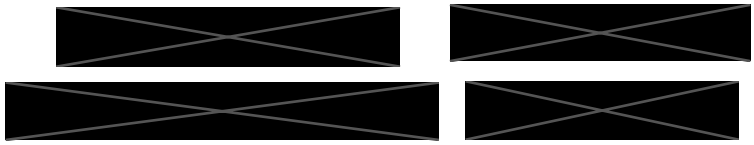
# TDDE09
# Beyond BM25: A Dense Retrieval Approach Using Sentence-BERT and FAISS




## Abstract

We present a semester project in NLP focused on passage retrieval using Sentence-BERT (sBERT) embeddings combined with the FAISS indexing. Using the MS MARCO passage retrieval dataset, we evaluated a dense retrieval system, comparing its performance against a sparse retrieval baseline system (BM25). Our system targets efficient retrieval of relevant passages, employing sBERT for semantic embeddings and FAISS for scalable nearest-neighbor search using cosine similarity. We evaluated the quality of the ranking primarily using MRR@10.

We compared different embedding models, *all-MiniLM-L6-v2* and *msmarco-MiniLM-L6-v3*, using FAISS IndexFlatIP. The results show that *msmarco-MiniLM-L6-v3* outperforms *all-MiniLM-L6-v2*, achieving an MRR@10 score of 31.8 compared to 30.3, demonstrating the advantage of using a fine-tuned embedding model. Both models significantly surpass the traditional BM25 baseline, which achieves an MRR@100 score of 16.7 on the same dataset.

These findings show the effectiveness of dense retrieval methods over traditional lexical-based retrieval and demonstrate the impact of fine-tuning in improving passage ranking performance.


## 1 Introduction

Information retrieval (IR) systems aim to retrieve relevant documents or passages given a user query, a task central to search engines and question answering. Traditional sparse retrievers like BM25 rely on lexical matching but struggle with semantic understanding. Dense retrieval methods, leveraging neural embeddings such as Sentence-BERT (sBERT) (Reimers and Gurevych, 2019), address this by encoding queries and passages into a latent representation within a shared semantic space. These latent representations can be efficiently indexed with Facebook AI Similarity Search (FAISS) (Johnson et al., 2019; Ghadekar et al., 2023). FAISS allows for fast nearest-neighbor search, which is crucial for large-scale applications.

Our project develops a passage retrieval system using FAISS-indexed sBERT embeddings, and we evaluated it on the MS MARCO dataset (Bajaj et al., 2016). We focus on retrieving short text passages where we compare our dense retrieval method with the BM25 baseline. We use two embedding models. One that has been pre-trained on multiple datasets, *all-MiniLM-L6-v2*, and a fine-tuned version, *msmarco-MiniLM-L6-v3*), that has specifically been optimized for the MS MARCO dataset. The MS MARCO dataset, with its large-scale, human-labeled queries and sparse labels, provides a robust testbed. We assess performance primarily with mean reciprocal rank (MRR), and demonstrate the impact of fine-tuning on retrieval quality.

## 2 Related work

Dense retrieval has emerged as a powerful paradigm in IR. Nogueira and Cho (2019) achieved an MRR@10 of 36.5 on MS MARCO using BERT re-ranking, while Zhuang et al. (2021) explored typos-aware dense retrieval with comparable metrics. Using the same dataset, Nogueira et al. (2019) achieved an MRR@10 of 18.4 using a tuned version of BM25. Whereas the official MS MARCO Team achieved an MRR@10 of 16.7 using BM25.

Efficiency-focused studies, such as Rahman et al. (2024), benchmark FAISS and ANNOY for scalable search, adaptable to text retrieval. Surveys like Chen et al. (2024) contextualize neural IR within MS MARCO and TREC frameworks. Our work extends these efforts by integrating sBERT with FAISS, emphasizing a simple fine-tuned embeddings system for passage retrieval.

## 3 Methodology

We used the MS MARCO (Microsoft Machine Reading Comprehension) dataset, specifically tailored for the passage-ranking task (Bajaj et al., 2016). MS MARCO is a large-scale, real-world dataset designed for question answering and passage retrieval, comprising queries from Bing search logs and human-annotated passages from web documents. For this study, we used the entire collection of passages, which consists of approximately 8.8 million rows. As for the evaluation queries we selected the dev subset that contains approximately 59,000 queries, each paired with a single positive passage. This subset simulates sparse-label scenarios, balancing computational feasibility with real-world challenges.

The MS MARCO passage-ranking dataset includes:

- **Queries**: Short, natural-language questions or phrases (e.g., "What is the capital of France?").
- **Passages**: Text snippets (60–100 words, average 73 tokens) (Bajaj et al., 2016).
- **Annotations**: Each query is paired with a relevant passage.

This structure supports evaluating both dense and sparse retrieval methods.

### 3.1 Model Architecture

We employ sBERT to generate 384-dimensional embeddings for queries and passages (Reimers and Gurevych, 2019). Two models are evaluated:

- **all-MiniLM-L6-v2**: A general-purpose model optimized for semantic similarity.
- **msmarco-MiniLM-L6-v3**: A fine-tuned model tailored to the MS MARCO dataset.

Both use a Siamese network architecture based on a pre-trained sentence transformer.

For indexing, we use FAISS with IndexFlatIP, which performs exact nearest-neighbor search using cosine similarity when the embeddings are L2 normalized:

$$\text{similarity}(q, p_i) = \frac{q \cdot p_i}{\|q\|_2 \|p_i\|_2}$$

where $q$ and $p_i$ are query and passage embeddings, respectively.

### 3.2 Evaluation Metrics

We evaluate ranking quality using:

- **MRR@10**: Mean Reciprocal Rank of the first relevant result in the top 10
- **Efficiency**: measured by queries per second (QPS)

## 4 Experiments

### 4.1 Setup

Passages are encoded with the sBERT models (*all-MiniLM-L6-v2* and *msmarco-MiniLM-L6-v3*), indexed with FAISS IndexFlatIP, and queried with evaluation embeddings. The top 100 results are retrieved and evaluated against MS MARCO's positive labels.

We used a single NVIDIA GeForce RTX 2060 SUPER GPU with 8GB of memory for all experiments. The system is equipped with a Ryzen 5600X CPU and 16GB of RAM. The GPU is used for encoding the passages and queries, while the CPU is used for indexing and searching. We provide the limitations of our hardware in **??**.

### 4.2 Results

Table 1 summarizes the findings.

**Ranking Quality**: The fine-tuned embeddings model, *msmarco-MiniLM-L6-v3*, achieves a higher MRR@10 score (31.8) than the general embeddings model, *all-MiniLM-L6-v2*, which achieved an MRR@10 score of 30.3. Both of the sBERT models significantly surpasses BM25 that achieved a score of 16.7.

**Efficiency**: As for efficiency, we measure the time taken to retrieve the top 100 passages for each query. The BM25 method achieves a speed of 18 queries per second (QPS), while the FAISS-based methods achieve 29 QPS for *all-MiniLM-L6-v2* and 20 QPS for *msmarco-MiniLM-L6-v3*.

**Comparison to Prior Work**: Comparing our results with prior work on the MS MARCO passage ranking task we find that, more advanced two-stage methods like in Nogueira and Cho (2019) and Zhuang et al. (2021) achieve slightly better results than our dense retrieval methods. For example:

Using BM25 + BERT base model gives an MRR@10 score of 34.7, while BM25 + BERT large model gives 36.5

This is likely due to the use of a two-stage approach, where ranking is more precise. In our case, we only used a single stage of retrieval.

| Method | MRR@10 | QPS (≈) |
|---|---|---|
| BM25 | 16.7 | 18 |
| BM25 tuned | 18.4 | 18 |
| all-MiniLM-L6-v2 + FAISS | 30.3 | 29 |
| msmarco-MiniLM-L6-v3 + FAISS | 31.8 | 20 |

Table 1: Retrieval performance on MS MARCO passage ranking dataset.

### 4.3 Discussion

The fine-tuned embedding model, *msmarco-MiniLM-L6-v3*, outperforms the general-purpose model *all-MiniLM-L6-v2*, highlighting the benefit of dataset-specific tuning. Both exceed BM25, confirming dense retrieval's semantic advantage. However, our MRR@10 (31.8) falls short of Nogueira and Cho (2019)'s 36.5, likely due to the simplicity of our approach and the absence of a two-stage retrieval process. However, given the method we used and the small embedding dimension (384), we achieved a good balance between efficiency and accuracy without the need for using a large language model.

The efficiency of our dense retrieval system, with FAISS achieving 29 QPS, demonstrates its scalability. The BM25 method, while being slower at 18 QPS, it lacks the semantic relevance compared to dense retrieval. However, it is important to note that QPS may vary based on hardware. Even though we used IndexFlatIP for exact search, which is slower than other FAISS indexing methods, it provides high accuracy. The choice of FAISS IndexFlatIP was made to ensure the highest accuracy in retrieval, as we were primarily focused on evaluating the performance of our dense retrieval system.

FAISS's exact search ensures high accuracy, though at a slight efficiency cost compared to other indexing methods like ANNOY. Future work could explore other indexing techniques and implement a multi-stage approach. Furthermore, we could also explore the use of larger embedding dimensions, such as 768 or 1024, to improve the performance of our dense retrieval system.

## 5 Examine

As the team member responsible for researching the BM25 baseline, state-of-the-art models on the MS MARCO leaderboard, and preprocessing the dataset, this project allowed me to critically assess my prior experience against the course content and additional readings. Below, I reflect on each area, highlighting similarities and differences in my understanding before and after the project, evaluating its adequacy, and noting how it enhanced my grasp of NLP concepts.

### 5.1 MS MARCO Dataset

Before the course, I had little experience with large-scale IR datasets, mostly working with small, curated datasets for simple text processing. During the course, we learned tokenization across different NLP datasets, which introduced me to handling data variety, but it didn't cover other preprocessing steps like deduplication or formatting, nor did it tackle massive dataset scales in depth. This left me unprepared for the 8.8 million passages and 59,000 dev queries of MS MARCO (Bajaj et al., 2016). I had to figure out preprocessing tokenization, deduplication, and formatting on my own, relying on the dataset's documentation. The project showed me how crucial cleaning sparse, noisy web data is for IR, deepening my knowledge in data preprocessing. My initial understanding was limited by the course's lack of hands-on preprocessing beyond tokenization, but the project filled that gap. Looking back, I see I could've optimized further, like using parallel processing, with more time.

### 5.2 BM25 Baseline

My prior knowledge of BM25 was derived from introductory IR tutorials, viewing it as a simple term frequency method. The course did not cover BM25, prompting me to investigate Robertson and Zaragoza (2009) to understand its probabilistic foundation as a sparse retrieval baseline. Implementing BM25 as our baseline (MRR@10 of 16.7) aligned with my expectations of lexical matching, but its semantic limitations became evident when compared to dense retrieval. The course IR metrics (for example, MRR@10) equipped me to evaluate it, yet I underestimated the tuning potential (for example, adjustments k and b), as seen in Nogueira et al. (2019)'s 18.4 MRR@10. My understanding was sufficient for a baseline, but lacked depth in

optimization, which the project clarified. Today, we explore parameter tuning to narrow the gap with dense methods.

### 5.3 State of the Art Model

Researching the MS MARCO leaderboard exposed me to state-of-the-art approaches, like Nogueira and Cho (2019)'s BM25+BERT re-ranking (MRR@10 of 36.5). Before this, I assumed single-stage retrieval was sufficient; the course did not cover multistage pipelines, leaving me unaware of the precision boost of the re-ranking. This gap limited our project to a single-stage dense approach, and while my research was adequate to contextualize our results, I lacked the practical knowledge to implement such systems. The project sharpened my awareness of advanced IR, and today, I would integrate a re-ranking layer (e.g., BERT-large) to approach leaderboard scores.

### 5.4 MRR@10 Metrics

My exposure to evaluation metrics such as MRR@10 began with this project, since TDDE09 discussed general NLP evaluation but not IR-specific ranking metrics. Researching MRR@10, I learned that it measures ranking quality by averaging the reciprocal rank of the first relevant result across queries, capped at the top 10 results. Formally, for a set of queries $Q$, the reciprocal rank for a single query $q_i$ is:

$$\text{RR}(q_i) = \frac{1}{\text{rank}_i},$$

where $\text{rank}_i$ is the position of the first relevant passage in the ranked list (or 0 if none of the top 10). The mean reciprocal rank is then:

$$\text{MRR@10} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \text{RR}(q_i).$$

Applying this to our MS MARCO dev set, BM25's MRR@10 of 16.7 reflected its lower relevance ranking compared to sBERT+FAISS's 31.8, aligning with my research on semantic vs. lexical methods. My initial understanding was limited, assuming precision alone sufficed, but the project clarified MRR's focus on ranking position, enhancing my evaluation skills. Today, I'd complement MRR@10 with recall or NDCG for a fuller picture.

Overall, the project significantly enhanced my understanding of IR metrics and retrieval paradigms from the course, despite its GPT focus. My preprocessing and BERT knowledge grew through necessity, but gaps in optimization and multistage design persisted. With current insights, I'd prioritize data efficiency, BM25 tuning, larger sBERT embeddings, and a re-ranking stage.

## 6 Articulation

This project leveraged several TDDE09 course concepts and skills relevant to my responsibilities. The course provided a general foundation in evaluation metrics for NLP tasks, which proved essential for assessing BM25 (16.7) and sBERT+FAISS (31.8) performance, offering a clear success measure. Python and Pytorch programming skills from course exercises enabled dataset preprocessing and model integration, while general NLP foundations (e.g., tokenization, embeddings) supported sBERT research and implementation. The discussion of cosine similarity, though tied to GPT contexts, directly applied to FAISS's IndexFlatIP search, aligning queries and passages semantically.

During the course, we covered tokenization for NLP datasets, but I struggled preprocessing MS MARCO's 8.8 million noisy passages. The course skipped deduplication and formatting, focusing on GPT tasks with clean inputs. I relied on trial-and-error, revealing a gap in my skills that slowed me down and highlighted the need for broader preprocessing training beyond tokenization. Reading Bajaj et al. (2016) clarified dataset quirks, but a course foundation in this area would have accelerated my work. The project thus underscored the limits of my preparation while reinforcing evaluation and embedding concepts, pushing me to adapt and expand my skills.

## 7 Conclusion

We presented a dense retrieval system using Sentence-BERT embeddings and FAISS indexing, evaluated on the MS MARCO passage ranking dataset. Our approach outperformed the BM25 baseline, achieving an MRR@10 of 31.8 with the fine-tuned *msmarco-MiniLM-L6-v3* model. The results highlight the effectiveness of dense retrieval methods over traditional lexical-based approaches and the impact of fine-tuning on passage ranking performance. We demonstrated that our dense retrieval system is efficient, achieving 29 queries per second (QPS) with FAISS. The findings suggest that dense retrieval methods can significantly

enhance passage retrieval tasks, especially when leveraging fine-tuned embeddings. However, we acknowledge that our results are not as high as those achieved by more complex two-stage methods. Future work could explore the integration of larger embedding dimensions and multi-stage retrieval processes to further enhance performance.

## References


Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. In *Proceedings of the NIPS Workshop on Cognitive Computation*.

Wei Chen, Zhicheng Dou, and Ji-Rong Wen. 2024. Large Language Models for Information Retrieval: A Survey. *arXiv preprint arXiv:2403.01276*.

Premanand P. Ghadekar, Sahil Mohite, Omkar More, Praiwal Patil, Sayantika, and Shubham Mangrule. 2023. Sentence meaning similarity detector using faiss. In *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–6.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale Similarity Search with GPUs. *IEEE Transactions on Big Data*.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *Preprint*, arXiv:1904.08375.

Md. Mostafizur Rahman, Maliha Tasnim Mishu, and M. Sohel Rahman. 2024. Optimizing Approximate Nearest Neighbor Search in High-Dimensional Spaces. *Journal of Data Science*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Stephen E. Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*.

Shuai Zhuang, Xiubo Geng, and Tao Qin. 2021. Dealing with Typos in Dense Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.