focusing on performance improvements through modern machine learning methods. Traditional IR models, such as n-gram, are compared with deep learning-based approaches, specifically the Sentence-BERT (SBERT) model, which leverages neural embeddings for semantic similarity assessment. Using the TREC Genomics dataset, the study implements a bi-encoder and cross-encoder retrieval architecture, aiming to enhance retrieval precision. Results show that SBERT significantly outperforms traditional models in key metrics, including mean precision and mean average precision, demonstrating its potential to improve IR systems' accuracy. Limitations in cosine similarity, token limits, and limitations on the pre-train model are discussed, along with future recommendations to utilize domain-specific models for enhanced retrieval in scientific texts.

## Keywords

Information Retrieval, text-based, performance, Python, Deep learning, BERT

## 1 Introduction

Information Retrieval (IR) systems are essential for efficiently locating relevant documents and information from large data collections. These systems are crucial in applications such as search engines, digital libraries, and recommendation systems. A key challenge in IR is determining how effectively the system can retrieve documents that are relevant to a user's query. With the ever-increasing volume of information available today, traditional retrieval models face limitations in delivering accurate and relevant results.

Machine learning has emerged as a powerful tool for enhancing the effectiveness of IR systems. Machine learning models can significantly improve retrieval performance by learning complex relationships between search terms and relevant documents. For instance, deep learning models can transform both queries and documents into vector representations, enabling semantic similarity to be measured using techniques like cosine similarity or dot product. This allows the retrieval process to move beyond mere keyword matching and focus more on the underlying meaning, ultimately leading to more accurate and relevant search results.

Current IR approaches can be broadly categorized into traditional and modern machine learning-based methods. Traditional models, such as Boolean and probabilistic models, continue to be widely used in systems like Elasticsearch. These models rely on keyword matching and probabilistic relevance estimates to retrieve documents. Additionally, language models, including those utilizing Jelinek-Mercer and Dirichlet smoothing, are utilised to estimate query likelihoods and improve retrieval quality. More recent developments involve integrating machine learning, particularly deep learning, into IR systems to leverage pre-trained embeddings and learn more sophisticated text relationships.

In this project, our objective is to enhance the baseline IR model by employing different retrieval techniques to achieve greater precision. One critical aspect of improving IR performance is the selection of more advanced retrieval models and algorithms. Factors such as query representation, model architecture, and retrieval algorithms play significant roles in determining the effectiveness of an IR system. In our approach, we aim to utilize more advanced retrieval models, including those based on deep learning, to optimize document retrieval and improve overall relevance. We believe that incorporating modern machine learning techniques is vital for enhancing the performance of IR systems, particularly when dealing with large-scale and dynamic datasets.

## 2 Dataset

The datasets used in this project are sourced from assignment 1, the Text REtrieval Conference(TREC) Genomics dataset. The primary dataset file is FIR-s05-medline.json, which contains all the documents regarding medical articles from MEDLINE, a primary component of PubMed[1]. It includes fields such as titles, abstracts, keywords, and more. The second file, FIR-s05-training-queries-simple.txt, includes all the search queries along with their corresponding query IDs. The third file, FIR-s05-training-qrels.txt, contains the relevant document IDs for each query. In total, there are about 260,000 documents and 38 queries.

## 3 Research Methods

This section reveals the research method that has been applied to improve the performance of retrieving information.

### 3.1 Sentence-BERT (SBERT)

In this section, we aim to use a more accurate retrieval model to achieve better search results. Nils Reimers et al. introduced a model based on BERT, called Sentence-BERT (SBERT), which encodes sentences into embeddings that allow for similarity comparisons[2]. SBERT employs a Siamese Network and Triplet Network architecture to embed sentences into a fixed-size vector space, enabling cosine similarity calculations to assess sentence relevance. Compared

to traditional BERT, SBERT is far more efficient for large-scale corpora, reducing BERT's 65-hour computation time to just 5 seconds. Additionally, SBERT supports unsupervised tasks like clustering and semantic retrieval.

In the sentence-transformers framework, Bi-Encoder and Cross-Encoder are two approaches for sentence similarity. Bi-Encoder independently encodes two sentences (e.g., with BERT) to generate embeddings, and then measures their similarity via cosine similarity. This approach is fast and well-suited for large-scale similarity search. By contrast, the Cross-Encoder processes both sentences jointly within a single model, directly outputting a similarity score. While Cross-Encoder offers higher accuracy, it lacks the efficiency of independent embeddings and is thus more suited to small-scale, high-precision comparisons.

In information retrieval, we often combine Bi-Encoders and Cross-Encoders to form a recall-rerank model, applied in semantic search scenarios. This improves both the speed and accuracy of finding relevant information. Figure 1 shows the pipeline on how Bi-encoders and Cross-Encoders are combined for handling search queries or question-answering.

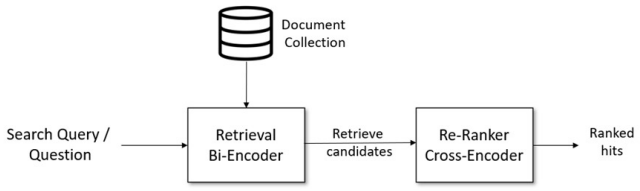


**Figure 1: Bi-Encoder/Cross-Encoder pipeline**

The authors demonstrated the application of this search model architecture in Similar Publication Retrieval[3]. They used all publications from the EMNLP 2016-2018 conferences as a corpus, with titles and abstracts of newer papers as queries, using the SPECTER model to find related publications. This setup closely resembles our project's application scenario.

**Experimental method**

In this project, several retrieval models and techniques were employed to enhance the retrieval results. Initially, different parameters for Jelinek-Mercer and Dirichlet smoothing were tested, and the best-performing parameters were selected. We then adopted a pre-trained Transformer model, which was fine-tuned to make it more suitable for the specific retrieval tasks in our project. The retrieval process followed a Retrieve & Re-rank approach, using a Bi-Encoder for initial filtering and a Cross-Encoder for re-ranking.

The pre-trained model transforms sentences into vectors, embedding the semantics within these vectors. This allows the measurement of semantic similarity between the query and the document using cosine similarity, thereby improving retrieval accuracy. Moreover, fine-tuning the pre-trained model helps adapt it to the specific domain of our dataset, enhancing its effectiveness for targeted retrieval tasks. To further improve the model, we added noise to the queries and corresponding documents during the training process, which served as augmented data for fine-tuning both the Bi-Encoder and Cross-Encoder.

## 3.2 N-gram Tokenization

N-gram Tokenization is widely used in Information Retrieval and Natural Language Processing [4], especially suitable for handling fuzzy queries, typos, or partial keyword matches. To implement N-gram Tokenization, we designed a custom N-gram tokenizer and applied it to a custom analyser within Elastic-search. We set the minimal gram to 3, which allows matching on small fragments, and set the maximal gram to 10, which helps cover longer phases. We also specified token characters as 'letter' and 'digit', which is important for biomedical text with identifiers and numeric strings. During the search, user queries are matched based on N-gram tokens. By using N-gram Tokenization, this setup allows the search system to find relevant documents even when users input fragments or slightly varied keywords. Figure 2 shows the flow of the N-gram Tokenizer. Begin with an input of the document, then we deployed N-gram Tokenizer to create sequences of tokens with lengths ranging from minimal gram to maximal gram. Afterwards, documents are retrieved based on the matching of N-gram tokens with the query. The final output represents retrieved documents based on N-gram Tokenization.

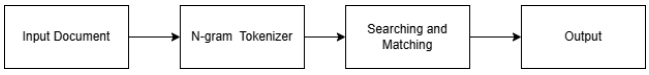


**Figure 2: Flow of N-gram Tokenizer**

# 4 Results

The results show that our Bi-Encoder and Cross-Encoder retrieval approach outperformed traditional models by a significant margin. We take the Jelinek-Mercer language model as the baseline model. This model performed the best in the assignment. Our best retrieval performance was achieved using a language model based on the Jelinek-Mercer similarity metric. In the Jelinek-Mercer method, the parameter $\lambda$ determines the weighting between the document model and the collection model. We performed a grid search over $\lambda$ from 0.1 to 0.7 and determined that the optimal parameter was 0.2. We used the language model with this parameter as our baseline model.

**Metric analysis**

In our analysis, we use the following metrics to perform analysis on the performance of the model:

- Mean success_at_k: measures the proportion of queries with at least one relevant document in the top k results, indicating whether the system meets users' needs within the first few results—a scenario especially relevant when users prioritize the top results.
- Mean r_precision: The precision when the number of returned results equals the number of relevant documents, evaluating the system's accuracy in returning a quantity of results equal to the relevant documents.

- Mean precision_at_k: represents the proportion of relevant documents within the top k results, assessing the relevance of the system's results at positions most likely to capture users' attention.
- Mean precision_at_recall_k: Measures precision at various recall levels, helping us understand whether the system maintains high precision at high recall or has strong relevance at low recall.
- Mean average_precision (MAP): The mean of average precision (AP) across queries, aggregating precision at different recall points to reflect the system's balance in relevance, recall, and accuracy.

When evaluating information retrieval systems using metrics like mean success_at_k, mean precision_at_k, and MAP, we sometimes encounter a limitation: certain queries have very few relevant documents (e.g., only 2 or 3). In these cases, if the chosen k value exceeds the number of relevant documents for a given query, these metrics cannot reach 100%, even if the system retrieves all relevant documents. For example, for a query with only 3 relevant documents, precision_at_5 or precision_at_10 can only reach a maximum of 60% or 30%, respectively, which does not fully reflect the system's actual performance. Therefore, a metric not reaching 100% does not necessarily indicate poor system performance; rather, it may be constrained by the limited number of relevant documents for certain queries.

**N-gram Tokenization**

Initially, we applied a tokenization strategy to process the document text and evaluated the retrieval method using the baseline model. The baseline model outperforms the N-gram Tokenization with the baseline model across all key performance indicators. The Jelinek-Mercer similarity model uses probability smoothing across terms to calculate similarity. When applying n-gram tokenization, the increased granularity (from words to small substrings) causes the distribution of terms to become sparser, making it challenging for the model to accurately represent the semantic meaning. The Jelinek-Mercer model may suffer from lower accuracy when processing highly granular n-gram tokens. This sparsity can distort the similarity calculations, especially in a complex text, where splitting words into n-grams may lose key context. N-gram tokenization generates numerous substrings, many of which do not represent meaningful words. This can generate lots of noise during retrieval, as these tokens may not have any useful semantic information, but still affect the similarity calculation. This added noise affects the similarity calculations in the Jelinek-Mercer model, leading to decreased precision and relevance.

**SBERT models**

We then introduced different SBERT models as the Bi encoder and Cross encoder.We used two retrieval models, called SBERTv1 and SBERTv2. In SBERTv1, we used msmarco-MiniLM-L-6-v3 as the bi-encoder and ms-marco-TinyBERT-L-2-v2 as the cross-encoder. These models had shorter embedding computation times, and Table 1 shows the retrieval results for SBERTv1. We then used higher-accuracy models, msmarco-distilbert-base-v4 and ms-marco-MiniLM-L-6-v2, as the bi-encoder and cross encoder, respectively. The results are recorded in the SBERTv2 column of Table 1. Although SBERTv2 took longer to compute, its accuracy significantly improved.

Subsequently, we adopted a recall-then-rerank vectorized retrieval model, which significantly improved retrieval performance compared to the baseline model. This two-stage retrieval framework uses a bi-encoder to first filter candidate documents relevant to the query text, followed by a cross-encoder for fine-grained ranking, achieving a balance between speed and precision. The bi-encoder quickly encodes the query and documents, compressing the vast document set into a relevant candidate pool, ensuring a high recall rate while enhancing retrieval speed. The cross-encoder then performs precise calculations on each candidate, capturing more complex semantic relationships and optimizing ranking accuracy to ensure that users see the most relevant results.

In terms of retrieval model choice, we opted against traditional probabilistic, language, and Boolean models. Instead, we used a pre-trained model, fine-tuned to ensure that the SBERT model accurately captures the retrieval semantics of our task and embeds both query and document text into vectors. Compared to SBERTv1, SBERTv2 uses the more complex and parameter-rich msmarco-distilbert-base-v4 as the bi-encoder and ms-marco-MiniLM-L-6-v2 as the cross-encoder, which enhances its ability to capture subtle semantic relationships between the query and documents, thereby improving vector representation quality.

From a metric standpoint, *mean_success_at_k* scores were high, indicating that, in this search scenario, most of the top k results from SBERTv2 were relevant documents. The mean_precision_at_k and mean_precision metrics showed significant improvement compared to the baseline, though they were around 0.5. This does not imply poor retrieval performance but rather reflects a limitation due to the finite number of relevant documents for certain queries.

**Table 1: Comparison of Baseline and N-gram Tokenization Performance**

| Performance Metrics | | | | |
|---|---|---|---|---|
| **Metric** | **Baseline** | **N-gram** | **SBERTv1** | **SBERTv2** |
| mean success_at_1 | 0.1316 | 0.0526 | 0.3684 | 0.4737 |
| mean success_at_5 | 0.2895 | 0.1579 | 0.6579 | 0.7895 |
| mean success_at_10 | 0.2895 | 0.2105 | 0.7105 | 0.8947 |
| mean r_precision | 0.1474 | 0.0335 | 0.2680 | 0.4411 |
| mean precision_at_1 | 0.1316 | 0.0526 | 0.3684 | 0.4737 |
| mean precision_at_5 | 0.0895 | 0.0368 | 0.1842 | 0.2474 |
| mean precision_at_10 | 0.0524 | 0.0289 | 0.1211 | 0.1684 |
| mean precision_at_recall_00 | 0.2092 | 0.1029 | 0.4693 | 0.6163 |
| mean precision_at_recall_01 | 0.2004 | 0.0996 | 0.4778 | 0.6171 |
| mean precision_at_recall_02 | 0.2004 | 0.0924 | 0.4576 | 0.6004 |
| mean precision_at_recall_03 | 0.1829 | 0.0884 | 0.4084 | 0.5661 |
| mean precision_at_recall_04 | 0.1776 | 0.0612 | 0.3788 | 0.5263 |
| mean precision_at_recall_05 | 0.1776 | 0.0528 | 0.3791 | 0.5237 |
| mean average_precision | 0.1842 | 0.0788 | 0.3248 | 0.5249 |

## 5 Limitation and Future work

There are several limitations that could potentially affect the retrieval performance. First, the similarity calculation method we adopted is cosine similarity. Although cosine similarity is widely used in vectorized text retrieval, an alternative approach such as the dot product method could provide advantages by considering

both the magnitude and direction of vectors, making the calculation more efficient and potentially enhancing retrieval speed.

Another limitation is the length restriction imposed by SBERT on input text. SBERT models typically ignore text exceeding 512 tokens, which can result in the loss of semantic information when processing lengthy scientific documents. This limitation may impact the model's ability to capture the full context and nuance of longer texts, potentially leading to decreased retrieval accuracy in cases where critical information appears beyond the model's token limit.

Additionally, the choice of pre-trained model could limit the semantic alignment with our target retrieval domain. In our study, we used MSMARCO models, which are trained on a large general-purpose information retrieval corpus. While this dataset provides robust training data for retrieval tasks, it is not specifically tailored to scientific text. Consequently, the model's semantic embeddings may lack alignment with the specific language and structure used in scientific literature. In future work, we propose that using SciBERT or another scientific domain-focused model and fine-tuning it for our retrieval context could yield better alignment with the specific semantic needs of scientific text retrieval, potentially leading to improved retrieval precision.

## 6 Conclusion

We have utilised multiple techniques aiming to improve the accuracy of text-based information retrieval. We compared the traditional approach like n-gram with the modern approach like Sentence-BERT. Our research demonstrates that using SBERT for retrieval significantly enhances performance, as the result indicated by the mean success rate, mean precision and mean average precision. We can conclude that employing deep learning techniques can dramatically improve the retrieval accuracy and efficiency of IR systems when dealing with large-scale data collections.

## References

[1] TREC, "Trec genomics - background," Ohsu.edu, 2024. [Online]. Available: https://dmice.ohsu.edu/trec-gen/background.html

[2] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[3] "Retrieve & re-rank — sentence transformers documentation," Sbert.net, 2024. [Online]. Available: https://sbert.net/examples/applications/retrieve_rerank/README.html

[4] P. McNamee and J. Mayfield, "N-gram morphemes for retrieval," *CLEF (Working Notes)*, 01 2007.

## Acknowledgement

This report uses AI-powered tools for report structural and language improvement.