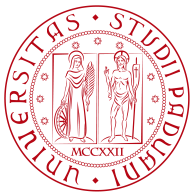

University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Data Science

Exploring Multilingual Embeddings for Italian Semantic Search: A Pretrained and Fine-tuned Approach

Supervisor
Professor Alberto Testolin
University of Padova

Co-supervisor
Dosso Dennis
Siav S.p.A.

Master Candidate
Nicolò Rinaldi
2091699

Academic Year
2023-2024

I would like to thank my advisor, Professor Alberto Testolin, for his invaluable support in the composition and realization of this thesis.

I am also grateful to Siav, the company that allowed me to undertake my internship with them, and in particular to the employees I worked closely with, Gioele Perin and Dennis Dosso, who guided and supported me in the creation and development of this thesis.

I extend my sincere thanks to my family, who enabled me to embark on and complete this journey.

I want to thank my hometown friends, who were always there for me during this period, who encouraged and motivated me to stay committed and work hard to achieve this goal.

Finally, I wish to express my gratitude to my university colleagues, who became dear friends. Together, we consistently supported and helped each other, whether during intense study sessions or lighter moments in between.

Lastly, I want to thank myself for having persevered through challenging times and having reached this milestone, growing both as a person and as a student.

Abstract

This thesis explores the application of multilingual embedding models to the semantic search for the Italian language, a critical step toward integrating these technologies into Retrieval-Augmented Generation (RAG) frameworks. The work leverages state-of-the-art pre-trained and fine-tuned neural models to address the challenges of document retrieval in both symmetric and asymmetric contexts. Using a variety of datasets, including translated corpora for validation, the study evaluates models such as LaBSE, multilingual-e5-large, and bge-m3 for their ability to generate meaningful embeddings and improve retrieval performance. Performance for the asymmetric framework is assessed using nDCG@10.

The fine-tuning phase, where the model is modified by inserting an adapter on top of the query embedding for each pre-trained model, demonstrates the adaptability of two of the aforementioned models to Italian-language tasks. The statistical significance has been assessed with the Wilcoxon signed-rank test, which results in a p-value < 0.001 for multilingual-e5-large and bge-m3, beating their counterpart without the addition of the adapter.

One of our models, multilingual-e5-large with the linear adapter, achieved superior results to proprietary solutions like OpenAI's text-embedding-3-small. The significance has been assessed with the same statistical test, resulting in a p-value < 0.05.

Additionally, our solution demonstrated substantial improvements in document retrieval times, reducing latency of OpenAI's model with our best-performing model of one order of magnitude. Furthermore, the training process is cost-effective and the lightweight design of the model enables it to operate on local hardware.

Contents

Listing of figures

Listing of tables

Listing of acronyms

AI ..... Artificial Intelligence

IR ..... Information Retrieval

NLP ..... Natural Language Proccessing

LLM ..... Large Language Models

RAG ..... Retrieval Augmented Generation

TF-IDF ..... Term Frequency-Inverse Document Frequency

BM25 ..... Best Match 25

LaBSE ..... Language-Agnostic BERT Sentence Embedding

QQP ..... Quora Question Pairs

MS MARCO ... Microsoft MAchine Reading Comprehension

mMARCO ..... multilingual version of MS MARCO

nDCG@k ..... Normalized Discounted Cumulative Gain using only the first k documents

DCG@k ..... Discounted Cumulative Gain using only the first k documents

IDCG@k ..... Ideal Discounted Cumulative Gain using only the first k documents

MRR@k ..... Mean Reciprocal Rank using only the first k documents

MAP@k ..... Mean Average Precision using only the first k documents

AP@k ..... Average Precision using only the first k documents

API ..... Application Programming Interface

BERT ..... Bidirectional Encoder Representations from Transformers

GPT ..... Generative Pre-trained Transformer

LLaMA ..... Large Language Model Meta AI

PaLM . . . . . Pathways Language Model

CPU . . . . . Central Processing Unit

GPU . . . . . Graphics Processing Unit

LPU . . . . . Language Processing Unit

AWS . . . . . Amazon Web Services

ROUGE . . . . . Recall-Oriented Understudy for Gisting Evaluation

BLUE . . . . . Bilingual Evaluation Understudy

MLM . . . . . Masked Language Modeling

NSP . . . . . Next Sentence Prediction

XLM . . . . . Cross-lingual Language Models

RoBERTa . . . . . Robustly Optimized BERT Pretraining Approach

MTEB . . . . . Massive Text Embedding Benchmark

BEIR . . . . . Benchmarking-IR

RDBMS . . . . . Relational Database Management System

ORM . . . . . Object-Relational Mapping

HNSW . . . . . Hierarchical Navigable Small World

1 Introduction

Siav S.p.A.* is an Italian software and IT services company based in Rubano, Padua, specializing in dematerialization, document management, and digital processes. Founded in 1989 by Alfieri Voltan, Siav has established itself as a leader in the **Enterprise Content Management** sector, providing innovative solutions for electronic document management, workflow management, and digital preservation.

With a commitment to improving operational efficiency, Siav offers a comprehensive suite of software solutions and **Business Process Outsourcing** services that enable organizations to manage the entire document lifecycle. This includes in-house, outsourcing, or hybrid approaches tailored to meet the specific needs of both public and private sectors.

Siav operates multiple offices in Italy, including locations in Milan, Genoa, Bologna, and Rome, and has expanded its reach to international markets such as Switzerland and Romania. The company boasts over 4,000 installations across various industries, serving notable clients such as the Bank of Italy and San Raffaele Hospital.

In addition to its robust service offerings, Siav dedicates more than 12% of its revenue to research and development, underscoring its commitment to innovation and excellence and highlighting it through partnerships with leading academic institutions. This focus on innovation is exemplified by projects involving advanced technologies such as process mining and social network analysis.

*https://www.siav.com/it/chi-siamo/azienda/

One of the projects that Siav is working on involves developing a system capable of accurately answering customer questions in natural language. Providing natural-language answers to queries offers several significant benefits to the company using this system.

First, the ease of interaction is greatly improved. When users ask a question in plain language, they expect a direct response that addresses their query, rather than having to search through multiple documents or links to find relevant information. This streamlined approach accelerates decision making, which is particularly crucial in environments where time is very important, such as business or emergency settings.

Second, there is a notable reduction of information overload. Simply presenting users with documents or links to resources can often overwhelm them, especially when the query is complex. Natural language responses, on the other hand, condense the necessary information into a clear and concise answer. This approach ensures that users are only provided with what is relevant and manageable, preventing the confusion that can arise from excessive or unrelated data.

Additionally, natural-language answers are more accessible to a broader range of people. Users who may not be familiar with technical language, complex document structures, or search syntax can still effectively use the system, making it more inclusive and user-friendly.

The main idea is to use a Large Language Model (LLM) as a kind of search engine, bypassing the need to examine various websites ranked by traditional algorithms. Instead, the goal is for the model to generate a natural language response directly. To achieve this, the most straightforward approach might be to rely on the assumption that the LLM has already learned the necessary information during its training. By simply inputting a query describing the topic of interest, one could hope the model would infer the answer. Although this may seem like a viable alternative, there exists a phenomenon known as "hallucination", where the model produces responses that are incorrect or entirely fabricated. This occurs because LLMs generate output based on patterns learned during training. This also means that the information learned by an LLM is fixed to the time of its training data. For instance, if we were to ask it questions about current events, it would either be unable to provide an answer or would likely generate fabricated information. As a result, they may often provide confident yet inaccurate responses, particularly when the query falls outside their knowledge base.

A potential solution to this problem is Retrieval-Augmented Generation (RAG). It combines the natural language generation capabilities of LLMs with external retrieval systems, offering a solution to some of the key limitations inherent in LLMs, particularly the issues of hallucination, incomplete or outdated information, and limited domain-specific knowledge.

In an RAG system, instead of relying solely on the pre-trained LLM parameters, which includes knowledge only from its training data, the system retrieves the most relevant documents from a predefined external database. These documents are then passed as input to the LLM, ensuring that the model has access to up-to-date or domain-specific information to generate a response. This allows RAG systems to provide an additional layer of transparency by allowing users to trace the sources of the generated responses. When the LLM generates an answer based on retrieved documents, those documents can be presented alongside the response, enabling users to verify the information and build trust in the system.

Despite the promising premises, the process presents numerous challenges. For example, integrating the retrieved context into the generative process is one of them. LLMs must effectively process the retrieved documents and understand their relevance to the query.

Another example is that RAG systems inherently introduce additional computational overhead due to the retrieval step. Retrieving documents from a large-scale vector database in real-time can increase response latency, making it challenging to deploy RAG in scenarios requiring instantaneous responses. This problem is particularly relevant with the integration of API calls and not suited vector database's settings or excessively large embeddings.

Another crucial point is that the effectiveness of a RAG system is highly dependent on the quality of the retrieval mechanism. Irrelevant or low-quality retrieved documents can mislead the LLM, resulting in inaccurate or incoherent responses. Ensuring that the retrieval component consistently identifies the most relevant information from large-scale databases is a significant challenge, especially when dealing with ambiguous or complex queries. This can be achieved using traditional count-based methods, such as BM25. However, these approaches often retrieve documents that are not entirely relevant to the query, leading to suboptimal results.

During my internship at Siav, I worked on a project aimed at developing a semantic search system tailored to the Italian language, with the goal of future integration of this system into an RAG application. The primary objective was to replace paid embeddings with privately fine-tuned or pre-trained models, offering a more cost-effective and scalable alternative for semantic representation, looking both at computational overhead and performance. This project spanned six months, encompassing the complete design, coding, and implementation of the semantic search framework presented in this Master's thesis.

The thesis examines semantic search systems in two different frameworks: symmetric and asymmetric retrieval frameworks. Symmetric search, which focuses on queries and documents of similar length, leverages the Quora Question Pairs (QQP) triplets dataset, translated into

Italian, to validate the generalizability of models such as LaBSE, multilingual-e5-large, and bge-m3. Conversely, asymmetric search, emphasizing short queries against longer documents, relies on the multilingual MS MARCO (Microsoft MAchine Reading Comprehension, in short, mMARCO) dataset for training and the DBpedia-Entity-v2 dataset for validation and testing, also adapted for Italian. This framework is the main focus of this work, where we shaped fine-tuning processes to build our proprietary model. Looking to recent research, we decided to work with an approach that involves the integration of adapters on top of query embeddings, with the goal of efficiently mapping short queries to a shared semantic space with documents that are more semantically rich than very short queries. While other methods like LoRA were explored, computational constraints led to a focus only on the adapter methodology.

We then compare all the models cited above with OpenAI's text embedder-3-small, the model currently used by the company, to evaluate the performance of the main selected metric, nDCG @ 10, which is a standard metric used to evaluate the quality of ranking retrieval systems. A higher nDCG@10 score indicates that the most relevant documents are ranked closer to the top, reflecting better retrieval performance.

Methodologies include the development of a relational database to handle complex datasets and facilitate efficient model testing. Embedding-based search algorithms, such as Hierarchical Navigable Small World (HNSW) and exact search, were used to rank documents based on semantic similarity through the Weaviate vector database. The thesis also details the deployment environment, leveraging AWS infrastructure for fast and high-performance training. This provides an opportunity to explore an alternative data type, known as bfloat16, which accelerates training by a factor of three compared to the standard type.

The results highlight the adaptability and statistical effectiveness of fine-tuned multilingual models for Italian semantic search for two of three models in our test set, achieving an improvement in nDCG@10 scores from 32.78 to 38.37 for LaBSE and from 45.23 to 53.89 for multilingual-e5-large. This is done by adding only a linear adapter on top of the query embedding.

Further investigations were conducted on the third model, bge-m3, to understand why it did not perform as expected. First, we attempted to enhance the adapter's representational capacity by using two layers instead of one and incorporating a non-linear activation function in between. Next, we reduced the window size and divided the text into smaller chunks. However, none of these adjustments ultimately led to any improvement.

This research also achieves a notable result in comparison with OpenAI's text-embedding-3-small model: we manage to statistically outperform it (though not by a large margin) with

the multilingual-e5-large model with the linear adapter fine-tuned by us. This, along with an improvement in search time by an order of magnitude (615 ms vs 47 ms), is because our model uses embeddings of smaller size and does not require an API call.

In summary, by focusing on adapter-based fine-tuning and leveraging advanced vector search techniques, this research offers a scalable and high-performance alternative to proprietary solutions for the Italian language, emphasizing the feasibility of adopting custom embeddings in real-world applications.

2 Related Works

2.1 LLM

Large Language Models (LLMs) have become a crucial point of research in NLP due to their remarkable capabilities in understanding and generating human language. The evolution of language models has seen a significant shift from traditional statistical approaches to neural network-based methodologies. The early models relied on n-grams and other statistical methods, but the introduction of deep learning, particularly the Transformer architecture [4], completely changed the field. Transformer-based models, such as BERT [5] and GPT [6], have allowed the training of models on enormous datasets, significantly improving performance in various NLP tasks.

Several prominent families of LLMs have emerged, each contributing uniquely to the landscape. The Generative Pre-trained Transformer (GPT), developed by OpenAI*, includes models such as GPT-1 [6], GPT-2 [7], GPT-3 [8], and GPT-4 [9], designed for general-purpose language understanding and generation. These models leverage unsupervised pretraining followed by fine-tuning for specific tasks, in particular, Reinforcement Learning with Human Feedback (RLHF). Another notable family is LLaMA (Large Language Model Meta AI), developed by Meta that has 3 main releases: LLaMA[10], LLaMA 2 [11], and LLaMA 3 [12]. The last one, in particular, focuses on efficiency and accessibility in training large models while

*https://openai.com/

emphasizing multi-modal capabilities that integrate both text and image data. Google's Pathways Language Model (PaLM) [13] is recognized for its advanced reasoning capabilities and improved multilingual support, showcasing the trend towards more capable and versatile LLMs.

Despite their success, LLMs pose significant efficiency challenges as a result of their high resource requirements. Recent research categorizes efforts to enhance efficiency into three main areas: model-centric approaches that optimize architecture or training methods to reduce computational costs, for example new data types and quantization; data-centric approaches that involve curating or augmenting training datasets to improve model performance without increasing size; and framework-centric approaches that innovate hardware frameworks to facilitate more efficient training and deployment of LLMs, like LPU from Groq [†].

Evaluation of LLMs is critical to understand their performance in various tasks. Common metrics include perplexity, accuracy on benchmark datasets, and task-specific evaluations. The literature has seen the development of various benchmarks aimed at assessing LLM capabilities in tasks such as summarization, translation, and question answering. For summarization tasks, metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [14] are commonly used. Translation quality is often evaluated using the BLEU (Bilingual Evaluation Understudy) [15] score, which computes matching n-grams between the LLM's predicted translation and a human-produced reference. Question answering capabilities are assessed using metrics such as accuracy, F1-score, and Macro F1-score.

Despite significant advancements in the field, several challenges remain. Issues related to bias and fairness are of main importance, as addressing inherent biases in training data is crucial for ethical deployment. In addition, enhancing the interpretability of LLM decision-making processes is necessary to gain trust in AI applications. Sustainability is also a growing concern; developing methods to reduce the environmental impact of training large models is increasingly important as their popularity continues to rise.

2.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) represents a significant advancement in the capabilities of LLMs by integrating external knowledge sources to enhance the accuracy and relevance of generated content. It was first introduced by Lewis et al. [16] in 2020.

RAG combines the strengths of retrieval systems and generative models, allowing real-time access to up-to-date information from external databases. This integration not only improves

[†] `https://groq.com/`

the factual accuracy of responses, but also enables LLMs to handle knowledge-intensive tasks more effectively. The RAG framework typically consists of three main components, each playing a crucial role in the overall process:

1. **Retrieval**: The retrieval phase involves sourcing relevant documents or data chunks from an external knowledge base based on semantic similarity to a user query. This is achieved through techniques such as vector embeddings, which convert text into numerical representations that facilitate efficient similarity searches. The effectiveness of this phase is **critical**, as it directly impacts the quality of the information available for the subsequent generation step.

2. **Augmentation**: Augmentation refers to the methods used to enhance the integration of the retrieved information with the generated responses. This may involve techniques for re-ranking retrieved documents or compressing context to focus on key details. Effective augmentation ensures that the generated output is coherent and contextually relevant, thereby improving user satisfaction and trust in the system.

3. **Generation**: In this phase, the LLM synthesizes a response using both the original user query and the retrieved information. The model can leverage its inherent knowledge while also grounding its responses in the specific context provided by the retrieved data. This dual approach helps mitigate issues such as hallucination, where the model could otherwise generate inaccurate or irrelevant content.

This first component is the main core of a RAG application and is usually implemented as a semantic search algorithm, the focus of this work. Semantic search is an advanced search methodology that focuses on understanding the meaning and intent behind a user's query rather than simply matching keywords. Unlike traditional keyword-based search, which mainly relies on finding exact word matches within documents, semantic search analyzes concepts, context, and relationships between terms in both the query and the content, making it possible to retrieve more relevant results even if they do not contain exact keyword matches. This is done with the use of embeddings, which are vectors that capture the semantic meaning of a piece of text (e.g. queries and documents). After embedding the query and documents, a similarity metric (like cosine similarity) measures the distance between vectors. Documents with vectors closer to the query vector are recognized as more semantically similar and prioritized in the results. In the end, the relevant documents are ranked and presented based on their semantic similarity score.

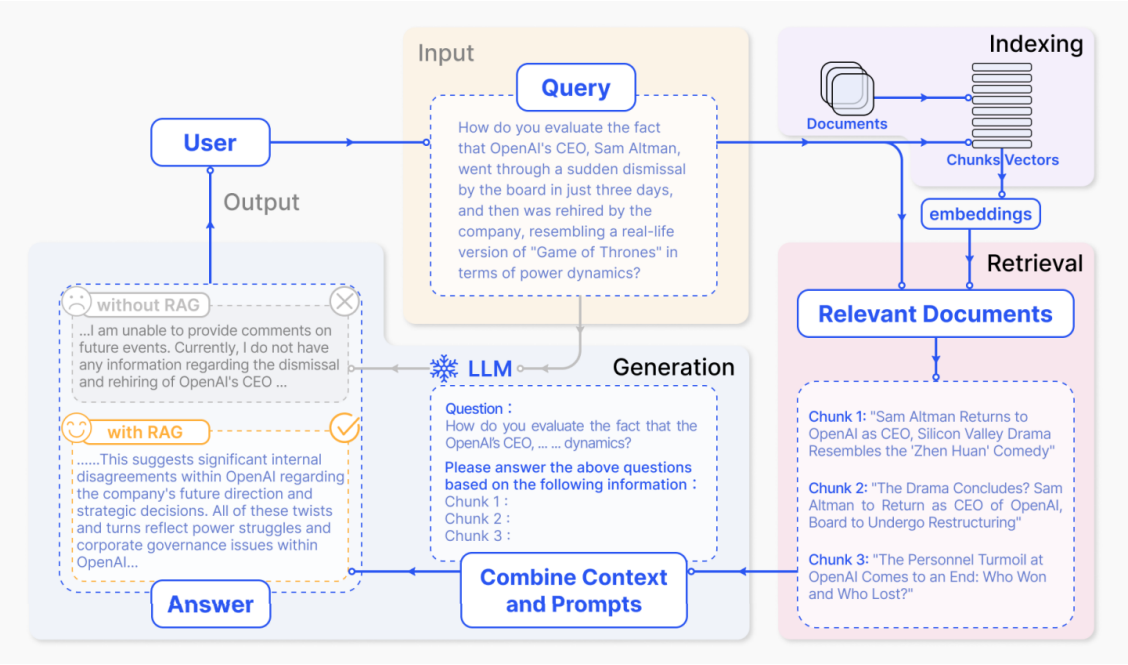

Figure 2.1: A representative instance of the RAG process applied to question answering. It mainly consists of 3 steps. 1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database. 2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity. 3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer. This is taken in the survey by Gao et al. [1].

2.2.1 Asymmetric vs Symmetric Framework

A critical distinction for our setup is symmetric vs. asymmetric semantic search:

- **Symmetric semantic search**: in this framework, the queries and the documents in the corpus are about the same length and have the same amount of content. An example would be searching for similar questions.

- **Asymmetric semantic search**: in this framework the queries are usually short (like a question or some keywords) and the documents we want to find are a longer paragraph answering the input query.

This distinction plays a crucial role in guiding our models and datasets selection, as we will see in chapter 4.

2.3 Embedding Models: an overview

In natural language processing, an embedding model is a type of model that transforms words, phrases, or entire pieces of text into vectors that capture their semantic meaning. These vectors, often referred to as embeddings, represent the relationships between words in a high-dimensional space where similar words are positioned closer together and dissimilar ones farther apart.

We are going to focus on the models that turn text into dense, continuous vectors. Unlike traditional methods such as one-hot encoding, which represents words as large, sparse vectors, dense embeddings provide a more compact and informative way to capture linguistic patterns.

Embedding models allow computers to process text by converting the language into numerical form, making it easier for algorithms to identify and understand relationships between words. The models achieve this by learning patterns from large amounts of text data, allowing them to place semantically related words near each other in the vector space.

More advanced embedding models, such as BERT or GPT, generate context-aware embeddings, meaning that the representation of a word changes depending on the words surrounding it. This allows for a finer understanding of language, as the same word can have different meanings in different contexts.

We need to exploit a little more what is BERT (and its variants) to gain a better understanding of the concept of neural embedding model. BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art model developed by Devlin et al. [5] at Google. It is designed to better understand the context of words in relation to other words in a sentence. The structure of BERT is based on the Transformer architecture introduced by Vaswani et al. [4], particularly its encoder component.

The architecture of BERT consists of multiple layers (12, in the base variant) of encoders stacked on top of each other, which allows it to process and understand complex relationships in language.

The key innovation of BERT lies in its bidirectional nature. Unlike previous models that processed text in one direction (either left-to-right or right-to-left), BERT reads the entire sentence in both directions simultaneously. This allows it to capture the full context of a word by looking at the words both before and after it, improving its ability to understand meaning based on context.

BERT is pre-trained on massive amounts of text using two specific tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In MLM, random words in a sen-

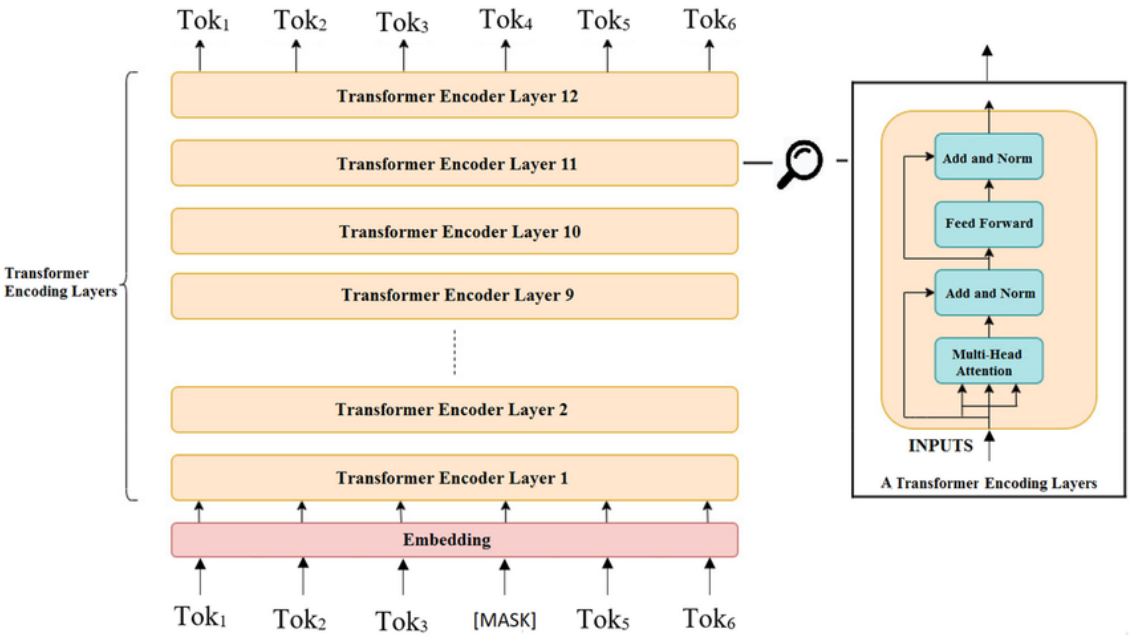

Figure 2.2: BERT model architecture.

tence are masked, and in BERT the masking strategy is to take 15% of the input tokens of each training sequence and divide them into 3 categories:

- 80% are replaced with `[MASK]`;
- 10% are replaced with randomly selected tokens;
- 10% are left unchanged.

These 15% are the tokens that the model tries to predict. In NSP, the model is presented with pairs of sentences and is asked to predict whether each pair consists of an actual pair of adjacent sentences from the training corpus or a pair of unrelated sentences. In BERT, 50% of the training pairs consisted of positive pairs, and in the other 50% the second sentence of a pair was randomly selected from elsewhere in the corpus. To do so, after tokenizing the input, the token `[CLS]` is placed before the input sentence pair, and the token `[SEP]` is placed between the sentences and after the final token of the second sentence. Finally, embeddings representing the first and second segments of the input are added to the word embeddings and positional embeddings to allow the model to more easily distinguish the input sentences.

At the end, each token has its own vector representation, and the whole context embedding is computed with a pooling layer that in the model we are going to see is a mean pooling or a `[CLS]` token embedding extraction, to capture the semantics of each token of text.

One of its variants, namely the XLM-RoBERTa architecture, is a model introduced by Conneau et al. [17] used as pre-trained initialization for two of our embedding models.

RoBERTa is an improved version of BERT, introduced by Liu et al. [18] to address limitations in the original pretraining methodology of BERT. The focus of RoBERTa is on optimizing pre-training strategies rather than altering the underlying architecture of BERT. The key changes are as follows:

- Longer training with larger batch sizes and increased dataset size;
- Dynamic masking, that dynamically generates a new masking pattern for each sequence in every training iteration instead of using a static pattern;
- Removal of Next Sentence Prediction;
- Training on longer sequences (up to 512 tokens).

In the XLM-RoBERTa training procedure, there are new training methods, different from the MLM described above.

Causal Language Modeling (CLM) is a task in which a Transformer-based language model is trained to estimate the probability of a word $P(w_t|w_1, \ldots, w_{t-1}, \theta)$ given all previous words in a sentence. The model learns to predict the next word in a sequence by attending only to the past context, making it well-suited for autoregressive tasks like text generation.

Translation Language Modeling (TLM) is a training objective designed to enhance cross-lingual pretraining by leveraging parallel data (aligned sentence pairs in two languages). It extends the MLM objective by jointly training in bilingual sentence pairs. Words are randomly masked in both the source and target sentences, and the model learns to predict the masked tokens using context from either the same language or its translation.

XLM (Cross-lingual Language Models), first introduced by Lample et al. [19], is an acronym used for Transformer based architecture that is pre-trained using one or both of the two language modeling objectives, Causal Language Modeling and Masked Language Modeling, alongside with Translation Language Modeling.

Putting it all together, XLM-RoBERTa is a model that adopts the same training setup and architecture as RoBERTa, while incorporating the XLM method to enhance its performance, particularly in multilingual contexts.

In models where this structure is present, there is always the "large" version, which includes 24 layers, 1024 embedding dimensions, 16 attention heads, and a Feed-Forward Network with a size of 4096.

2.4 Benchmark: MTEB

The Massive Text Embedding Benchmark (MTEB) is an evaluation framework designed by Muenninghoff et al.[2] to evaluate the performance of text embedding models with different tasks, datasets, and languages. MTEB covers eight different tasks, with the goal of evaluating the generalizability and effectiveness of text embeddings in different applications. Figure 2.3 illustrates an overview of the benchmark.

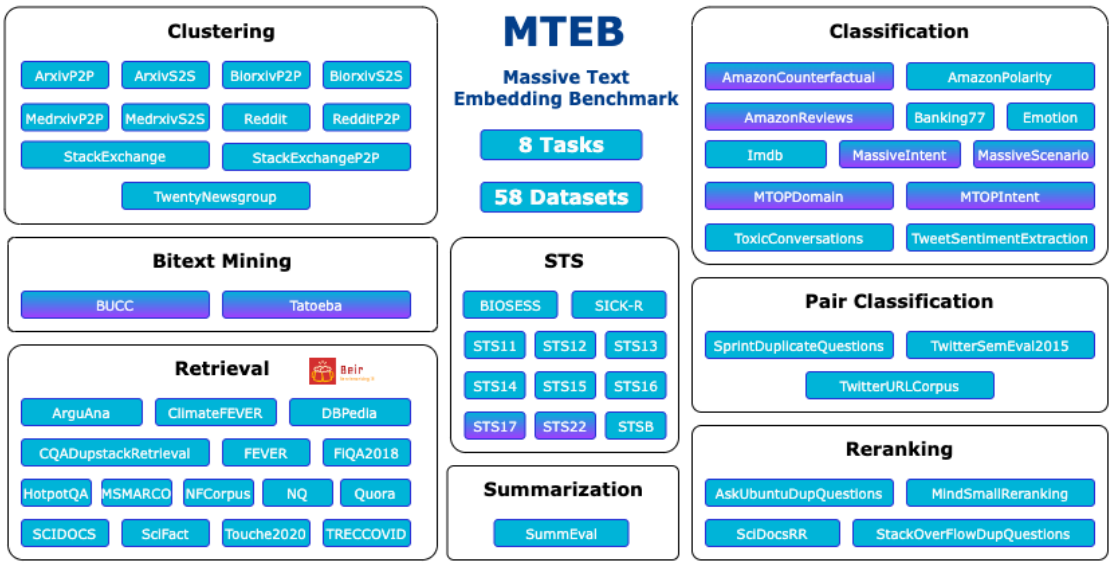

Figure 2.3: Overview of tasks and datasets in MTEB. Multilingual datasets are marked with a purple shade. Taken from Muennighoff et al. [2].

For our use case, we want to focus specifically on one task: document retrieval. The retrieval task is one of the eight tasks in the MTEB benchmark. It involves finding relevant documents from a corpus based on a given set of queries. For this task, each retrieval dataset consists of a corpus of documents, a set of queries, and a mapping of each query to relevant documents in the corpus. The objective is to rank the documents in the corpus according to their relevance to each query.

The evaluation process is structured as follows: first, the embedding model is used to generate embeddings for both the queries and the documents. Then cosine similarity is computed between the embeddings of each query and all documents in the corpus and based on these similarity scores, the documents are ranked for each query.

In our work, we adopt the same metrics as used in the referenced paper, which are as follows:

- **nDCG@k (Normalized Discounted Cumulative Gain)**: This is the main metric used for evaluating retrieval performance in MTEB. nDCG@k measures the quality of the ranking, giving more weight to the positions at the top of the list; the formula is the following:

$$\text{nDCG} = \frac{\text{DCG@}k}{\text{IDCG@}k} \tag{2.1}$$

where

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{\text{rel}(i)}{\log_2(i+1)} \tag{2.2}$$

with rel@i is an indicator function telling us if the i-th document is relevant (and it takes value 1) or not (and it takes value 0) and IDCG@k is the ideal DCG@k, so the one computed when all the relevant documents are retrieved in the first places.

- **MRR@k (Mean Reciprocal Rank)**: It measures the average of the reciprocal ranks of the first relevant document for each query; the formula is the following:

$$\text{MRR@}k = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{\text{rank}(n)} \tag{2.3}$$

where N is the number of queries and rank(n) is the position of the first relevant item for query q in the top-k results.

- **MAP@k (Mean Average Precision)**: This metric calculates the mean of the average precision scores for all queries, where precision is computed at each rank position; the formula is the following:

$$\text{MAP@}k = \frac{1}{N} \sum_{n=1}^{N} \text{AP}_{q_n}\text{@}k \tag{2.4}$$

where N is the number of queries and AP is called Average Precision and it is computed for each query q as follows:

$$\text{AP@}k = \frac{1}{\text{Total number of relevant items for the query } q} \sum_{i=1}^{k} \text{Prec@}i \times \text{rel}(i) \tag{2.5}$$

where rel@i is an indicator function telling us if the i-th document is relevant (and it takes value 1) or not (and it takes value 0).

- **Precision@k**: It measures the proportion of relevant documents in the top-k ranked

results; the formula for one query q is the following:

$$\text{Prec}_q@k = \frac{\text{Number of relevant items within the top-}k}{k} \tag{2.6}$$

Then we compute the mean over all the queries as

$$\text{Prec}@k = \frac{1}{N}\sum_{n=1}^{N}\text{Prec}_{q_n}@k \tag{2.7}$$

where N is the number of queries.

- **Recall@k**: This metric evaluates the ability of the model to retrieve all relevant documents within the top-k results; the formula for one query q is the following:

$$\text{Recall}@k = \frac{\text{Number of relevant items within the top-}k}{\text{Total number of relevant items}} \tag{2.8}$$

Then we compute the mean over all the queries as

$$\text{Recall}@k = \frac{1}{N}\sum_{n=1}^{N}\text{Recall}_{q_n}@k \tag{2.9}$$

where N is the number of queries.

- F_1**@k**: This metric combines the two aspect of precision and recall through the harmonic mean; the formula for one query q is the following:

$$F_{1q}@k = 2 \cdot \frac{\text{Prec}@k \cdot \text{Recall}@k}{\text{Prec}@k + \text{Recall}@k} \tag{2.10}$$

Then we compute the mean over all the queries as

$$F_1@k = \frac{1}{N}\sum_{n=1}^{N}F_{1q_n}@k \tag{2.11}$$

where N is the number of queries.

The retrieval task in MTEB primarily reuses datasets and evaluation protocols from the Benchmarking-IR (BEIR) [20] benchmark, ensuring consistency and comparability with previous work in information retrieval. The nDCG@10 score is particularly emphasized as the main metric for evaluating the retrieval performance in MTEB.

For our work, we took inspiration from the above paper to adapt their method (which works for the English language and a multilingual setting) to the Italian language, selecting $k = 10$ and nDCG@10 as our main metric for the asymmetric framework and MRR@10 for the symmetric one.

3 Models

3.1 Baselines (Frequency Based)

3.1.1 TF-IDF

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, is a widely used statistical measure in NLP and information retrieval that evaluates the importance of a word within a document relative to a collection of documents, known as a corpus. The fundamental principle behind TF-IDF is to quantify how relevant a term is to a specific document while considering its frequency across multiple documents. The idea is to use it as an embedding model by creating a vector of dimension $|V|$ where V is the vocabulary made from all the words inside the corpus documents.

The Term Frequency (TF) component measures how frequently a term t appears in a particular document d. It is calculated by taking the number of times a term appears in the document and dividing it by the total number of terms in that document. This gives a normalized score that reflects the term's importance within the document itself. The formula for $TF_{t,d}$ can be expressed as:

$$TF_{t,d} = \frac{\text{Number of times term } t \text{ appears in the document } d}{\text{Total number of terms in the document } d} = \text{count}(t, d) \tag{3.1}$$

It is common to squash the raw frequency a bit, using the $\log_{10}$ of the frequency instead, resulting in a loss of the amplitude.

$$TF_{t,d} = \log_{10}(\text{count}(t, d) + 1) \tag{3.2}$$

On the other hand, Inverse Document Frequency (IDF) assesses how common or rare a term is across all documents in the corpus. It helps reduce the weight of terms that occur frequently in many documents, thus highlighting terms that are more unique and potentially more informative. IDF_t is calculated using the following formula:

$$IDF_t = \log_{10}\left(\frac{\text{Total number of documents}}{\text{Number of documents containing the term } t}\right) \tag{3.3}$$

The final TF-IDF$_{t,d}$ score for a term is obtained by multiplying its $TF_{t,d}$ and IDF_t values:

$$\text{TF-IDF}_{t,d} = TF_{t,d} \times IDF_t \tag{3.4}$$

A high TF-IDF$_{t,d}$ score indicates that the term t is significant within that specific document d but not common among others, which makes it likely to be crucial for understanding the content of the document. For example, if a document frequently uses a specialized term that rarely appears in other documents, it will have a high TF-IDF score, suggesting its relevance to the topic discussed.

At this point, we know the TF-IDF$_{t,d}$ for each t and d in the corpus, so we can proceed by constructing a vector of dimension $|V|$ and entering the TF-IDF$_{t,d}$ value for each term t in that document d and 0 for each term in the vocabulary that is not in the document d.

We chose TF-IDF because it is a great baseline in NLP, because of its simplicity and the fact that it is count-based. Thus, it does not account for semantic meaning or context beyond word counting, which can lead to suboptimal results for queries and documents that require deeper understanding.

3.1.2 BM25

BM25, or Best Match 25, also known as Okapi BM25, is a variant of the TF-IDF family used in information retrieval systems to assess the relevance of documents to a given search query. Developed in the 1990s, BM25 builds on earlier probabilistic models and introduces several enhancements over traditional methods such as TF-IDF.

The core components of BM25 include Term Frequency (Equation 3.2), Inverse Document Frequency (Equation 3.3), and document length normalization. Term Frequency measures how often a term appears in a document, with BM25 employing a modified version that accounts for diminishing returns, meaning that after a certain point, additional occurrences of the term contribute less to the overall score. This adjustment helps prevent overly high scores for documents that contain many repetitions of a term.

Inverse Document Frequency evaluates the rarity of a term across the entire corpus. It operates on the principle that terms appearing in fewer documents should carry more weight, thus enhancing the relevance of documents containing these rare terms. The formula for IDF (Equation 3.3) is structured to provide higher scores for terms that are less common across documents.

The length of the document is another critical factor in BM25. The algorithm normalizes the scores based on the length of each document compared to the average length of the document in the collection. This normalization ensures that longer documents do not receive an unfair advantage simply due to their size, which could lead to skewed relevance rankings.

BM25 incorporates two tunable parameters: $k1$ and b. The parameter $k1$ adjusts the influence of term frequency, while b controls the extent to which the length of the document affects the score. Typically, $k1$ is set between 1.2 and 2.0, and b is often set around 0.75.

The overall BM25 score for a document concerning a query is computed using the following formula:

$$\text{BM25}(q, d) = \sum_{i=1}^{n} IDF_{q_i} \cdot \frac{TF_{q_i,d}}{k1 \cdot (1 - b + b \cdot \frac{|d|}{|d_{\text{avg}}|}) + TF_{q_i,d}} \tag{3.5}$$

In this equation:

- $TF_{q_i,d}$ is the frequency of term q_i in the document d.
- $|d|$ is the length of the document d.
- d_{avg} is the length of the average document of all the documents in the corpus.
- IDF_{q_i} denotes the Inverse Document Frequency for term q_i.

BM25 has several advantages over its predecessors. It dynamically adjusts rankings based on the distribution of terms within a collection, making it more adaptable to different types of queries and documents. It also performs particularly well with longer queries, effectively addressing issues related to term saturation.

However, BM25 also has limitations. It does not account for semantic meaning or context beyond term matching, which can lead to suboptimal results for queries and documents requiring deeper understanding.

3.2 Neural Embedding Models

As we anticipated in chapter 2, all the neural models we are going to use are based on BERT or one of its numerous variants.

3.2.1 LaBSE

LaBSE * [21], or Language-Agnostic BERT Sentence Embedding, is a sophisticated multilingual sentence embedding model developed by Google AI. It is designed to encode text from 109 different languages in a shared vector space.

Its architecture consists of a 12-layer transformer that leverages a vocabulary size of 500,000 tokens, ensuring comprehensive coverage across the supported languages (including Italian), allowing a window size of 256 tokens for the generation of the embedding. The window size is only of 256 tokens for this model. This is the smallest of the neural embedding models, with a total of 471.517.440 parameters.

The full model architecture is as follows:

```
SentenceTransformer(
  (0): Transformer({'max_seq_length': 256, 'do_lower_case': False})
     with Transformer model: BertModel
  (1): Pooling({'word_embedding_dimension': 768, '
     pooling_mode_cls_token': True, 'pooling_mode_mean_tokens': False,
      'pooling_mode_max_tokens': False, '
     pooling_mode_mean_sqrt_len_tokens': False})
  (2): Dense({'in_features': 768, 'out_features': 768, 'bias': True, '
     activation_function': 'torch.nn.modules.activation.Tanh'})
  (3): Normalize()
)
```

*https://huggingface.co/sentence-transformers/LaBSE

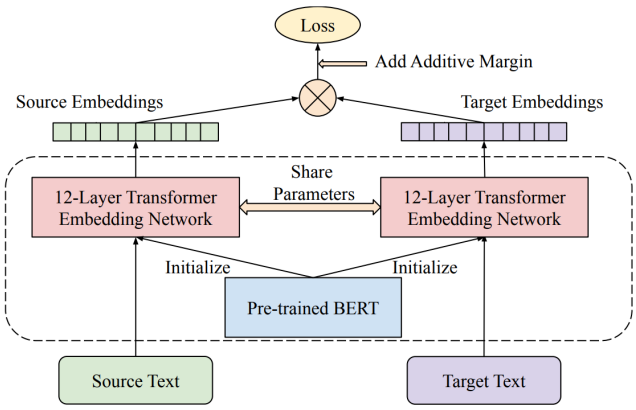

Figure 3.1: LaBSE training structure.

Figure 3.1 illustrates the LaBSE training scheme. This model is trained as a dual encoder with the goal of generating cross-lingual embeddings. It consists of paired encoding networks that independently process the source and target sentences. The resulting embeddings are compared via a scoring function. The model leverages a translation ranking task with in-batch negative sampling and incorporates an additive margin softmax to increase the separation between correct translations and closely related non-translations.

The embedding networks in LaBSE are initialized using pre-trained BERT weights and further trained through two techniques: Masked Language Modeling (MLM) and Translation Language Modeling (TLM). These steps are explained in more detail in section 2.3. The MLM pre-training uses monolingual data from sources like CommonCrawl and Wikipedia, while TLM employs concatenated bilingual translation pairs derived from a bitext mining system.

The embedding related to the `[CLS]` token is used as the embedding of the full text. For computational efficiency, the model uses a dot-product scoring function to compute pairwise similarity scores on the embeddings. The softmax operation is applied to the resulting scores to classify the embeddings.

LaBSE model achieved medium results on benchmarks such as the MTEB (English text embeddings), scoring an nDCG@10 of 45.2.

Despite its capabilities, the model has some limitations. For example, it truncates long texts to a maximum of 256 tokens, which surely affects its performance on longer documents. Additionally, while it performs robustly across many languages, some low-resource languages may experience reduced effectiveness. Moreover, the embedding dimension is 768, which is lower compared to the other model that will be presented later. That could result in a lower represen-

tation capability.

3.2.2 MULTILINGUAL-E5-LARGE

The multilingual-e5-large model [†] [22] represents a multilingual extension of the English e5 model, allowing it to support more than 100 languages. Developed based on the XLM-RoBERTa architecture, it comprises 24 layers and generates embeddings of 1024 size, as described in section 2.3. The window size is of 512 tokens for this model. This is the middle neural embedding model in terms of parameters, with a total of 559.890.432 parameters. The full model architecture is as follows:

```
Model: SentenceTransformer(
  (0): Transformer({'max_seq_length': 512, 'do_lower_case': False})
     with Transformer model: XLMRobertaModel
  (1): Pooling({'word_embedding_dimension': 1024, '
     pooling_mode_cls_token': False, 'pooling_mode_mean_tokens': True,
      'pooling_mode_max_tokens': False, '
     pooling_mode_mean_sqrt_len_tokens': False, '
     pooling_mode_weightedmean_tokens': False, 'pooling_mode_lasttoken
     ': False, 'include_prompt': True})
  (2): Normalize()
)
```

It is trained through a two-stage methodology. First, it undergoes weakly-supervised contrastive pre-training on approximately 1 billion multilingual text pairs from diverse sources (e.g., Wikipedia, mC4, and Reddit). This is followed by supervised fine-tuning on 1.6 million high-quality labeled datasets, incorporating mined hard negatives and knowledge distillation to enhance embedding quality.

The input handling of multilingual-e5-large requires specific prefixes for queries and documents' passages, denoted as "query: " and "passage: ", respectively.

The embedding of the full text is computed with the element-wise mean of all the embeddings of the tokens that compose the text, as we can see in the model structure above.

The model achieved good results on benchmarks such as MTEB (English text embeddings) and MIRACL (multilingual retrieval across 16 languages). In the MTEB, Multilingual-E5-

[†] https://huggingface.co/intfloat/multilingual-e5-large

large scored an nDCG@10 of 61.5. In the MIRACL dev set, this model achieved an average nDCG@10 of 66.5.

Despite its impressive capabilities, the model has some limitations. For example, it truncates long texts to a maximum of 512 tokens, which may affect its performance on longer documents. Additionally, while it performs robustly across many languages, some low-resource languages may experience reduced effectiveness.

3.2.3 BGE-M3

bge-m3[‡] [23], developed by the Beijing Academy of Artificial Intelligence (BAAI), is an advanced sentence transformer model that stands out for its multi-functionality, multi-linguality, and multi-granularity capabilities (the 3 Ms). Developed based on the XLM-RoBERTa architecture adapted by the RetroMAE method (a retrieval-oriented pre-training paradigm based on the Masked Auto-Encoder from Xiao et al. [24]), it comprises 24 layers and generates embeddings of size 1024, as described in section 2.3. One of the standout features of this model is its window size of 8192 tokens, allowing it to accommodate entire documents. This is the largest of the neural embedding models, with a total of 567.754.752 parameters. The full model architecture is as follows:

```
Model: SentenceTransformer(
  (0): Transformer({'max_seq_length': 8192, 'do_lower_case': False})
     with Transformer model: XLMRobertaModel
  (1): Pooling({'word_embedding_dimension': 1024, '
     pooling_mode_cls_token': True, 'pooling_mode_mean_tokens': False,
      'pooling_mode_max_tokens': False, '
     pooling_mode_mean_sqrt_len_tokens': False, '
     pooling_mode_weightedmean_tokens': False, 'pooling_mode_lasttoken
     ': False, 'include_prompt': True})
  (2): Normalize()
)
```

This model is developed to address significant challenges in multi-linguality, retrieval functionalities, and input granularity. By learning a shared semantic space, it enables both monolingual retrieval within individual languages and cross-lingual retrieval between different lan-

[‡] https://huggingface.co/BAAI/bge-m3

guages, supporting more than 100 world languages. This versatility makes BGE-M3 a powerful tool for multilingual and cross-lingual applications.

The model is designed to generate embeddings that are not restricted to a single retrieval approach. Instead, it supports multiple retrieval paradigms, including dense retrieval, sparse retrieval, and multi-vector retrieval. The `[CLS]` token embedding is optimized for dense retrieval, whereas embeddings derived from other tokens enable sparse and multi-vector retrieval functionalities. We decided to stick to the default one in the sentence-transformers library, using the embedding related to the `[CLS]` token as the embedding of the full input text (as we can see in the model architecture above).

Another notable feature of bge-m3 is its ability to handle inputs of varying length. From short texts such as sentences and passages to long documents containing up to 8,192 tokens, the model effectively processes input across this broad spectrum.

The bge-m3 training process incorporates several innovative techniques to enhance the quality of the embeddings. One key advancement is the introduction of a self-knowledge distillation framework. This approach jointly learns and reinforces multiple retrieval functionalities by integrating relevance signals from dense, sparse, and multi-vector retrieval methods. These signals act as teacher guidance, enabling the model to improve performance through an ensemble learning approach.

To further optimize training, the model employs a high-performance batching strategy that facilitates large batch sizes, contributing significantly to the discriminative power of the embeddings. In addition, the training process benefits from extensive and high-quality data curation. This includes leveraging unsupervised data from vast multilingual corpora, integrating related supervised datasets, and synthesizing data to address gaps in scarce training samples. Each of these data sources complements the others and is applied at different training stages, ensuring a robust training process.

The model achieved remarkable results on benchmarks such as MTEB (English text embeddings) and MIRACL (multilingual retrieval across 16 languages). In the MIRACL dev set, this model achieved an average nDCG@10 of 71.5.

3.2.4 TEXT-EMBEDDING-3-SMALL

The text-embedding-3-small[§] model is an embedding model from OpenAI, designed to enhance the performance of text representation tasks while being cost-effective..

[§] `https://openai.com/index/new-embedding-models-and-api-updates/`

Compared to its predecessor, the text-embedding-ada-002 model, text-embedding-3-small shows significant performance enhancements. The average score on the multi-language retrieval benchmark (MIRACL) has risen from 31.4% to 44.0%, while the average score on English tasks (MTEB) has increased from 61.0% to 62.3%.

The price for using text-embedding-3-small has been reduced significantly (by five times compared to the previous model), making it more accessible to developers. The cost is low: $0.00002 per 1,000 tokens.

The model produces embeddings with a size of 1536 dimensions, allowing for a rich representation of the text's semantic content.

This is the model Siav is trying to replace, since the embedding dimension is big and it is not optimal for our semantic search task. Furthermore, the cost is low but not zero, and some lighter models can run locally (depending on the number of documents requiring embeddings for integration into the system) to avoid also the cost of virtual machines like Amazon Web Services (AWS).

4 Datasets

4.1 Symmetric Dataset: QQP triplets

The QQP triplets dataset [25] is primarily focused on the subtask of semantic similarity classification. It contains data in English, and the dataset is available under the MIT license, which is an open-source license that allows for a wide range of uses. The dataset is provided by the hugging-face website (April 2024 version*). QQP triplets are built on another dataset called Quora Question Pairs (QQP), consists of more than 400,000 question pairs, and each question pair is annotated with a binary value indicating whether the two questions are paraphrased of each other. The result is a dataset of 101762 samples, where each instance is composed by 3 elements:

- a query, that is used as anchor,
- a positive sentence, that has the same meaning of the query,
- a list of negative sentences, that is composed by sentences with different meanings with respect to the query.

An example is presented in the following. Since I have a bachelor's degree in mathematics, I thought that this would be the best example to report.

*`https://huggingface.co/datasets/embedding-data/QQP_triplets`

```
{
"query": "How can you overcome phobia of mathematics?",
"pos": ["Can I overcome a phobia of mathematics?" ],
"neg": ["What are some mind blowing uses of mathematics?",
        "How do I fall in love with mathematics?",
        "How do you fall in love with mathematics?",
        "Why should I study mathematics?",
        "What are the taboos, if any, in mathematics?",
        "How do I truly understand mathematics?",
        "Is mathematics meaningless?",
        "Can you suck at math and be smart?",
        "How can an individual become an expert in mathematics?",
        "When I look at some math Olympiad problems I feel really
           intimidated. I only know how to solve basic problems,
           nothing special, but I like math very much. Is there any
           hope for me to become a mathematician?",
        "How would you revolutionize the teaching of mathematics to
           children?",
        "How can I be an expert in mathematics?",
        "Is mathematics becoming less relevant?",
        "Why do people think that girls can't be as good in
           mathematics?",
        "What are some unsolved problems in mathematics?",
        "Does being good at mathematics make you intelligent?",
        "Is mathematics always necessary in doing good science?",
        "Is it possible for someone to have academic interests in
           mathematics but not physics?",
        "Can I hate math but love science?",
        "Why are 'bright ideas' ridiculed by those who know how to
           manipulate mathematics? Do they want to build authority/
           monopoly of mathematics?",
        "Are girls really bad at mathematics?",
        "Is mathematics another language?",
```

```
        "I love science, but I hate maths. Is that bad?",
        "What can you do with a mathematics degree?",
        "Why do people find mathematics difficult?",
        "What are good ways to mentally grasp foreign mathematical
           concepts?",
        "I struggle with math and science and it makes me feel stupid.
            What can I do to get better?",
        "What are some mind-blowing facts about mathematics?",
        "Does being good at mathematics make you intelligent in other
           subjects?",
        "How do you become a prodigy in math?" ]
}
```

We can see from these examples that the negative sentences are instead more similar to hard-negative sentences. Hard negatives are examples that are similar to positive samples, but belong to a different class or category. These samples make it difficult for the model to distinguish them from positive examples, making them "hard" to classify correctly. In this case, all sentences refer to mathematics in a certain way, but have different meanings.

The goal of this dataset is to test the models we choose for the symmetric semantic search. To the best of our knowledge, this is one of the best datasets for semantic similarity for the presence of hard negatives instead of classical negatives.

Since we are interested in the application of embeddings for the Italian language, we translated the whole dataset to Italian using an open source transformer-based model made by the Language Technology Research Group at the University of Helsinki specifically trained to translate from English to Italian, namely opus-mt-en-it[†] [26] [27].

To test our models, we decided to use the split 70/10/20: 70% of the sample for a possible training set, 10% for validation, and 20% for the test set.

4.2 Asymmetric Training Dataset: unicamp-dl/mmarco

The unicamp-dl/mmarco[‡] dataset is a multilingual version of the Microsoft MS MARCO passage ranking dataset [28], created by researchers at the University of Campinas (Unicamp) in

[†] https://huggingface.co/Helsinki-NLP/opus-mt-en-it
[‡] https://github.com/unicamp-dl/mMARCO

Brazil[29]. This dataset is composed of 14 different languages, but we are interested only in the Italian part. Each language is composed of about 39 million samples. Here is an example from the Italian mMARCO.

```
{'query': "quale frutto è originario dell'australia",
'pos': ["Passiflora herbertiana. Un raro frutto della passione
   originario dell'Australia. I frutti sono a buccia verde, a
   polpa bianca, con una valutazione commestibile sconosciuta.
   Alcune fonti elencano il frutto come commestibile, dolce e
   gustoso, mentre altre elencano i frutti come amari e non
   commestibili.assiflora herbertiana. Un raro frutto della
   passione originario dell'Australia. I frutti sono a buccia
   verde, a polpa bianca, con una valutazione commestibile
   sconosciuta. Alcune fonti elencano il frutto come commestibile,
    dolce e gustoso, mentre altri elencano i frutti come amari e
   non commestibili."],
'neg': ["La noce di cola è il frutto dell'albero di cola, un
   genere (Cola) di alberi originari delle foreste pluviali
   tropicali dell'Africa."]}
```

The structure is the same as the QQP triplets dataset described in section 4.1, but with a significant difference: each sample has only one negative.

The goal of this dataset is to implement the fine-tuning of the models we choose for the asymmetric semantic search. To the best of our knowledge it is one of the best datasets for semantic similarity for the presence of hard negatives instead of classical negatives. We decided to use this dataset for the asymmetric semantic search section of this work only as training set.

4.3 Asymmetric Validation/Test Dataset: Dbpedia-Entity-v2

DBpedia-Entity-v2[§], as explained by Hasibi et al. [3], is a standard test collection for entity search over the DBpedia knowledge base and the dataset is available under the MIT license. The dataset is provided by the hugging-face website (April 2024 version) [¶]. DBpedia is a com-

[§] https://github.com/iai-group/DBpedia-Entity/

[¶] https://huggingface.co/datasets/mteb/dbpedia

munity effort, where a set of rules are collaboratively created to extract structured information from Wikipedia. DBpedia-Entity-v2 is designed to evaluate retrieval systems that return a ranked list of entities in response to a free text user query. It was released in 2017 and is a result of a collaborative effort between the IAI[‖] group of the University of Stavanger, the Norwegian University of Science and Technology, Wayne State University, and Carnegie Mellon University.

The dataset is divided into 3 parts:

- corpus;
- queries;
- qrels.

Let us first talk about the corpus. This is composed of about 4.6M documents coming from the DBpedia collection, and the structure is the following: (id, title, text). I report an example below.

```
{"_id": "<dbpedia:Animalia_(book)>",
 "title": "Animalia (book)",
 "text": "Animalia is an illustrated children's book by Graeme Base.
    It was originally published in 1986, followed by a tenth
    anniversary edition in 1996, and a 25th anniversary edition in 201
    2. Over three million copies have been sold. A special numbered
    and signed anniversary edition was also published in 1996, with an
     embossed gold jacket."}
```

The queries of the collection are organized into four categories:

- **SemSearch ES**: these are short and ambiguous queries that search for a particular entity. An example can be "Harry Potter movie". These are taken from the ad-hoc entity search task of the Semantic Search Challenge series[30][31]
- **INEX-LD**: these are keyword queries, much more like traditional Information Retrieval. An example can be "Eiffel", which can refer to either the Italian music group (Eiffel 65) or the Eiffel tower located in Paris, France. These are taken from the ad-hoc search task in the INEX 2012 Linked Data track [32]

[‖] https://iai.group/

- **List Search**: these are made to find a particular list of entities. An example can be "Winners of the ACM Athena award". These are taken from the list search task of the 2011 Semantic Search Challenge (SemSearch LS) [30], from the INEX 2009 Entity Ranking track (INEX-XER) [33], and from the Related Entity Finding task on the TREC 2009 Entity track [34].
- **QALD-2**: these are classical questions in natural language. An example can be "Which professional surfers were born on the Philippines?". These are taken from the question answering challenge on linked data [35]

This leads to a total of 467 queries, following a similar structure to the corpus, but now without the title, i.e. (id, text). An example is presented in the following.

```
{"_id": "INEX_LD-20120112",
 "text": "vietnam war facts"}
```

It can be seen that the *_id* field contains the name of the category to which the query belongs. This will be useful in our models.

The dataset comes with about 49.2k relevance judgments (which we will call "qrels") that are organized in the following way: (query-id, corpus-id, score). The score takes value in 0, 1, 2 with the following meanings:

- Highly relevant (2): the entity is a direct answer to the query (i.e. the entity should be among the top answers).
- Relevant (1): the entity helps one to find the answer to the query (i.e. the entity can be shown as an answer to the query, but not among the top results).
- Irrelevant (0): the entity has no relation to the intent of the query (i.e. the entity should not be shown as an answer)

For the sake of our models, we decide to deal with highly relevant and relevant documents as if they were the same. To do so, we assign the value of 1 to both relevant classes and keep the zero value to the irrelevant documents. Our choice has been made for future work in the context of RAG.

We decided to use this dataset for the asymmetric semantic search section of this work only as a validation and test set. Since we are interested in the application of embeddings for the Italian language, we translated the entire dataset (corpus and query texts) into Italian using an open source transformer-based model made by the Language Technology Research Group

Category	#queries	Type	R_1	R_2
SemSearch ES	113	named entities	12.5	3.0
INEX-LD	99	keyword queries	23.5	9.2
ListSearch	115	list of entities	18.1	12.7
QALD-2	140	NL questions	28.4	29.8
Total	467		21.0	14.7

Table 4.1: Query categories in Dbpedia-Entity-v2. R_1 and R_2 refer to the average number of relevant and highly relevant entries per query, respectively. This table has been taken from Hasibi et al.[3]

of the University of Helsinki specifically trained to translate from English to Italian, namely opus-mt-en-it** [26] [27].

Sometime, this model had some trouble translating. This happens in some different cases:

1. The number of tokens exceeds the window of the model of 512 tokens (typically this case where the lists of a certain topic); in this case, we tried to split the text by its punctuation marks. If this is still not working, we save the original documents as an error document and proceed with the translation in a second moment.

2. Handling characters coming from different languages results in really bad translations, such as

```
Mores (in genere pronunciato / &lt; m2 &gt; m2 &gt; m2 &gt;
m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt;
m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt;
m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &gt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &gt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
m2 &lt; m2 &lt; m2 &gt; m2 &lt; m2 &lt; m2 &lt; m2 t;
m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt; m2 &lt;
```

**https://huggingface.co/Helsinki-NLP/opus-mt-en-it

3. Some documents were just translated as:

```
== Note ==== Altri progetti ==== Collegamenti esterni ==* Sito ufficiale
```

   with no evident pattern in these cases.

We decided to address this problem by deploying a Google Translate API where we spent the free trial of 300$ to correct the errors made by the original model coming from Helsinki University.

5 Methodologies

5.1 Methodologies for the symmetric framework

Since the symmetric part is not the main focus of our work, we decided to just test the models without fine-tuning them. This choice has been made for one main reason: budget. Considering that the company focus is to build an RAG application where the documents can last pages and we expect the query to be much shorter, it makes no sense to use time and resources to try to fine-tune the model for the symmetric dataset.

However, we used this part of the work to investigate the ability of our models to create a good semantic space from queries of the QQP triplets dataset (as seen in section 4.1), which are very similar to each other in terms of topic but far in terms of semantic meaning.

Because of the structure of the dataset we decided to count as relevant document only the positive query and consider all the other documents as irrelevant. This leads to the following choice of evaluation metrics:

- **Recall@k**: that tells us if the only relevant document has been retrieved in the first k retrieved documents.
- **MRR@k**: that tells us the reciprocal of the retrieved position of the right documents and zero otherwise.

5.2 Methodologies for the asymmetric framework

5.2.1 Virtual machine with AWS

The company used Amazon Web Services (AWS) to develop all its applications, so they give me the possibility to use a virtual machine to have more computational resources. In this service there are a lot of different choices: we decided to go with an Ubuntu based machine with a GPU called g5.xlarge with 24GB of vRAM and a pre-made deep learning setting to be able to track the GPU usage with the terminal command `nvidia-smi -l 1` that automatically updates every second. In this way we were able to select the max batch size possible for every model to have the best possible estimation of the gradient. The choice of this particular GPU has been made with respect to the use of bfloat16 datatype. It was originally developed by Google and is called the "Brain Floating Point Format". The name comes from "Google Brain", which is an artificial intelligence research group at Google where the idea for this format was conceived. The bfloat16 format uses 16 bit to represent a number (as in float16) but maintaining the original range from float32. This is done by truncate the float32 to use only 7 bits for the decimal places, allows for fast conversion to and from a float32, at the expense of precision but gaining in speed. Figure 5.1 shown the differences between the three data types just described.

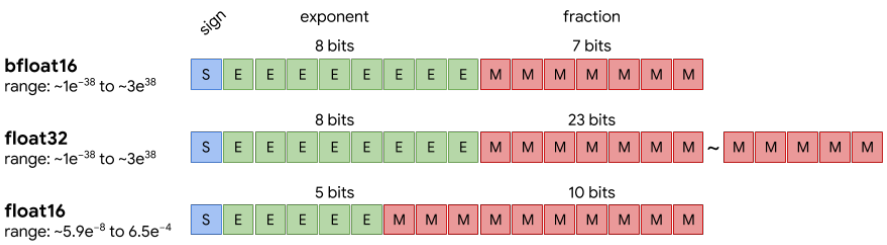

Figure 5.1: Comparison between the bits used for each data type: bfloat16 (bfloat16), double-precision floating-point (float32) and single-precision floating-point (float16).

This datatype is supported only on some type of GPU, like the NVIDIA A10 ones adopted in the g5 instances by AWS. This gives us a 3 times faster training only by adding a line of command in our code when computing the gradients. We also tried to compile the model using `toch.compile()` but we could not get the command to cooperate with models from sentence-transformers library, so we decided to give up on that upgrade, which ideally would have brought about a further improvement in terms of training time.

5.2.2 Fine tuning

Initially the idea was to fine-tune in 3 ways:

- Add an adapter only on top if the query embedding [1], with the idea of letting the model learn to map shorter documents (only the query) near the embeddings of longer documents, to create a unique semantic space where both the query and the documents lay close to each other.
- using LoRA (Low Rank Adaptation) from Hu et al.[36] to mimic the full fine-tuning of the model without losing all the pre-trained initialization and use way fewer parameters.
- Both previous works combined to see how they cooperate each other and try to further improve performance.

We initially chose Low-Rank Adaptation (LoRA) [36] since this fine-tuning method provides an efficient method for adjusting pre-trained models, thereby reducing the high computational costs typically associated with traditional fine-tuning approaches. It is based on the concept of matrix decomposition, which addresses the observation that while the weight matrices in LLMs are typically high-rank, the updates required for fine-tuning can often be represented as low-rank matrices. By reducing the rank of these matrices, the number of parameters requiring training is significantly reduced, which in turn lowers computational costs.

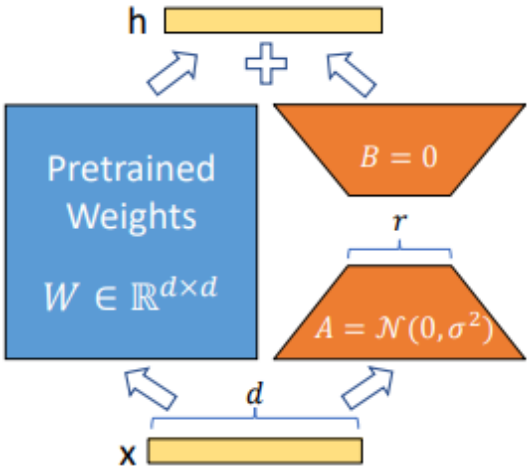

Figure 5.2: LoRA's new parameters visualization.

More than that, LoRA operates by freezing the pretrained weights of the model, allowing for low-rank matrices to be injected into specific layers. This preserves the original knowledge of the model while allowing it to adapt to new tasks. This makes it more resource-efficient.

LoRA decomposes the update matrix ΔW into two smaller matrices, A and B, such that A is of size $d \times r$ and B is of size $r \times d$. The variable r, which represents the rank, is typically much smaller than d, resulting in a significant reduction in the number of parameters requiring adjustment, down to $2dr$, particularly useful when $r << d$ (and the rank is typically chosen as a hyperparameter, so we just pick some low r values, e.g. 4, 8, 16,...).

During training, the outputs are computed using both the original pre-trained weights and the low-rank matrices. The model then combines these components to produce a final output, allowing the model to learn effectively with reduced computational resources.

In the end, we just save the weight of A and B and modify the pre-trained model parameter one time just before use. This allows us to save in memory very few parameters, meaning less memory requirements.

We exploited the peft * library from huggingface to fine-tune with LoRA adapters but the computational burden was too heavy and we decided to drop it; more than this the library was very instable to some longer documents, and sometimes the GPU crashed during the use of the bigger model, even with 24GB vRAM.

These considerations lead to the use only of the first method, the one related to adding an adapter on top of the query embedding, as seen in the survey by Gao et al.[1]. As an architectural consideration, to ensure consistency with the pre-trained models, we added a normalization layer after the adapter. A significant advantage of using only the adapter is that we need to store only its parameters. This results in minimal memory usage, as the number of parameters is limited to

$$\text{embedding\_dimension} \times (\text{embedding\_dimension} + 1)$$

for the linear adapter and

$$\text{inner\_dimension} \times (\text{embedding\_dimension} + 1) + \text{embedding\_dimension} \times (\text{inner\_dimension} + 1)$$

for the non-linear one.

For the largest model tested, namely bge-m3, the simpler linear adapter on top of the query was not working as expected. We thought that this could be caused by two things:

- too big chunks;
- too little representation power by the adapter.

So we tried both by decreasing the chunk size of the biggest model to only 512 tokens or using a non-linear adapter with inner dimension as a new hyperparameter.

Talking about the evaluation metrics, we stick to those cited in section 2.4.

*https://github.com/huggingface/peft

5.2.3 Relational Database: dbpedia_ita

Looking at the structure of the dataset, we decide to create a Relational Database containing the translated dataset of Dbpedia-Entity-v2 (section 4.3) and add some useful tables to deal with the window sizes of the embedding models described in chapter 3.

We used the Python library called SQLmodel to interact and create our database, which is an Object-Relational Mapping (ORM). An ORM is a programming technique that promotes the integration of software systems adhering to the object-oriented programming paradigm with Relational Database Management System (RDBMS) systems. We decided to use the SQLite format to be able to store the file `dbpedia_ita.db` locally and to do versioning during the process to avoid starting from zero each time and save time. The structure of the different tables of the database can be seen in Figure 5.3.

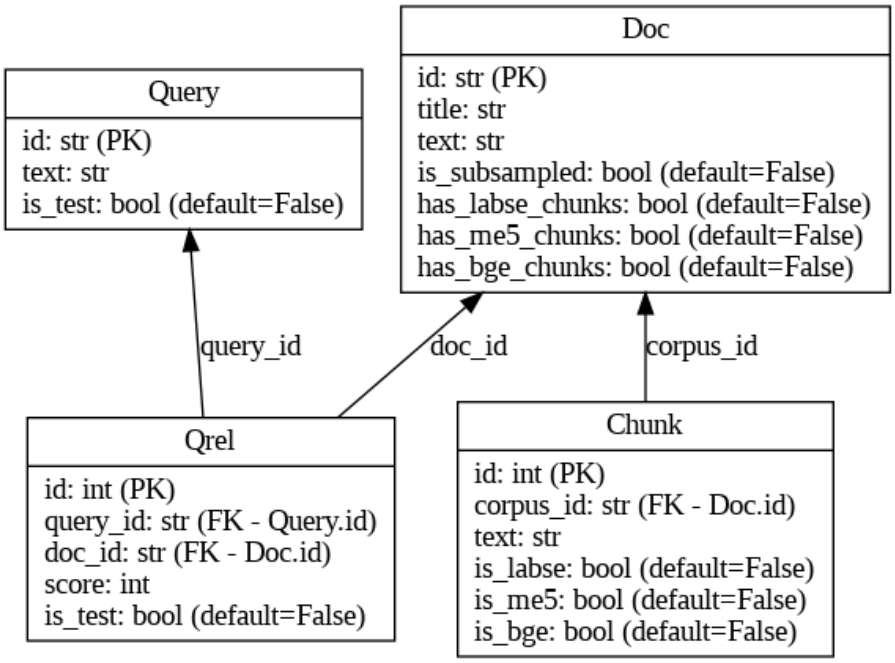

Figure 5.3: An UML diagram describing the structure of the database.

The `Doc` class models documents in the system. The purpose of this table is to store document metadata and provide flags for different types of chunking that may be applied to the document. Chunks are created using an overlap of 100 tokens for multilinual-e5-large and bge-m3 and an overlap of 50 for the LaBSE. Each document has the following fields:

- **id (str)**: A unique string identifier for each document that serves as the primary key. It

is also marked as unique, so that there are no duplicate IDs.

- **title (str)**: A string field that stores the title of the document.
- **text (str)**: A string field that contains the document's full textual content.
- **is_subsampled (bool)**: A Boolean field indicating whether the document has been subsampled to be included in the index during the test phase, with a default value of `False`.
- **has_labse_chunks (bool)**: A Boolean field indicates whether the document has been processed into chunks for the LaBSE model. The default is `False`.
- **has_me5_chunks (bool)**: A Boolean field indicating whether the document has been processed into chunks for the multilingual-e5-large model. The default is `False`.
- **has_bge_chunks (bool)**: A Boolean field indicating whether the document has been processed into chunks for the bge-m3 model. The default is `False`.

The `Query` class models search queries that can be associated with documents for document retrieval. This table is used to represent queries that users or systems can issue against the document corpus. It can differentiate between queries used for training and those used for testing.

- **id (str)**: A unique string identifier for each query, used as the primary key. This ensures that each query in the system is distinct.
- **text (str)**: A string field that stores the text of the query.
- **is_test (bool)**: A Boolean field indicating whether the query is part of a test set. By default, this field is `False`.

The `Qrel` class represents a query-document relevance judgment, which is used for the evaluation. This table is essential for relevance assessments, as it links queries to documents and assigns them a relevance score.

- **id (int)**: An autoincrementing integer that serves as the primary key, ensuring that each record in the table is unique.
- **query_id (str)**: A foreign key that references the `id` field in the `Query` table. This establishes a relationship between a query and its relevance assessments.
- **doc_id (str)**: A foreign key that references the `id` field in the `Doc` table, linking a specific document to a query.

- **score (int)**: An integer field that represents the relevance score of the document for the query. Higher scores typically indicate higher relevance.
- **is_test (bool)**: A Boolean field indicating whether this relevance judgment is part of a test set. The default is `False`.

The class `Chunk` represents chunks or segments of documents, often created during preprocessing.

- **id (int)**: An auto-incrementing integer that serves as the primary key for the chunk.
- **corpus_id (str)**: A foreign key that references the field `id` in the `Doc` table. This associates the chunk with a specific document in the corpus.
- **text (str)**: This field stores the textual content of the chunk.
- **is_labse (bool)**: A Boolean field indicating whether this chunk was created using the tokenizer from LaBSE model. The default is `False`.
- **is_me5 (bool)**: A Boolean field indicating whether this chunk was created using the tokenizer from the multilingual-e5-large model. The default is `False`.
- **is_bge (bool)**: A Boolean field indicating whether this chunk was created using the tokenizer from the bge-m3 model. The default is `False`.

The database schema establishes several relationships between the tables:

- The table `Doc` is linked to the table `Chunk` through the field `corpus_id`, representing the association between a document and its processed chunks, creating a one-to-many relation between the 2 tables.
- The table `Qrel` connects queries from the table `Query` with documents from the table `Doc` through foreign keys `query_id` and `doc_id`. This allows for relevance judgments linking queries to documents. Specifically, this table is used to create a many-to-many relation inside the database.

We selected this schema because we think it is well suited for the semantic search system, taking inspiration from the structure of the Dbpedia-Entity-v2 dataset from Hasibi et al. [3]. The design supports training and testing tasks, chunking with various models, and relevance judgments to facilitate ranking and evaluation, also including the different models.

5.2.4 Wilcoxon signed-rank test

In NLP research, statistical tests are essential to validate whether improvements in model performance are statistically significant or due to random chance. This chapter focuses on the Wilcoxon signed-rank test, that is, a non-parametric rank test for statistical hypothesis testing used to compare the locations of two populations using two matched samples, in our case the results of two models on the Dbpedia-Entity-v2 test set [3].

The choice of the Wilcoxon signed-rank test has been made since the non-normal distribution is often observed in NLP model performance metrics. Unlike parametric tests, which assume a normal distribution of the data, non-parametric tests like the Wilcoxon signed-rank test do not require this assumption. This flexibility makes the Wilcoxon test particularly suitable for our evaluations, since we tested the data's normality, and it was not always the case.

We define the test hypotheses between two models X and Y as follows:

- **Null Hypothesis H_0:** There is no significant difference in performance between the two models. Formally,
$$H_0 : X - Y = 0.$$
This implies that any difference observed in performance is due to random chance.
- **Two sided alternative Hypothesis H_1:** There is a significant difference in performance between the two models. Formally,
$$H_1 : X - Y \neq 0.$$
This suggests that the performance of one model consistently differs from the other.

In applying the Wilcoxon signed-rank test, each data point in the two samples represents the performance of the two models on a single instance from the Dbpedia-Entity-v2 test set. By focusing on the relative differences between paired observations, the Wilcoxon test allows us to determine whether the observed differences in performance are statistically significant on our data.

The test is conducted as follows:

1. For each test instance, we calculate the difference between the performance metric nDCG@10 of Model X and Model Y.
2. The differences are then ranked in ascending order, based on their absolute values. Zero differences are discarded from the ranking and the ties are replaced with the average of the respective ranks.

3. After ranking the differences, 2 values are computed: T^+ and T^-, which are respectively the sums of the ranks of positive and negative differences.

4. Finally, the Wilcoxon signed-rank statistic W is calculated as the minimum between T^+ and T^-.

This test statistic W is then used to calculate the p-value in the following way:

1. First, we compute the mean
$$\mu_W = \frac{n \cdot (n+1)}{4}, \tag{5.1}$$
where n is the number of samples minus the number of differences discarded.

2. Then we compute the standard deviation
$$\sigma_W = \sqrt{\frac{n(n+1)(2n+1) - \sum_{i=1}^{k} \frac{t_i^3 - t_i}{2}}{24}}, \tag{5.2}$$
where k is the number of tied ranks and t_i is the number of samples that share the same rank i.

3. The last step is to compute the z-statistic with the formula
$$z = \frac{W - \mu_W}{\sigma_W} \tag{5.3}$$

In our case W is supposed to be normally distributed, since the sample size is large ($n = 400$). A very low p-value (typically $p < 0.001$) indicates that the differences in performance between the two models are strongly statistically significant, allowing us to reject the null hypothesis H_0 that the models perform similarly in the test set.

Using the Wilcoxon signed-rank test allows us to provide a robust statistical basis for comparing two selected models on the Dbpedia-Entity-v2 test set. By assessing whether observed differences in retrieval performance are statistically significant, we can determine, for example, if our fine-tuning approach yields genuine improvements in semantic search accuracy by looking at the base model. The statistical significance of any performance gain is crucial for validating that the observed model enhancements are not artifacts of random variability.

In our experiments, we applied this test to the main metric in the Dbpedia-Entity-v2 dataset: **nDCG@10**.

5.3 Vector Database

In our research, vector databases have proven to be a highly effective tool. These specialized databases are designed to handle both structured and unstructured data types, including integers and text, while also managing their high-dimensional vector representations, thus their embeddings. What distinguishes vector databases is their ability to facilitate efficient access and search across these complex data structures.

The relevance of vector databases to our semantic search domain is due to three key advantages. First, their efficiency is evident in their optimized design, which supports rapid searches within high-dimensional spaces. Second, they offer impressive scalability, easily accommodating large datasets and handling high volumes of queries. Finally, they achieve high accuracy by leveraging advanced algorithms that retrieve the most relevant and similar results.

5.3.1 Search Algorithm: Hierarchical Navigable Small World and Exact Search

In our project, we focussed on two specific search algorithms within the vector database: Hierarchical Navigable Small World (HNSW) algorithm, first presented by Malkov et al. [37], and exhaustive search.

Its older version, NSW, is a graph-based algorithm that finds approximate nearest neighbors in a dataset. The general idea here is first to imagine many nodes in a network. Each node will have short-, medium- and long-range connections to other nodes. When performing a vector search, the algorithm begins at some pre-defined entry point. From there, it evaluates connections to other nodes and jumps to the one closest to the one we hope to find. This process repeats until our nearest neighbor is found.

HNSW is an approximate K-nearest neighbor search based on navigable small world graphs with a controllable hierarchy (Hierarchical NSW, HNSW). It is fully graph-based, without any need for additional search structures, which are typically used at the search stage of the most proximity graph techniques. HNSW incrementally builds a multi-layer structure consisting of hierarchical set of proximity graphs (layers) for nested subsets of the stored elements. The maximum layer in which an element is present is randomly selected with an exponentially decaying probability distribution. This allows producing graphs similar to NSW structures while additionally having the links separated by their characteristic distance scales. Starting search from the upper layer together with utilizing the scale separation boosts the performance compared

to NSW and allows a logarithmic complexity scaling. Thus, the HNSW algorithm provides a smart solution for efficient similarity search in large datasets, since the hierarchical approach makes HNSW highly efficient for both insertion and search tasks, which is particularly valuable when working with high-dimensional vector data. HNSW offers a viable alternative to exhaustive search methods, which have become increasingly impractical in large datasets due to the time required to calculate similarity scores across all entries. An overview of the search process can be seen in Figure 5.4.

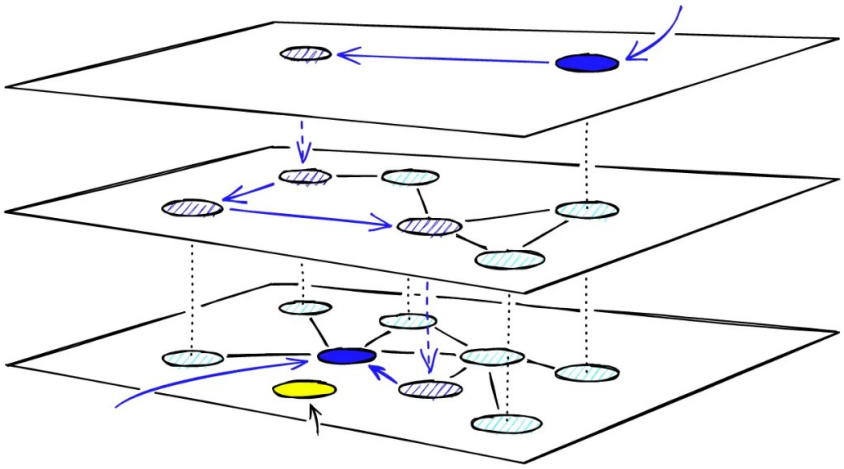

Figure 5.4: Hierarchical Navigable Small World (HNSW) graph structure demonstrating multi-layered search with long-range and local connections for efficient K-nearest neighbor search. This image is taken from Pinecone website.

Our implementation used Weaviate[†], a powerful vector database platform that offers robust functionality, including local deployment through Docker, which proved to be particularly practical for our work. Based on how models are trained, we used cosine similarity as our similarity metric. We were able to compare both the HNSW and exhaustive search approaches, as our dataset of 150,000 documents was manageable enough to allow both. Although the exhaustive search produced the most accurate results, HNSW showed a substantial advantage in speed. In chapter 6, we have reported the findings of both methods, providing a comprehensive comparison. Notably, in a production environment, HNSW would be the preferred choice due to its scalability and efficiency.

[†]`https://weaviate.io/`

6

Experiments and Results

6.1 Symmetric Results

In the task concerning the symmetric dataset, our approach was to perform feature extraction across the selected models. This decision aimed to evaluate and compare the semantic representation capabilities of both neural and non-neural models. The full results are presented in Table 7.1 and Table 7.2 in the Appendix. Our discussion will center on our primary performance metric: MRR@10.

By analyzing the histogram in Figure 6.1 of the results and considering the dataset structure detailed in section 4.1, we observe that neural models consistently outperformed non-neural ones. This outcome likely reflects the challenge non-neural models face in capturing the dissimilar meanings of sentences that, while using similar key words, have distinctly different concepts. Keyword-based approaches, being limited to literal matches, struggle to interpret variations in semantic content where the same terms may apply in varied contexts.

The results also show a remarkable consistency in the performance of neural models, regardless of their differences in the training methodologies or underlying architectures. We hypothesize that this consistency is due to two key factors. First, the shortness of the queries minimizes the likelihood of running into the limitations associated with a model's context window, as shorter text typically requires less contextual management. Second, the simplified nature of individual queries, as opposed to longer documents, allows for a more straightforward semantic

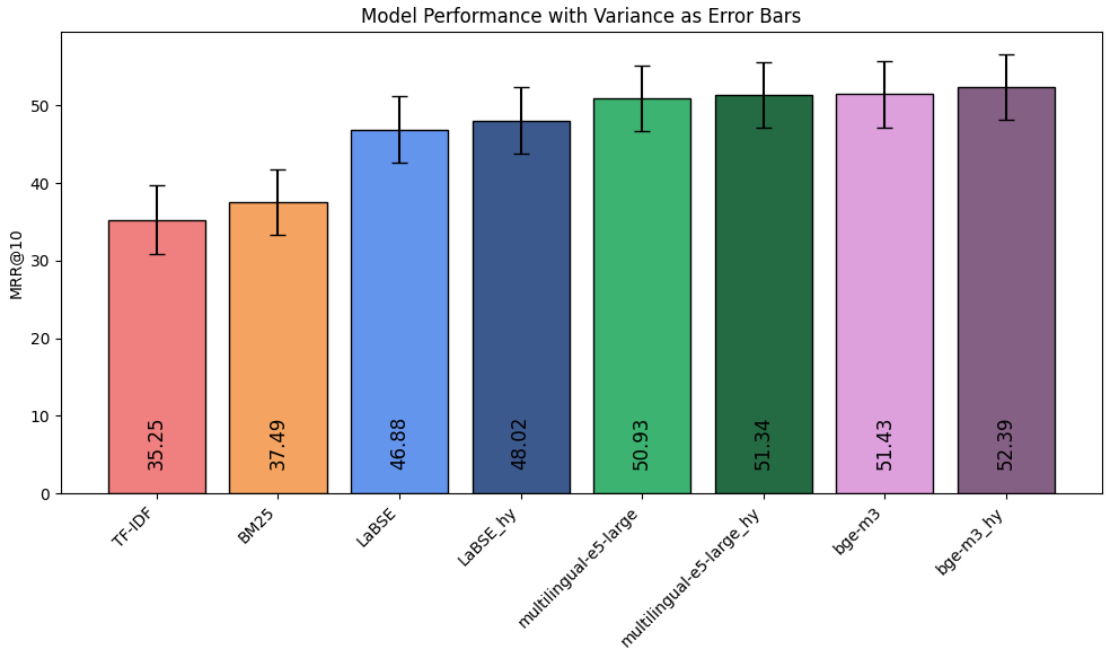

Figure 6.1: An histogram giving a visual representation of the performance of the different models in the case of the exact search for the symmetric case.

representation, leading to minimal performance variability between models. This suggests that neural models may have a consistent advantage in scenarios where semantic understanding is essential and contextual complexity is moderate, as observed in our symmetric dataset.

As illustrated in Figure 6.1, we evaluated twice the number of models, half of which included the suffix "_hy", indicating hybrid search. Hybrid search combines the outcomes of a vector-based search with BM25F – a keyword-based search method from the BM25 family – by fusing the two result sets. For this analysis, we used the default configuration for hybrid search without further exploration of hyperparameters, as our primary focus was on evaluating neural models.

The decision to include hybrid search was driven by its simplicity of implementation; it required only a single line of code modification in the testing script for Weaviate's vector database. The results shown in Figure 6.1 suggest that the hybrid search slightly outperformed the standard vector-based approach. We hypothesize that this improvement may be attributed to the structure of the dataset. Many semantically similar queries in the dataset often utilize overlapping terms arranged in varied ways to convey equivalent meanings. Incorporating keyword-based retrieval probably enhanced the system's ability to capture these nuanced similarities, thereby improving overall performance.

6.2 Asymmetric results

Given our two main objectives, this section is divided into two parts. The first part addresses the results of our fine-tuning process along with a detailed discussion of our fine-tuning methodology. Here, we analyze how effectively our approach adapts the model to our specific dataset, focusing on the decisions and adjustments that optimize retrieval relevance.

The second part presents a comparative analysis between our fine-tuned models and the OpenAI text-embedding-3-small model.

In these comparisons, we consistently use the nDCG@10 metric as our primary measure, as it captures the relevance of the top 10 retrieved documents, emphasizing how well each model ranks relevant results at the highest positions. Table 7.3 and Table 7.4 in the Appendix provide the complete set of results for both parts of this section, offering a detailed view of model performance across various configurations.

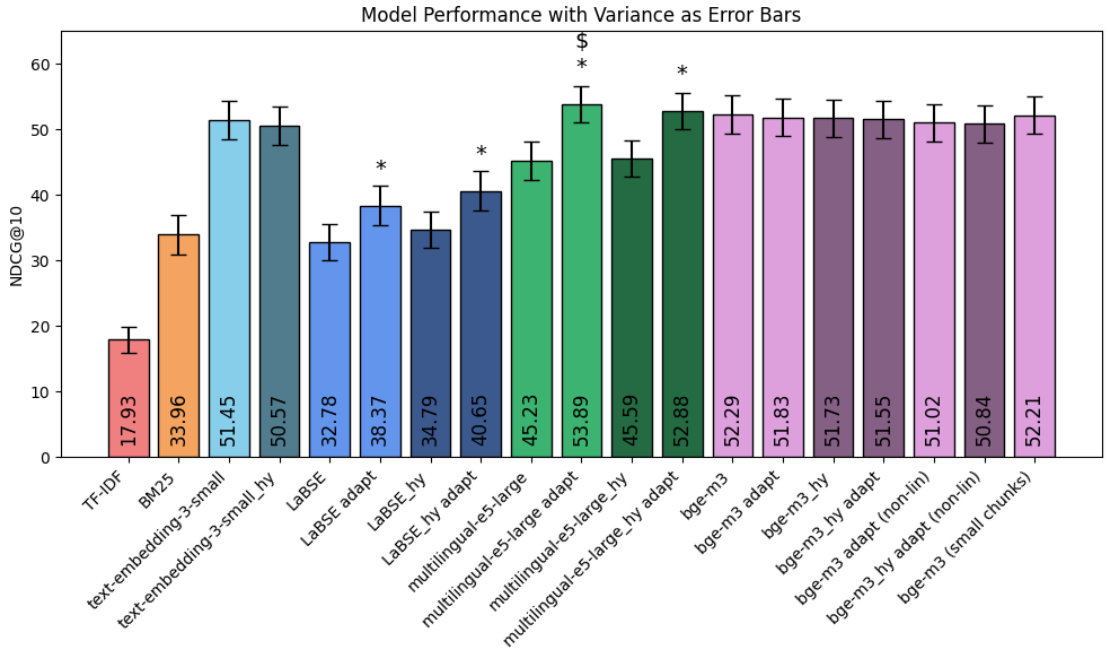

Figure 6.2: An histogram giving a visual representation of the performance of the different models in the case of the exact search for the asymmetric case.

6.2.1 Discussion on fine-tuning

Fine-tuning was performed only on a subset of the training dataset, as mMARCO is very large, as discussed in section 4.2.

Referring to the documentation of sentence-transformer [*] library, we decided to choose `TripletLoss` from the PyTorch library as the loss function for our model. This specific type of loss function requires data to be organized into triplets composed of: a query, a relevant document, and a non-relevant document. For this reason, each element of both the training and validation datasets was processed to achieve this configuration. This procedure exponentially increased the amount of data in the validation set, as all possible combinations between the relevant and non-relevant documents in the table of the database described in subsection 5.2.3 had to be created.

We decided to limit the fine-tuning to only 5M training samples and use a validation set of 10k elements, which validates 5 times per epoch, to minimize the usage of virtual machines as much as possible. We are aware that this method is not optimal, but performing a grid search would have taken too much time, so it was conducted based on partial training.

We decided to focus solely on the learning rate as the hyperparameter to optimize, given the limited computational resources and the need to avoid an excessively large grid search.

Looking at [38], we decided to schedule the learning rate. We used the function `SequentialLR` in pytorch, linearly increasing the learning rate in the first 10% of the training phase and decreasing cosinely in the remaining 90%, with the function `CosineAnnealingLR`. Linear increase has been chosen to simulate a linear warm-up, and cosine annealing is frequently used by the sentence-transformer library [†]. The plot of the full schedule is shown in Figure 6.3.

The learning rate is paired with the AdamW optimizer, which combines the classic Adam algorithm with weight decay. To avoid introducing an additional hyperparameter, we kept the weight decay at its default value of 0.01.

The possible values of the hyperparameters used are presented in Table 6.1.

To perform the grid search, only one epoch was used instead of the total 10. Each epoch for the LaBSE model lasts 3 minutes, while for the other two models, multilingual-e5-large and bge-m3, it takes 10 minutes per epoch. It should be noted that bfloat16 was used as a data type, which significantly accelerated the training. In fact, before switching data type, the times were 10 and 30 minutes, respectively, for the models mentioned above.

[*] `https://sbert.net/`
[†] `https://sbert.net/`

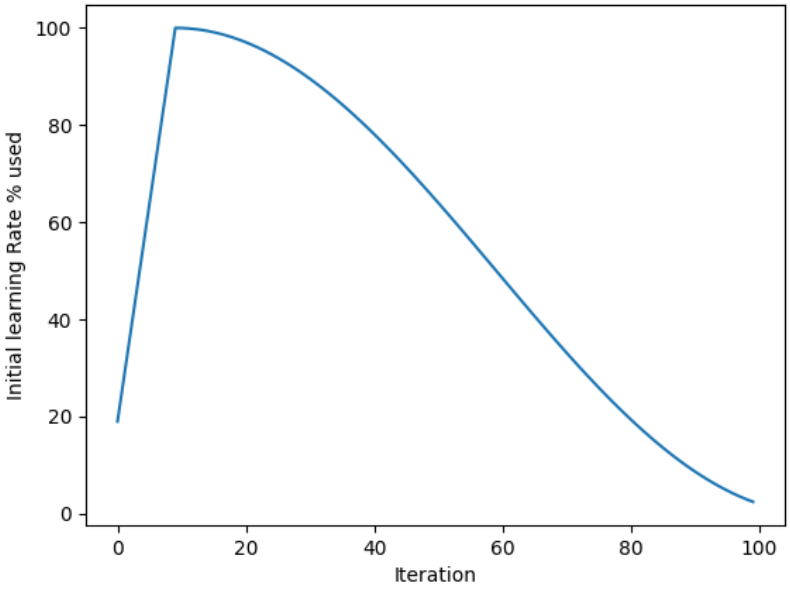

Figure 6.3: Learning rate schedule plot, with the percentage of the used learning rate in y-axis and over 100 iteration (example with 10 hypothetical epochs with 10 iterations each).

The batch size was chosen with the aim of minimizing computation time. This was achieved by manually monitoring the utilization of the available GPU using the command `nvidia-smi` and conducting tests to maximize its usage. Consequently, each model has a different batch size, as listed in Table 6.2, where the selected hyperparameters for all our models are detailed.

As mentioned in subsection 5.2.2, the number of parameters for each adapter is very small, resulting in model files that occupy minimal memory. When initializing our custom embedding model, it is sufficient to import the corresponding pre-trained model from HuggingFace and update only the adapter parameters with those obtained from our fine-tuning.

Model	Adapter	Learning Rate	Adapter inner dim
LaBSE	linear	$[5 \cdot 10^{-6}, 10^{-6}, 5 \cdot 10^{-5}, 10^{-5}, 5 \cdot 10^{-4}, 10^{-4}$, 0.005, 0.001, 0.05. 0.01]	None
mult-e5-large	linear	$[5 \cdot 10^{-6}, 10^{-6}, 5 \cdot 10^{-5}, 10^{-5}, 5 \cdot 10^{-4}, 10^{-4}$, 0.005, 0.001, 0.05. 0.01]	None
bge-m3	linear non-linear	$[5 \cdot 10^{-6}, 10^{-6}, 5 \cdot 10^{-5}, 10^{-5}, 5 \cdot 10^{-4}, 2.5 \cdot 10^{-4}, 10^{-4}$, 0.005, 0.001, 0.05. 0.01]	[512, 1024, 2048]

Table 6.1: Hyperparameter tested for each model configuration.

Model	Adapter	Learning Rate	Batch size	Adapter memory
LaBSE	linear	0.005	256	2.25 MB
mult-e5-large	linear	0.001	128	4.00 MB
bge-m3	linear	0.00025	128	4.00 MB
bge-m3	non-linear (ReLU w/ 512 inner dim)	0.001	32	16.01 MB

Table 6.2: Best hyperparameter selected for each model configuration.

The significant reduction in batch size for the bge-m3 model, after investigating the GPU memory usage during training, is believed to be due to the model's large context window (8196 tokens). In fact, when very long documents were processed by the model, the allocated memory increased dramatically. To avoid this issue, we had to reduce the batch size to only 32 documents at a time.

Obviously, the memory usage of the adapter parameters varies depending on the embedding size. For example, LaBSE is the lightest (with an embedding size of 768), while the adapters for multilingual-e5-large and bge-m3 share the same size. However, when using a 2-layer adapter, the memory usage increases accordingly due to the additional parameters.

The complete fine-tuning results with the hyperparameters listed in Table 6.2 can be viewed in Table 7.3 and Table 7.4.

Similarly to the approach used in section 6.1, as we can see in Figure 6.2, we evaluated twice the number of models, half of which included the suffix "_hy", indicating hybrid search. Hybrid search combines the outcomes of a vector-based search with BM25F – a keyword-based search method from the BM25 family – by fusing the two result sets. For this analysis, we used the default configuration for hybrid search without further exploration of hyperparameters, as our primary focus was on evaluating neural models.

We tested our models on the 400 queries of the Dbpedia-Entity-v2 test set, recording the inference times using the fastest search method, which leverages the HNSW index. As explained in subsection 5.3.1, this index is significantly more scalable for databases with a large number of documents, at the cost of slight retrieval accuracy. As expected, the larger the model, the longer the inference time, with a notable discrepancy compared to the OpenAI model. The inference times were measured locally on my personal machine, which is equipped with a laptop GPU 1050Ti and a 9th-generation i5 processor. These times will inevitably improve on more performant machines.

In the histogram in Figure 6.2, the results are statistically significant at 0.001% level in the

comparison between the fine-tuned and non-fine-tuned models are marked with the $*$ (asterisk) symbol, indicating where the p-value is less than 0.001.

Results from the Wilcoxon signed-rank test showed that the fine-tuned model outperformed its non-finetuned version for the two models LaBSE and multilingual-e5-large, suggesting that it retrieves more relevant documents within the top 10 results. This improvement is particularly important in an RAG context, where retrieving high-quality supporting documents improves the generation relevance score.

Our best model among all those presented is the multilingual-e5-large with a linear adapter, fine-tuned by us, achieving an nDCG@10 of 53.89 and an average inference time of 47 milliseconds per query.

As for the bge-m3 model, unfortunately, we were unable to improve its performance with the resources we had. This could be the subject of future investigations. We also decided to stop trying to improve this model, as the results with multilingual-e5-large were quite promising. We will discuss this in more detail in subsection 6.2.2.

6.2.2 MULTILINGUAL-E5-LARGE WITH ADAPTER VS TEXT-EMBEDDING-3-SMALL

We decided to test all our models with text-embedding-3-small. The only one worth testing was our best model: multilingual-e5-large with an adapter.

Once again, we used the Wilcoxon signed-rank test to verify the significance of our results. It turns out that our model is statistically better with a p-value < 0.05, which is indicated with the \$ (dollar) symbol. Although this is not an excellent result, considering that it was tested on a relatively small test set ($n = 400$ samples), it is still a good outcome.

In the future, we could consider using our best fine-tuned model instead of the OpenAI model, thus avoiding the costs associated with embeddings and API calls during usage. The models we have presented are also capable of running on machines equipped only with CPUs, although with significantly longer times for the computation of the query embedding. As for the search in the database, it is still conducted through a Docker container that utilizes the CPU.

From Table 6.3, we can observe that the inference times increase by an order of magnitude when using OpenAI embeddings (615 ms compared to 47 ms for the multilingual-e5-large model). This is due to both the API call to the client and the fact that the search must be performed on embeddings with a size of 1536 instead of 1024. In a potential RAG application,

Model	Emebedding dimension	Average search time
LaBSE linear	768	32 ms
multilingual-e5-large linear	1024	47 ms
bge-m3	1024	52 ms
text-embedding-3-small	1536	615 ms

Table 6.3: Average search times for the test set of 400 queries, based on the embedding dimension of each model.

this is highly relevant as we want the documents to be retrieved as quickly as possible, to then be fed into a language model capable of extracting and reformulating the information present in the documents. This second step takes more time because an LLM typically has many more parameters.

The cost of generating embeddings using the text-embedding-3-small model is 0.28 \$ (via batch processing to minimize API calls), as the 150,000 documents selected from the Dbpedia-Entity-v2 dataset consist of 13,782,932 tokens. However, it is important to note that each user query incurs an additional API call. Although this cost is relatively low, as discussed in chapter 3, it is not negligible.

The total cost of training the multilingual-e5-large model, including grid search (2 hours) and training (1 hour and 40 minutes) on a g5.xlarge machine costing 1.258 \$ per hour, amounts to 4.61 \$.

On the other hand, embedding generation for other models was cost-free, as it was performed in approximately one hour on my personal machine equipped with a 1050 Ti Mobile GPU, as mentioned earlier.

Overall, these are very low costs, especially considering the improvement in the nDCG@10 metric and the significant reduction in document ranking errors achieved.

7 Conclusion

The main objective of this thesis was to develop and evaluate a cost-effective semantic search system for the Italian language by fine-tuning state-of-the-art multilingual embedding models, specifically those featured in the MTEB benchmark leaderboard. The primary focus was to create an optimal semantic space for embedding both queries and documents, with an emphasis on asymmetric semantic search frameworks due to their relevance in future integrations with RAG systems.

To achieve this goal, we compared state-of-the-art models (e.g. LaBSE, multilingual-e5-large, bge-m3) with newly fine-tuned versions utilizing an adapter-based fine-tuning approach. In particular, adapters were applied to query embeddings to enhance their alignment with document embeddings, addressing the inherent semantic disparity between the two.

The models were trained and tested on diverse datasets, including Quora Question Pairs, mMARCO, and Dbpedia-Entity-v2. Since two of these datasets were originally available only in English, we leveraged a transformer-based model from Helsinki University for translation, complemented by the Google Translate API to improve translation quality. Computational resources were provided through Google Colab for the translation step and through AWS for the fine-tuning stage.

We evaluated the models using metrics such as nDCG@10 and MRR@10 and employed efficient vector search algorithms such as Hierarchical Navigable Small World (HNSW), alongside exhaustive search, to ensure robust benchmarking. Despite resource constraints that limit the database to 150,000 documents, the results provided valuable insights.

Our fine-tuned models demonstrated statistically significant improvements over their base versions for two out of three models. Specifically, adapter-enhanced multilingual e5-large and LaBSE models outperformed their baseline counterparts with a p-value < 0.001. In particular, the adapter-equipped multilingual E5-large model also outperformed OpenAI's text embedder-3-small in our test set, achieving a p-value < 0.05. However, bge-m3 fine-tuning did not yield improvements, highlighting potential limitations in its compatibility with the adapter-based approach.

Although this thesis highlights the potential of adapter-based fine-tuning to improve semantic search in Italian, several limitations emerged.

First, the relatively small size and limited diversity of the test dataset and the document database could affect the generalizability of the findings. Expanding the experiments to include larger datasets, such as the full Dbpedia-Entity-v2 or domain-specific corpora, could enable a more comprehensive evaluation and help mitigate biases introduced by dataset constraints.

Second, the reliance on translation tools like Helsinki's neural translation model and Google Translate API, though effective for this study, introduces potential artifacts or errors that could distort semantic relationships. Future work could address this limitation by using prealigned English-Italian datasets, trying to eliminate translation-related biases. Moreover, a domain-specific dataset could be explored and those models could be evaluated to determine whether they perform effectively only in general scenarios or are better suited for specific use cases, like in legal or medical domains.

In addition, the study revealed model-specific limitations. For example, bge-m3 did not show any improvement in adapter-based fine-tuning, suggesting compatibility issues. Further investigation into the internal representations of such models could clarify why certain architectures fail to benefit from this approach.

Future research could build on this work in several ways. Exploring alternative fine-tuning techniques, such as prompt-tuning or prefix tuning, might uncover more effective methods for adapting models to specific tasks for Italian language. Investigation of failed fine-tuning techniques might be further pursued. For example, by modifying the LoRA implementation or trying to implement it manually, although the latter would require a lot of time.

Additionally, a reranking strategy could be implemented to increase the reliability of the retrieved documents. This would further boost performance in our pipeline.

Furthermore, pre-processing stages like query rewriting could be integrated in the pipeline, where we can prompt LLM to rewrite the queries providing further context to address any lack of specific semantic load, thereby ensuring the optimal relevance of the generated answers.

Another promising direction involves embedding the fine-tuned models into a complete RAG pipeline. Such an integration would provide valuable insights into their performance in real-world applications, particularly when tested on larger or specialized datasets, such as legal or medical domains.

In addition, incorporating user-centered evaluation metrics would offer a more clear understanding of system effectiveness. Testing under adversarial conditions or with noisy queries could further assess the robustness of the models.

Finally, making fine-tuned models and code publicly available could encourage community-driven improvements and support the development of open-source alternatives to proprietary embedding systems.

By addressing these limitations and exploring these future directions, subsequent studies could advance semantic search capabilities for Italian and other languages, promoting more accessible NLP solutions.

References

[1] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2312.10997

[2] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," 2023. [Online]. Available: https://arxiv.org/abs/2210.07316

[3] F. Hasibi, F. Nikolaev, C. Xiong, K. Balog, S. E. Bratsberg, A. Kotov, and J. Callan, "Dbpedia-entity v2: A test collection for entity search," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17. ACM, 2017, pp. 1265–1268.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[6] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49313245

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford,

I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[9] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, and R. Campbell, "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[10] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[11] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, and T. Lavril, "Llama 2: Open foundation and fine-tuned chat models," 2023. [Online]. Available: https://arxiv.org/abs/2307.09288

[12] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, A. Yang, A. Fan, A. Goyal, A. Hartshorn, A. Yang, A. Mitra, A. Sravankumar, A. Korenev, A. Hinsvark, A. Rao, A. Zhang, A. Rodriguez, A. Gregerson, A. Spataru, B. Roziere, B. Biron, B. Tang, B. Chern, C. Caucheteux, C. Nayak, C. Bi, C. Marra, C. McConnell, and C. Keller, "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[13] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, and V. Misra, "Palm: Scaling language modeling with pathways," 2022. [Online]. Available: https://arxiv.org/abs/2204.02311

[14] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013

[15] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318. [Online]. Available: https://doi.org/10.3115/1073083.1073135

[16] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[17] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," 2020. [Online]. Available: https://arxiv.org/abs/1911.02116

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[19] G. Lample and A. Conneau, "Cross-lingual language model pretraining," 2019. [Online]. Available: https://arxiv.org/abs/1901.07291

[20] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models," 2021. [Online]. Available: https://arxiv.org/abs/2104.08663

[21] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic bert sentence embedding," 2022. [Online]. Available: https://arxiv.org/abs/2007.01852

[22] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei, "Multilingual e5 text embeddings: A technical report," 2024. [Online]. Available: https://arxiv.org/abs/2402.05672

[23] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "Bge m3-embedding: Multilingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," 2024. [Online]. Available: https://arxiv.org/abs/2402.03216

[24] S. Xiao, Z. Liu, Y. Shao, and Z. Cao, "Retromae: Pre-training retrieval-oriented language models via masked auto-encoder," 2022. [Online]. Available: https://arxiv.org/abs/2205.12035

[25] DataCanary, hilfialkaff, L. Jiang, M. Risdal, N. Dandekar, and tomtung, "Quora question pairs," https://kaggle.com/competitions/quora-question-pairs, 2017.

[26] J. Tiedemann and S. Thottingal, "OPUS-MT — Building open translation services for the World," in *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal, 2020.

[27] J. Tiedemann, M. Aulamo, D. Bakshandaeva, M. Boggia, S.-A. Grönroos, T. Nieminen, A. Raganato Y. Scherrer, R. Vazquez, and S. Virpioja, "Democratizing neural machine translation with OPUS-MT," *Language Resources and Evaluation*, no. 58, pp. 713–755, 2023.

[28] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang, "Ms marco: A human generated machine reading comprehension dataset," 2018. [Online]. Available: https://arxiv.org/abs/1611.09268

[29] L. Bonifacio, V. Jeronymo, H. Q. Abonizio, I. Campiotti, M. Fadaee, R. Lotufo, and R. Nogueira, "mmarco: A multilingual version of the ms marco passage ranking dataset," 2022. [Online]. Available: https://arxiv.org/abs/2108.13897

[30] R. Blanco, H. Halpin, D. M. Herzig, P. Mika, J. Pound, and H. S. Thompson, "Entity search evaluation over structured web data," 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:9926859

[31] H. Halpin, D. Herzig, P. Mika, R. Blanco, J. Pound, H. Thompson, and T. Duc, "Evaluating ad-hoc object retrieval," 01 2010.

[32] Q. Wang, J. Kamps, G. R. Camps, M. Marx, A. Schuth, M. Theobald, S. Gurajada, and A. Mishra, "Overview of the INEX 2012 linked data track," in *CLEF 2012*

Evaluation Labs and Workshop, Online Working Notes, Rome, Italy, September 17-20, 2012, ser. CEUR Workshop Proceedings, P. Forner, J. Karlgren, and C. Womser-Hacker, Eds., vol. 1178. CEUR-WS.org, 2012. [Online]. Available: https://ceur-ws.org/Vol-1178/CLEF2012wn-INEX-WangEt2012.pdf

[33] G. Demartini, T. Iofciu, and A. P. de Vries, "Overview of the inex 2009 entity ranking track," in *Focused Retrieval and Evaluation*, S. Geva, J. Kamps, and A. Trotman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 254–264.

[34] K. Balog, A. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld, "Overview of the trec 2009 entity track," 01 2009.

[35] V. Lopez, C. Unger, P. Cimiano, and E. Motta, "Evaluating question answering over linked data," *Journal of Web Semantics*, vol. 21, 06 2013.

[36] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[37] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," 2018. [Online]. Available: https://arxiv.org/abs/1603.09320

[38] J. Kaddour, O. Key, P. Nawrot, P. Minervini, and M. J. Kusner, "No train no gain: Revisiting efficient training algorithms for transformer-based language models," 2023. [Online]. Available: https://arxiv.org/abs/2307.06440

Appendix

Model	MRR@10	Recall@10
TF-IDF	35.25 ± 19.49	43.05 ± 24.52
BM25	37.49 ± 17.98	57.54 ± 24.43
LaBSE	46.88 ± 18.3	70.64 ± 20.74
LaBSE_hy	48.02 ± 18.26	72.09 ± 20.12
multilingual-e5-large	50.93 ± 17.8	76.36 ± 18.05
multilingual-e5-large_hy	51.34 ± 17.82	76.98 ± 17.72
bge-m3	51.43 ± 17.91	76.81 ± 17.81
bge-m3_hy	52.39 ± 17.79	77.74 ± 17.3

Table 7.1: Comparison of different models on various evaluation metrics for the symmetric case with exact search.

Model	MRR@10	Recall@10
TF-IDF	35.25 ± 19.49	43.05 ± 24.52
BM25	37.49 ± 17.98	57.54 ± 24.43
LaBSE	46.51 ± 18.25	70.52 ± 20.79
LaBSE_hy	48.06 ± 18.23	72.44 ± 19.96
multilingual-e5-large	51.94 ± 17.97	77.08 ± 17.67
multilingual-e5-large_hy	51.45 ± 17.79	77.13 ± 17.64
bge-m3	51.76 ± 17.86	77.22 ± 17.59
bge-m3_hy	51.56 ± 17.69	77.59 ± 17.39

Table 7.2: Comparison of different models on various evaluation metrics for the symmetric case with HNSW search.

Model	nDCG@10	MAP@10	MRR@10	Prec@10	Recall@10	f1@10
TF-IDF	17.93 ± 4.09	5.7 ± 1.23	42.08 ± 18.99	7.88 ± 2.09	27.0 ± 9.08	9.65 ± 1.61
BM25	33.96 ± 8.83	11.63 ± 2.75	52.57 ± 17.59	18.03 ± 4.59	27.65 ± 7.36	16.48 ± 2.16
text-embedding-3-small	51.45 ± 8.44	21.45 ± 5.02	74.17 ± 13.61	28.9 ± 6.55	35.91 ± 6.66	24.13 ± 2.39
text-embedding-3-small_hy	50.57 ± 8.43	20.95 ± 4.81	72.7 ± 13.71	28.83 ± 6.76	35.5 ± 6.62	23.98 ± 2.42
LaBSE	32.78 ± 7.82	10.43 ± 1.95	53.46 ± 17.8	16.23 ± 3.38	24.3 ± 5.61	14.98 ± 1.71
LaBSE adapt	38.37 ± 9.17	13.12 ± 2.74	59.51 ± 17.81	19.28 ± 4.38	27.8 ± 6.65	17.1 ± 1.88
LaBSE_hy	34.79 ± 7.58	11.44 ± 2.02	54.86 ± 16.81	18.32 ± 4.03	25.7 ± 5.52	16.27 ± 1.74
LaBSE_hy adapt	40.65 ± 9.13	14.39 ± 3.14	63.21 ± 17.17	20.52 ± 4.37	29.09 ± 6.68	18.13 ± 1.92
multilingual-e5-large	45.23 ± 8.22	17.81 ± 3.54	67.41 ± 15.55	25.69 ± 5.98	31.91 ± 6.26	21.39 ± 2.17
multilingual-e5-large adapt	**53.89** ± 7.55	**21.55** ± 3.92	**78.47** ± 12.23	**29.33** ± 6.2	**38.07** ± 6.82	**24.93** ± 2.13
multilingual-e5-large_hy	45.59 ± 7.81	17.99 ± 3.59	67.35 ± 14.59	26.45 ± 6.2	32.02 ± 6.0	21.65 ± 2.14
multilingual-e5-large_hy adapt	52.88 ± 7.66	21.3 ± 4.03	76.4 ± 12.47	28.96 ± 6.2	37.2 ± 6.65	24.5 ± 2.17
bge-m3	52.29 ± 8.22	21.42 ± 4.65	76.55 ± 12.89	27.85 ± 6.04	36.36 ± 6.88	23.71 ± 2.31
bge-m3 adapt	51.83 ± 7.88	20.49 ± 4.13	75.91 ± 13.13	27.67 ± 5.89	36.34 ± 6.58	23.57 ± 2.12
bge-m3_hy	51.73 ± 8.04	20.87 ± 4.29	75.52 ± 12.93	28.19 ± 6.05	36.34 ± 6.87	23.8 ± 2.21
bge-m3_hy adapt	51.55 ± 7.86	20.41 ± 4.18	75.64 ± 13.29	27.89 ± 6.1	36.09 ± 6.55	23.54 ± 2.13
bge-m3 adapt (non-lin)	51.02 ± 8.12	20.11 ± 4.32	75.76 ± 13.05	27.03 ± 6.08	35.73 ± 6.71	22.97 ± 2.15
bge-m3_hy adapt (non-lin)	50.84 ± 8.1	20.08 ± 4.26	75.31 ± 13.2	26.91 ± 5.98	35.64 ± 6.73	22.99 ± 2.17
bge-m3 (small chunks)	52.21 ± 8.22	21.41 ± 4.65	76.45 ± 12.87	27.85 ± 6.04	36.32 ± 6.88	23.7 ± 2.31

Table 7.3: Comparison of different models on various evaluation metrics for the asymmetric case with exact search.

Model	nDCG@10	MAP@10	MRR@10	Prec@10	Recall@10	f1@10
TF-IDF	17.93 ± 4.09	5.7 ± 1.23	42.08 ± 18.99	7.88 ± 2.09	27.0 ± 9.08	9.65 ± 1.61
BM25	33.96 ± 8.83	11.63 ± 2.75	52.57 ± 17.59	18.03 ± 4.59	27.65 ± 7.36	16.48 ± 2.16
text-embedding-3-small	51.14 ± 8.51	21.17 ± 4.89	73.83 ± 13.84	28.47 ± 6.47	35.86 ± 6.71	24.02 ± 2.42
text-embedding-3-small_hy	50.73 ± 8.43	21.05 ± 4.84	72.9 ± 13.66	29.0 ± 6.84	35.59 ± 6.61	24.08 ± 2.43
LaBSE	32.65 ± 7.8	10.43 ± 1.95	53.08 ± 17.76	16.37 ± 3.45	24.16 ± 5.59	14.97 ± 1.71
LaBSE adapt	38.33 ± 9.24	13.09 ± 2.75	59.42 ± 17.89	19.11 ± 4.28	27.86 ± 6.69	17.13 ± 1.91
LaBSE_hy	34.66 ± 7.59	11.42 ± 2.04	54.65 ± 16.82	18.33 ± 4.08	25.61 ± 5.5	16.25 ± 1.77
LaBSE_hy adapt	40.62 ± 9.13	14.39 ± 3.15	63.34 ± 17.19	20.47 ± 4.37	29.02 ± 6.66	18.08 ± 1.92
multilingual-e5-large	45.31 ± 8.23	17.87 ± 3.55	67.46 ± 15.52	25.71 ± 5.98	31.95 ± 6.26	21.42 ± 2.17
multilingual-e5-large adapt	**53.69** ± 7.54	**21.33** ± 3.79	**78.27** ± 12.34	**29.18** ± 6.12	**38.07** ± 6.8	**24.94** ± 2.14
multilingual-e5-large_hy	45.49 ± 7.87	17.97 ± 3.58	67.44 ± 14.65	26.22 ± 6.14	31.95 ± 6.04	21.6 ± 2.17
multilingual-e5-large_hy adapt	52.96 ± 7.71	21.37 ± 4.05	76.19 ± 12.45	29.05 ± 6.25	37.32 ± 6.71	24.59 ± 2.2
bge-m3	52.34 ± 8.22	21.46 ± 4.67	76.61 ± 12.83	27.86 ± 6.04	36.39 ± 6.89	23.72 ± 2.31
bge-m3 adapt	51.78 ± 7.9	20.47 ± 4.14	75.74 ± 13.16	27.61 ± 5.88	36.32 ± 6.59	23.54 ± 2.12
bge-m3_hy	51.77 ± 7.99	20.84 ± 4.26	75.67 ± 12.88	28.18 ± 5.98	36.39 ± 6.82	23.81 ± 2.18
bge-m3_hy adapt	51.62 ± 7.89	20.45 ± 4.19	75.77 ± 13.29	27.96 ± 6.14	36.16 ± 6.59	23.6 ± 2.15
bge-m3 adapt (non-lin)	51.05 ± 8.1	20.12 ± 4.32	75.77 ± 13.05	27.04 ± 6.07	35.75 ± 6.7	22.99 ± 2.15
bge-m3_hy adapt (non-lin)	50.82 ± 8.11	20.08 ± 4.27	75.44 ± 13.2	26.86 ± 5.98	35.59 ± 6.71	22.96 ± 2.18
bge-m3 (small chunks)	52.26 ± 8.22	21.43 ± 4.65	76.48 ± 12.84	27.87 ± 6.04	36.36 ± 6.89	23.73 ± 2.32

Table 7.4: Comparison of different models on various evaluation metrics for the asymmetric case with HNSW search.