BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation

Jianlv Chen[♣] **Shitao Xiao**[♠†] **Peitian Zhang**[♠] **Kun Luo**[♠] **Defu Lian**[♣] **Zheng Liu**[♠*]

[♠] BAAI [♣] University of Science and Technology of China

stxiao@baai.ac.cn {namespace.pt,luokun695,zhengliu1026}@gmail.com

chenjianlv@mail.ustc.edu.cn liandefu@ustc.edu.cn

Abstract

In this paper, we present a new embedding model, called **M3-Embedding**, which is distinguished for its versatility in *Multi-Linguality*, *Multi-Functionality*, and *Multi-Granularity*. It can support more than 100 working languages, leading to new state-of-the-art performances on multi-lingual and cross-lingual retrieval tasks. It can simultaneously perform the three common retrieval functionalities of embedding model: dense retrieval, multi-vector retrieval, and sparse retrieval, which provides a unified model foundation for real-world IR applications. It is able to process inputs of different granularities, spanning from short sentences to long documents of up to 8192 tokens. The effective training of M3-Embedding involves the following technical contributions. We propose a novel self-knowledge distillation approach, where the relevance scores from different retrieval functionalities can be integrated as the teacher signal to enhance the training quality. We also optimize the batching strategy, enabling a large batch size and high training throughput to ensure the discriminativeness of embeddings. To the best of our knowledge, M3-Embedding is the first embedding model which realizes such a strong versatility. The model and code will be publicly available at https://github.com/FlagOpen/FlagEmbedding.

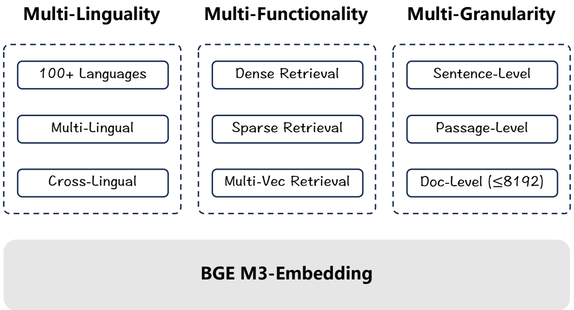

Figure 1: **Characters of M3-Embedding.**

1 Introduction

Embedding models are a critical form of DNN application in natural language processing. They encode the textual data in the latent space, where the underlying semantics of the data can be expressed by the output embeddings (Reimers and Gurevych, 2019; Ni et al., 2021). With the advent of pre-trained language models, the quality of text embeddings have been substantially improved, making them imperative components for information retrieval (IR). One common form of embedding-based IR is dense retrieval, where relevant answers to the query can be retrieved based on the embedding similarity (Karpukhin et al., 2020; Xiong et al., 2020; Neelakantan et al., 2022; Wang et al., 2022; Xiao et al., 2023). Besides, the embedding model can also be applied to other IR tasks, such as multi-vector retrieval where the fine-grained relevance between query and document is computed based on the interaction score of multiple embeddings (Khattab and Zaharia, 2020), and sparse or lexical retrieval where the importance of each term is estimated by its output embedding (Gao et al., 2021a; Lin and Ma, 2021; Dai and Callan, 2020).

Despite the widespread popularity of text embeddings, the existing methods are still limited in versatility. First of all, most of the embedding models are tailored only for English, leaving few viable options for the other languages. Secondly, the existing embedding models are usually trained for one single retrieval functionality. However, typical IR systems call for the compound workflow of multiple retrieval methods. Thirdly, it is challenging to train a competitive long-document retriever due to the overwhelming training cost, where most of the embedding models can only support short inputs.

To address the above challenges, we introduce **M3-Embedding**, which is pronounced for its breakthrough of versatility in *working languages*,

†. Co-first author
∗. Correspondence author

retrieval functionalities, and *input granularities*. Particularly, M3-Embedding is proficient in multi-linguality, which is able to support more than 100 world languages. By learning a common semantic space for different languages, enables both multi-lingual retrieval within each language and cross-lingual retrieval between different languages. Besides, it is able to generate versatile embeddings to support different retrieval functionalities, not just dense retrieval, but also sparse retrieval and multi-vector retrieval. Finally, M3-Embedding is learned to process different input granularities, spanning from short inputs like sentences and passages, to long documents of up to 8,192 input tokens.

The effective training of such a versatile embedding model poses a significant challenge. In our work, the following technical contributions are made to optimize the training quality. Firstly, we propose a novel *self-knowledge distillatio*n framework, where the multiple retrieval functionalities can be jointly learned and mutually reinforced. In M3-Embedding, the [CLS] embedding is used for dense retrieval, while embeddings from other tokens are used for sparse retrieval and multi-vector retrieval. Based on the principle of ensemble learning (Bühlmann, 2012), such heterogenous predictors can be combined as a stronger predictor. Thus, we integrate the relevance scores from different retrieval functions as the teacher signal, which is used to enhance the learning process via knowledge distillation. Secondly, we *optimize the batching strategy* to achieve a large batch size and high training throughput, which substantially contributes to the discriminativeness of embeddings. Last but not least, we perform comprehensive and high-quality *data curation*. Our dataset consists of three sources: 1) the extraction of unsupervised data from massive multi-lingual corpora, 2) the integration of closely related supervised data, 3) the synthesization of scarce training data. The three sources of data are complement to each other and applied to different stages of the training process, which lays foundation for the versatile text embeddings.

M3-Embedding exhibits remarkable versatility in our experiments. It achieves superior retrieval quality for a variety of languages, leading to state-of-the-art performances on popular multi-lingual and cross-lingual benchmarks like MIRACL (Zhang et al., 2023c) and MKQA (Longpre et al., 2021). It effectively learns the three retrieval functionalities, which can not only work individually but also work together for an even stronger retrieval quality. It also well preserves its superior capability across different input granularities within 8192 tokens, which outperforms the existing methods by a notable advantage.

Our work makes the following contributions. 1) We present M3-Embedding, which is the first model which supports multi-linguality, multi-functionality, and multi-granularity. 2) We propose a novel training framework of self-knowledge distillation and efficient batching strategy. And we also perform high-quality curation of training data. 3) Our model, code, and data will all be publicly available, which provides critical resources for both direct usage and future development of text embeddings.

2 Related Work

The related works are reviewed from three aspects: general text embeddings, embedding models for neural retrieval, embeddings of multi-linguality.

In the past few years, substantial progress has been achieved in the field of text embedding. One major driving force is the popularity of pre-trained language models, where the underlying semantic of the data can be effectively encoded by such powerful text encoders (Reimers and Gurevych, 2019; Karpukhin et al., 2020; Ni et al., 2021). In addition, the progress of contrastive learning is another critical factor, especially the improvement of negative sampling (Xiong et al., 2020; Qu et al., 2020) and the exploitation of knowledge distillation (Hofstätter et al., 2021; Ren et al., 2021; Zhang et al., 2021a). On top of these well-established techniques, it becomes increasingly popular to learn versatile embedding models, which are able to uniformly support a variety of application scenarios. So far, there have been many impactful methods in the direction, like Contriever (Izacard et al., 2022), LLM-Embedder (Zhang et al., 2023a), E5 (Wang et al., 2022), BGE (Xiao et al., 2023), SGPT (Muennighoff, 2022), and Open Text Embedding (Neelakantan et al., 2022), which significantly advance the usage of text embeddings for general tasks.

One major application of embedding models is neural retrieval (Lin et al., 2022). By measuring the semantic relationship with the text embeddings, the relevant answers to the input query can be retrieved based on the embedding similarity. The most common form of embedding-based retrieval method is dense retrieval (Karpukhin et al., 2020), where the text encoder's outputs are aggregated (e.g., via [CLS] or mean-pooling) to compute the embedding similarity. Another common alternative is known as multi-vecor retrieval (Khattab and Zaharia, 2020; Humeau et al., 2020), which applies fine-grained interactions for the text encoder's outputs to compute the embedding similarity. Finally, the text embeddings can also be transformed into term weights, which facilitates sparse or lexical retrieval (Luan et al., 2021; Dai and Callan, 2020; Lin and Ma, 2021). Typically, the above retrieval methods are realized by different embedding models. To the best of our knowledge, no existing method is able to unify all these functionalities.

Despite the substantial technical advancement, most of the existing text embeddings are developed only for English, where other languages are lagging behind. To mitigate this problem, continual efforts are presented from multiple directions. One is the development of pre-trained multi-lingual text encoders, such as mBERT (Pires et al., 2019), mT5 (Xue et al., 2020), XLM-R (Conneau et al., 2020). Another one is the curation of training and evaluation data for multi-lingual text embeddings, e.g., MIRACL (Zhang et al., 2023c), mMARCO (Bonifacio et al., 2021), Mr. TyDi (Zhang et al., 2021b), MKQA (Longpre et al., 2021). At the same time, the multi-lingual text embeddings are continually developed from the community, e.g., mDPR (Zhang et al., 2023b), mContriever (Izacard et al., 2022), mE5 (Wang et al., 2022), etc. However, the current progress is still far from enough given the notable gap with English models and the huge imbalance between different languages.

3 M3-Embedding

M3-Embedding realizes three-fold versatility. It supports a wide variety of languages and handles input data of different granularities. Besides, it unifies the common retrieval functionalities of text embeddings. Formally, given a query q in an arbitrary language x, it is able to retrieve document d in language y from the corpus D^y: $d^y \leftarrow \mathrm{fn}^*(q^x, D^y)$. In this place, $\mathrm{fn}^*(\cdot)$ belongs to any of the functions: dense, sparse/lexical, or multi-vector retrieval; y can be another language or the same language as x.

3.1 Data Curation

The training of M3-Embedding calls for a large-scale and diverse multi-lingual dataset. In this work, we perform comprehensive data collection from three sources: the unsupervised data from unlabeled corpora, the fine-tuning data from labeled corpora, and the fine-tuning data via synthesization (shown as Table 1). The three data sources complement to each other, which are applied to different stages of the training process. Particularly, the **unsupervised data** is curated by extracting the rich-semantic structures, e.g., title-body, title-abstract, instruction-output, etc., within a wide variety of multi-lingual corpora, including Wikipedia, S2ORC (Lo et al., 2020), xP3 (Muennighoff et al., 2022), mC4 (Raffel et al., 2019), and CC-News (Hamborg et al., 2017). Besides, the well-curated data from MTP (Xiao et al., 2023) is directly incorporated. To learn the unified embedding space for cross-lingual semantic matching, the parallel sentences are introduced from two translation datasets, NLLB (NLLB Team et al., 2022) and CCMatrix (Schwenk et al., 2021). The raw data is filtered to remove potential bad contents and low-relevance samples. In total, it brings in *1.2 billion* text pairs of *194 languages* and *2655 cross-lingual* correspondences.

Data Source	Language	Size
Unsupervised Data		
MTP	EN, ZH	291.1M
S2ORC, Wikipeida	EN	48.3M
xP3, mC4, CC-News	Multi-Lingual	488.4M
NLLB, CCMatrix	Cross-Lingual	391.3M
CodeSearchNet	Text-Code	344.1K
Total	–	1.2B
Fine-tuning Data		
MS MARCO, HotpotQA, NQ, NLI, etc.	EN	1.1M
DuReader, T^2-Ranking, NLI-zh, etc.	ZH	386.6K
MIRACL, Mr.TyDi	Multi-Lingual	88.9K
MultiLongDoc	Multi-Lingual	41.4K

Table 1: Specification of training data.

In addition, we collect relatively small but diverse and high-quality **fine-tuning data** from labeled corpora. For English, we incorporate eight datasets, including HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), NQ (Kwiatkowski et al., 2019), MS MARCO (Nguyen et al., 2016),

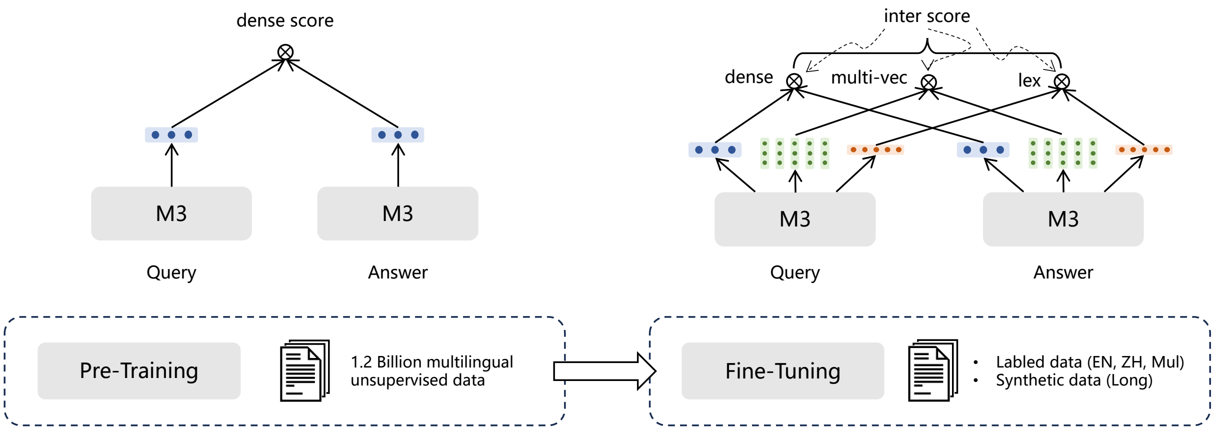

Figure 2: Multi-stage training process of M3-Embedding with self-knowledge distillation.

COLIEE (Kim et al., 2022), PubMedQA (Jin et al., 2019), SQuAD (Rajpurkar et al., 2016), and NLI data collected by SimCSE (Gao et al., 2021b). For Chinese, we incorporate seven datasets, including DuReader (He et al., 2017), mMARCO-ZH (Bonifacio et al., 2021), T^2-Ranking (Xie et al., 2023), LawGPT[1], CMedQAv2 (Zhang et al., 2018), NLI-zh[2], and LeCaRDv2 (Li et al., 2023). For other languages, we leverage the training data from Mr. Tydi (Zhang et al., 2021b) and MIRACL (Zhang et al., 2023c).

Finally, we generate **synthetic data** to mitigate the shortage of long document retrieval tasks and introduce extra multi-lingual fine-tuning data (denoted as MultiLongDoc). Specifically, we sample lengthy articles from Wiki and MC4 datasets and randomly choose paragraphs from them. Then we use GPT-3.5 to generate questions based on these paragraphs. The generated question and the sampled article constitute a new text pair to the fine-tuning data. Detailed specifications are presented in Appendix A.2.

3.2 Hybrid Retrieval

M3-Embedding unifies all three common retrieval functionalities of the embedding model, i.e. dense retrieval, lexical (sparse) retrieval, and multi-vector retrieval. The formulations are presented as follows.

• **Dense retrieval**. The input query q is transformed into the hidden states $\mathbf{H_q}$ based on a text encoder. We use the normalized hidden state of the special token "[CLS]" for the representation of the query: $e_q = norm(\mathbf{H_q}[0])$. Similarly, we can get the embedding of passage p as $e_p = norm(\mathbf{H_p}[0])$. Thus, the relevance score between query and passage is measured by the inner product between the two embeddings e_q and e_p: $s_{dense} \leftarrow \langle e_p, e_q \rangle$.

• **Lexical Retrieval**. The output embeddings are also used to estimate the importance of each term to facilitate lexical retrieval. For each term t within the query (a term is corresponding to a token in our work), the term weight is computed as $w_{q_t} \leftarrow \mathsf{Relu}(\mathbf{W}_{lex}^T \mathbf{H_q}[i]))$, where $\mathbf{W}_{lex} \in \mathcal{R}^{d\times 1}$ is the matrix mapping the hidden state to a float number. If a term t appears multiple times in the query, we only retain its max weight. We use the same way to compute the weight of each term in the passage. Based on the estimation term weights, the relevance score between query and passage is computed by the joint importance of the co-existed terms (denoted as $q \cap p$) within the query and passage: $s_{lex} \leftarrow \sum_{t\in q\cap p}(w_{q_t} * w_{p_t})$.

• **Multi-Vector Retrieval**. As an extension of dense retrieval, the multi-vector method makes use of the entire output embeddings for the representation of query and passage: $E_q = norm(\mathbf{W}_{mul}^T \mathbf{H_q})$, $E_p = norm(\mathbf{W}_{mul}^T \mathbf{H_p})$, where $\mathbf{W}_{mul} \in \mathbb{R}^{d\times d}$ is the learnable projection matrix. Following ColBert(Khattab and Zaharia, 2020), we use late-interaction to compute the fine-grained relevance score: $s_{mul} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \max_{j=1}^{M} E_q[i] \cdot E_p^T[j]$; N and M are the lengths of query and passage.

Thanks to the multi-functionality of the embedding model, the retrieval process can be conducted in a **hybrid process**. First of all, the candidate results can be individually retrieved by each of the methods (the multi-vector method can be exempted from this step due to its heavy cost). Then, the final retrieval result is re-ranked based on the integrated relevance score: $s_{rank} \leftarrow s_{dense} + s_{lex} + s_{mul}$.

1. https://github.com/LiuHC0428/LAW-GPT
2. https://huggingface.co/datasets/shibing624/nli-zh-all

3.3 Self-Knowledge Distillation

The embedding model is trained to discriminate the positive samples from the negative ones. For each of the retrieval methods, it is expected to assign a higher score for the query's positive samples compared with the negative ones. Therefore, the training process is conducted to minimize the InfoNCE loss, whose general form is presented by the following loss function:

$$\mathcal{L} = -\log \frac{\exp(s(q, p^*)/\tau)}{\sum_{p\in\{p^*, P'\}} \exp(s(q, p)/\tau)}. \tag{1}$$

Here, p^* and P' stand for the positive and negative samples to the query q; $s(\cdot)$ is any of the functions within $\{s_{dense}(\cdot), s_{lex}(\cdot), s_{mul}(\cdot)\}$.

The training objectives of different retrieval methods can be mutually conflicting with each their. Therefore, the native multi-objective training can be unfavorable to the embedding's quality. To facilitate the optimization of multiple retrieval functions, we propose to unify the training process on top of **self-knowledge distillation**. Particularly, based on the principle of ensemble learning (Bühlmann, 2012), the predictions from different retrieval methods can be integrated as a more accurate relevance score given their heterogeneous nature. In the simplest form, the integration can just be the sum-up of different prediction scores:

$$s_{inter} \leftarrow s_{dense} + s_{lex} + s_{mul}. \tag{2}$$

In previous studies, the training quality of embedding model can benefit from knowledge distillation, which takes advantage of fine-grained soft labels from another ranking model (Hofstätter et al., 2021). In this place, we simply employ the integration score s_{inter} as the teacher, where the loss function of each retrieval method is modified as:

$$\mathcal{L}'_* \leftarrow -p(s_{inter}) * \log p(s_*). \tag{3}$$

Here, $p(\cdot)$ is the softmax activation; s_* is any of the members within s_{dense}, s_{lex}, and s_{mul}. We further integrate and normalize the modified loss function:

$$\mathcal{L}' \leftarrow \left(\mathcal{L}'_{dense} + \mathcal{L}'_{lex} + \mathcal{L}'_{mul}\right)/3. \tag{4}$$

Finally, we derive the final loss function for self-knowledge distillation with the linear combination of $\mathcal{L}$ and $\mathcal{L}'$: $\mathcal{L}_{final} \leftarrow \mathcal{L} + \mathcal{L}'$.

The overall training process is a multi-stage workflow (Figure 2). We use an XLM-RoBERTa (Conneau et al., 2020) model pre-trained further through the RetroMAE (Xiao et al., 2022) method as the base text encoder. Firstly, the text encoder is pre-trained with the massive unsupervised data, where only the dense retrieval is trained in the basic form of contrastive learning. The self-knowledge distillation is applied to the second stage, where the embedding model is fine-tuned to establish the three retrieval functionalities. Both labeled and synthetic data are used in this stage, where hard negative samples are introduced for each query following the ANCE method (Xiong et al., 2020). Detailed processing is presented in Appendix B.1.

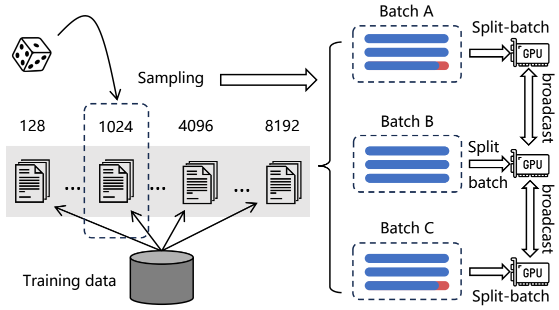

Figure 3: **Efficient Batching.** (Data is grouped and sampled by length. Gradient-checkpointing and cross-GPU broadcasting are enabled to save memory.)

3.4 Efficient Batching

The embedding model needs to learn from diverse and massive multi-lingual data to fully capture the general semantic of different languages. It also needs to keep the batch size as large as possible (where a huge amount of in-batch negatives can be leveraged) so as to ensure the discriminativeness of text embeddings. Given the limitations on GPU's memory and computation power, people usually truncate the input data into short sequences for high throughput of training and a large batch size. However, the common practice is not a feasible option for M3-Embedding because it needs to learn from both short and long-sequence data to effectively handle the input of different granularities. In our work, we improve the training efficiency by optimizing the batching strategy, which enables high training throughput and large batch sizes.

Particularly, the training data is pre-processed by being grouped by sequence length. When producing a mini-batch, the training instances are sampled from the same group. Due to the similar sequence lengths, it significantly reduces sequence padding (marked in red) and facilitates a more effective utilization of GPUs. Besides, when sampling the

Model	Avg	ar	bn	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	zh	de	yo
Baselines (*Prior Work*)																			
BM25	31.9	39.5	48.2	26.7	7.7	28.7	45.8	11.5	35.0	29.7	31.2	37.1	25.6	35.1	38.3	49.1	17.5	12.0	56.1
mDPR	41.8	49.9	44.3	39.4	47.8	48.0	47.2	43.5	38.3	27.2	43.9	41.9	40.7	29.9	35.6	35.8	51.2	49.0	39.6
mContriever	43.1	52.5	50.1	36.4	41.8	21.5	60.2	31.4	28.6	39.2	42.4	48.3	39.1	56.0	52.8	51.7	41.0	40.8	41.5
$\text{mE5}_{\text{large}}$	65.4	76.0	75.9	52.9	52.9	59.0	77.8	54.5	62.0	52.9	70.6	66.5	67.4	74.9	84.6	80.2	56.0	56.4	56.5
$\text{E5}_{\text{mistral-7b}}$	62.2	73.3	70.3	57.3	52.2	52.1	74.7	55.2	52.1	52.7	66.8	61.8	67.7	68.4	73.9	74.0	54.0	54.0	58.8
OpenAI-3	54.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
M3-Embedding (*Our Work*)																			
Dense	67.8	78.4	80.0	56.9	55.5	57.7	78.6	57.8	59.3	56.0	72.8	69.9	70.1	78.6	86.2	82.6	61.7	56.8	60.7
Sparse	53.9	67.1	68.7	43.7	38.8	45.2	65.3	35.5	48.2	48.9	56.3	61.5	44.5	57.9	79.0	70.9	36.3	32.2	70.0
Multi-vec	69.0	79.6	81.1	59.4	57.2	58.8	80.1	59.0	61.4	58.2	74.5	71.2	71.2	79.0	87.9	83.0	62.7	57.9	60.4
Dense+Sparse	68.9	79.6	80.7	58.8	57.5	59.2	79.7	57.6	62.8	58.3	73.9	71.3	69.8	78.5	87.2	83.1	62.5	57.6	**61.8**
All	**70.0**	**80.2**	**81.5**	**59.8**	**59.2**	**60.3**	**80.4**	**60.7**	**63.2**	**59.1**	**75.2**	**72.2**	**71.7**	**79.6**	**88.2**	**83.8**	**63.9**	**59.8**	61.5

Table 2: Multi-lingual retrieval performance on the MIRACL dev set (measured by nDCG@10).

training data for different GPUs, the random seed is always fixed, which ensures the load balance and minimizes the waiting time in each training step. Besides, when handling long-sequence training data, the mini-batch is further divided into sub-batches, which takes less memory footprint. We iteratively encode each sub-batch using gradient checkpointing (Chen et al., 2016) and gather all generated embeddings. This method can significantly increase the batch size. For example, when processing text with a length of 8192, the batch size can be increased by more than 20 times. For more details please refer to Appendx B.3. Finally, the embeddings from different GPUs are broadcasted, allowing each device to obtain all embeddings in the distributed environment, which notably expands the scale of in-bath negative samples.

However, users may lack sufficient computational resources or data to train a long-text model. Therefore, we also propose an MCLS strategy to enhance the model's long-text capabilities without the need for training. This strategy leverages multiple CLS tokens to capture text semantics, applied during inference. Refer to Appendix B.2 for more details.

4 Experiment

In the following part, we evaluate our model on three tasks: multi-lingual retrieval, cross-lingual retrieval, and long-doc retrieval.

4.1 Multi-Lingual Retrieval

We evaluate the multi-lingual retrieval performance with MIRACL (Zhang et al., 2023c), which consists of ad-hoc retrieval tasks in 18 languages. Each task is made up of query and passage presented in the same language. Following the official benchmark, we evaluate our method using Pyserini (Lin et al., 2021), and use nDCG@10 as the primary evaluation metric (Recall@100 is also measured and reported in Appendix C.1). We incorporate the following baselines in our experiment: the lexical retrieval method: BM25 (Robertson and Zaragoza, 2009); the dense retrieval methods: mDPR[3] (Zhang et al., 2023b), mContriever[4] (Izacard et al., 2022), $\text{mE5}_{\text{large}}$ (Wang et al., 2022) and $\text{E5}_{\text{mistral-7b}}$ (Wang et al., 2023). To make the BM25 and M3 more comparable, in the experiment, we use the same tokenizer as M3 (i.e., the tokenizer of XLM-Roberta) for BM25. Using the same vocabulary from XLM-Roberta can also ensure that both approaches have the same retrieval latency. The results of BM25 with different tokenizers are shown in Appendix C.2. We also make a comparison with Text-Embedding-3-Large(abbreviated as OpenAI-3), which was recently released by OpenAI[5].

We can make the following observations according to the experiment result in Table 2. Firstly, M3-Embedding already achieves a superior retrieval performance with only its dense retrieval functionality (denoted as *Dense*). It not only outperforms other baseline methods in the average performance, but also maintains a consistent empirical advantage in most of individual languages. Even compared with $\text{E5}_{\text{mistral-7b}}$, which leverages a much larger Mistral-7B model as the text encoder and specifically trained with English data, our method is able to produce a similar result in English and notably higher results in the other languages. Besides, the sparse retrieval functionality (denoted as *Sparse*) is also effectively trained by M3-Embedding, as

3. https://huggingface.co/castorini/mdpr-tied-pft-msmarco
4. https://huggingface.co/facebook/mcontriever-msmarco
5. https://platform.openai.com/docs/guides/embeddings

	Baselines (*Prior Work*)						M3-Embedding (*Our Work*)				
	BM25	mDPR	mContriever	$\text{mE5}_{\text{large}}$	$\text{E5}_{\text{mistral-7b}}$	OpenAI-3	Dense	Sparse	Multi-vec	Dense+Sparse	All
ar	18.9	48.2	58.2	68.7	59.6	65.6	71.1	23.5	71.4	71.1	**71.5**
da	49.3	67.4	73.9	77.4	**77.8**	73.6	77.2	55.4	77.5	77.4	77.6
de	35.4	65.8	71.7	76.9	**77.0**	73.6	76.2	43.3	76.3	76.4	76.3
es	43.4	66.8	72.6	76.4	**77.4**	73.9	76.4	50.6	76.6	76.7	76.9
fi	46.3	56.2	70.2	74.0	72.0	72.7	75.1	51.1	75.3	75.3	**75.5**
fr	45.3	68.2	72.8	75.5	**78.0**	74.1	76.2	53.9	76.4	76.6	76.6
he	26.9	49.7	63.8	69.6	47.2	58.1	72.4	31.1	72.9	72.5	**73.0**
hu	38.2	60.4	69.7	74.7	**75.0**	71.2	74.7	44.6	74.6	74.9	**75.0**
it	45.2	66.0	72.3	76.8	**77.1**	73.6	76.0	52.5	76.4	76.3	76.5
ja	24.5	60.3	64.8	71.5	65.1	71.9	75.0	31.3	75.1	75.0	**75.2**
km	27.8	29.5	26.8	28.1	34.3	33.9	68.6	30.1	69.1	68.8	**69.2**
ko	27.9	50.9	59.7	68.1	59.4	63.9	71.6	31.4	71.7	71.6	**71.8**
ms	55.9	65.5	74.1	76.3	77.2	73.3	77.2	62.4	**77.4**	**77.4**	**77.4**
nl	56.2	68.2	73.7	77.8	**79.1**	74.2	77.4	62.4	77.6	77.7	77.6
no	52.1	66.7	73.5	77.3	76.6	73.3	77.1	57.9	77.2	**77.4**	77.3
pl	40.8	63.3	71.6	76.7	**77.1**	72.7	76.3	46.1	76.5	76.3	76.6
pt	44.9	65.5	72.0	73.5	**77.5**	73.7	76.3	50.9	76.4	76.5	76.4
ru	33.2	62.7	69.8	**76.8**	75.5	72.0	76.2	36.9	76.4	76.2	76.5
sv	54.6	66.9	73.2	77.6	**78.3**	74.0	76.9	59.6	77.2	77.4	77.4
th	37.8	53.8	66.9	76.0	67.4	65.2	76.4	42.0	76.5	76.5	**76.6**
tr	45.8	59.1	71.1	74.3	73.0	71.8	75.6	51.8	75.9	**76.0**	**76.0**
vi	46.6	63.4	70.9	75.4	70.9	71.1	76.6	51.8	76.7	76.8	**76.9**
zh_cn	31.0	63.7	68.1	56.6	69.3	70.7	74.6	35.4	74.9	74.7	**75.0**
zh_hk	35.0	62.8	68.0	58.1	65.1	69.6	73.8	39.8	74.1	74.0	**74.3**
zh_tw	33.5	64.0	67.9	58.1	65.8	69.7	73.5	37.7	73.5	**73.6**	**73.6**
Avg	39.9	60.6	67.9	70.9	70.1	69.5	75.1	45.3	75.3	75.3	**75.5**

Table 3: Cross-lingual retrieval performance on MKQA (measured by Recall@100).

it outperforms the typical BM25 methods in all languages. We can also observe the additional improvement from multi-vector retrieval[6] (denoted as <u>*Mult-vec*</u>), which relies on fine-grained interactions between query and passage's embeddings to compute the relevance score. Finally, the collaboration of dense and sparse method, e.g., Dense+Sparse[7], leads to a further improvement over each individual method; and the collaboration of all three methods[8] (denoted as <u>*All*</u>) brings forth the best performance.

4.2 Cross-Lingual Retrieval

We make evaluation for the cross-lingual retrieval performance with the MKQA benchmark (Longpre et al., 2021), which includes queries in 25 non-English languages. For each query, it needs to retrieve the ground-truth passage from the English Wikipedia corpus. In our experiment, we make use of the well-processed corpus offered by the BEIR[9] (Thakur et al., 2021). Following the previous study (Karpukhin et al., 2020), we report Recall@100 as the primary metric (Recall@20 is reported as an auxiliary metric in the Appendix).

The experiment result is shown in Table 3. Similar to our observation in multi-lingual retrieval, M3-Embedding continues to produce a superior performance, where it notably outperforms other baseline methods purely with its dense retrieval functionality (Dense). The collaboration of different retrieval methods brings in further improvements, leading to the best empirical performance of cross-lingual retrieval. Besides, we can also observe the following interesting results which are unique to this benchmark. Firstly, the performance gaps are not as significant as MIRACL, where competitive baselines like $\text{E5}_{\text{mistral-7b}}$ is able to produce similar or even better results on some of the testing languages. However, the baselines are prone to bad performances in many other languages, especially the low-resource languages, such as ar, km, he, etc. In contrast, M3-Embedding maintains relatively stable performances in all languages, which can largely be attributed to its pre-training over comprehensive unsupervised data. Secondly, although M3-Embedding (Sparse) is still better than BM25, it performs badly compared with other methods. This

6. Mult-vec is used to re-rank the top-200 candidates from Dense for efficient processing.
7. Retrieve the top-1000 candidates with dense and sparse method; then re-rank using the sum of two scores.
8. Re-rank based on the sum of all three scores.
9. https://huggingface.co/datasets/BeIR/nq

	Max Length	Avg	ar	de	en	es	fr	hi	it	ja	ko	pt	ru	th	zh
Baselines (*Prior Work*)															
BM25	8192	53.6	45.1	52.6	57.0	78.0	75.7	43.7	70.9	36.2	25.7	82.6	61.3	33.6	34.6
mDPR	512	23.5	15.6	17.1	23.9	34.1	39.6	14.6	35.4	23.7	16.5	43.3	28.8	3.4	9.5
mContriever	512	31.0	25.4	24.2	28.7	44.6	50.3	17.2	43.2	27.3	23.6	56.6	37.7	9.0	15.3
$\text{mE5}_{\text{large}}$	512	34.2	33.0	26.9	33.0	51.1	49.5	21.0	43.1	29.9	27.1	58.7	42.4	15.9	13.2
$\text{E5}_{\text{mistral-7b}}$	8192	42.6	29.6	40.6	43.3	70.2	60.5	23.2	55.3	41.6	32.7	69.5	52.4	18.2	16.8
text-embedding-ada-002	8191	32.5	16.3	34.4	38.7	59.8	53.9	8.0	46.5	28.6	20.7	60.6	34.8	9.0	11.2
jina-embeddings-v2-base-en	8192	-	-	-	37.0	-	-	-	-	-	-	-	-	-	-
M3-Embedding (*Our Work*)															
Dense	8192	52.5	47.6	46.1	48.9	74.8	73.8	40.7	62.7	50.9	42.9	74.4	59.5	33.6	26.0
Sparse	8192	62.2	58.7	53.0	62.1	87.4	82.7	49.6	74.7	53.9	47.9	85.2	72.9	40.3	40.5
Multi-vec	8192	57.6	56.6	50.4	55.8	79.5	77.2	46.6	66.8	52.8	48.8	77.5	64.2	39.4	32.7
Dense+Sparse	8192	64.8	63.0	56.4	**64.2**	**88.7**	**84.2**	**52.3**	**75.8**	58.5	53.1	**86.0**	**75.6**	42.9	**42.0**
All	8192	**65.0**	**64.7**	**57.9**	63.8	86.8	83.9	52.2	75.5	**60.1**	**55.7**	85.4	73.8	**44.7**	40.0
M3-w.o.long															
Dense-w.o.long	8192	41.2	35.4	35.2	37.5	64.0	59.3	28.8	53.1	41.7	29.8	63.5	51.1	19.5	16.5
Dense-w.o.long (MCLS)	8192	45.0	37.9	43.3	41.2	67.7	64.6	32.0	55.8	43.4	33.1	67.8	52.8	27.2	18.2

Table 4: Evaluation of multilingual long-doc retrieval on the MLDR test set (measured by nDCG@10).

Model	Max Length	nDCG@10
Baselines (*Prior Work*)		
mDPR	512	16.3
mContriever	512	23.3
$\text{mE5}_{\text{large}}$	512	24.2
bge-large-en-v1.5	512	27.3
$\text{E5}_{\text{mistral-7b}}$	8192	49.9
text-embedding-ada-002	8191	41.1
text-embedding-3-large	8192	51.6
jina-embeddings-v2-base-en	8192	39.4
M3-Embedding (*Our Work*)		
Dense	8192	48.7
Sparse	8192	57.5
Multi-vec	8192	55.4
Dense+Sparse	8192	60.1
All	8192	**61.7**

Table 5: Evaluation on NarrativeQA (nDCG@10).

is because there are only very limited co-existed terms for cross-lingual retrieval as the query and passage are presented in different languages.

4.3 Multilingual Long-Doc Retrieval

We evaluate the retrieval performance with longer sequences with two benchmarks: MLDR (Multilingual Long-Doc Retrieval), which is curated by the multilingual articles from Wikipedia, Wudao and mC4 (see Table 8), and NarrativeQA[10] (s Koˇ ciský et al., 2018; Günther et al., 2024), which is only for English. In addition to the previous baselines, we further introduce JinaEmbeddingv2[11] (Günther et al., 2024), text-embedding-ada-002 and text-embedding-3-large from OpenAI given their outstanding long-doc retrieval capability.

10. Using the evaluation pipeline from (Günther et al., 2024)
11. https://huggingface.co/jinaai/jina-embeddings-v2-base-en

The evaluation result on MLDR is presented in Table 4. Interestingly, M3 (Sparse) turns out to be a more effective method for long document retrieval, which achieves another about 10 points improvement over the dense method. Besides, the multi-vector retrieval is also impressive, which brings 5.1+ points improvement over M3 (Dense). Finally, the combination of different retrieval methods leads to a remarkable average performance of 65.0.

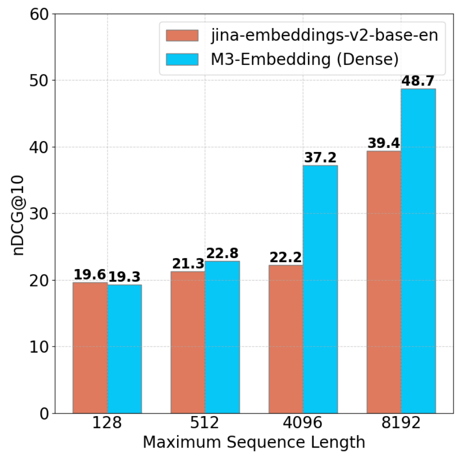

Figure 4: NarrativeQA with variant sequence length.

To explore the reason for M3-Embedding's competitiveness in long-document retrieval, we perform the ablation study by removing the long document data from the fine-tuning stage (denoted as w.o. long). After this modification, the dense method, i.e. Dense-w.o.long, can still outperform the majority of baselines, which indicates that its empirical advantage has been well established during the pre-training stage. We also propose a simple strat-

Model	Avg	ar	bn	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	zh	de	yo
M3-w.skd																			
Dense	67.8	78.4	80.0	56.9	55.5	57.7	78.6	57.8	59.3	56.0	72.8	69.9	70.1	78.6	86.2	82.6	61.7	56.8	60.7
Sparse	53.9	67.1	68.7	43.7	38.8	45.2	65.3	35.5	48.2	48.9	56.3	61.5	44.5	57.9	79.0	70.9	36.3	32.2	70.0
Multi-vec	69.0	79.6	81.1	59.4	57.2	58.8	80.1	59.0	61.4	58.2	74.5	71.2	71.2	79.0	87.9	83.0	62.7	57.9	60.4
M3-w.o.skd																			
Dense	67.2	78.0	79.1	56.4	54.9	57.4	78.3	57.8	58.9	55.1	72.3	68.7	69.5	77.8	85.8	82.5	62.1	55.9	59.9
Sparse	36.7	48.2	52.1	24.3	20.3	25.9	48.6	16.9	30.1	32.1	33.0	43.1	27.1	45.3	63.8	52.0	22.6	16.5	59.4
Multi-vec	67.8	78.7	80.2	57.6	56.1	57.4	79.0	57.9	59.2	57.5	74.0	70.3	70.2	78.6	86.9	82.1	61.0	56.7	57.4

Table 6: Ablation study of self-knowledge distillation with MIRACL (nDCG@10).

egy, MCLS, to address this situation (no data or no GPU resource for document-retrieval fine-tuning). Experimental results indicate that MCLS can significantly improve the performance of document retrieval without training ($41.2 \rightarrow 45.0$).

We make further analysis with NarrativeQA (Table 5), where we have similar observations as MLDR. Besides, with the growing of sequence length, our method gradually expands its advantage over baseline (Figure 4), which reflects its proficiency in handling long inputs.

4.4 Ablation study

Self-knowledge distillation This ablation study is performed to analyze the impact of self-knowledge distillation (skd). Particularly, we disable the distillation processing and have each retrieval method trained independently (denoted as M3-w.o.skd). According to our evaluation on MIRACL (Table 6), the original method, i.e. M3 w.skd, brings in better performances than the ablation method in all settings, i.e., Dense, Sparse, Multi-vec. Notably, the impact is more pronounced for sparse retrieval, which indicates the incompatibility between dense and sparse retrieval methods.

Impact of multi-stage training We also conducted experiments to explore the impact of different stages. *Fine-tuning* indicates fine-tuning directly from the xlm-roberta (Conneau et al., 2020) model, and *RetroMAE+Fine-tuning* refers to fine-tuning on a model trained with RetroMAE (Xiao et al., 2022). Meanwhile, *RetroMAE+Unsup+Fine-tuning* involves fine-tuning on a model trained with RetroMAE and then pre-trained on unsupervised data. The results are summarized in Table 7. We can see that RetroMAE can significantly improve retrieval performance, and pre-training on unsupervised data can further enhance the retrieval ability of the embedding model.

Model (Dense)	MIRACL
Fine-tune	59.3
RetroMAE + Fine-tune	64.8
RetroMAE + Unsup + Fine-tune	67.8

Table 7: Ablation study of multi-stage training with MIRACL (nDCG@10).

5 Conclusion

In this paper, we present M3-Embedding, which achieves notable versatility in supporting multi-lingual retrieval, handling input of diverse granularities, and unifying different retrieval functionalities. We perform comprehensive and high-quality curation of training data, optimize the learning process with self-knowledge distillation, and improve the training through and batch size with efficient batching. The effectiveness of M3-Embedding is verified by our experimental studies, where it leads to superior performances on multi-lingual retrieval, cross-lingual retrieval, and multi-lingual long-document retrieval tasks.

References

Luiz Bonifacio, Vitor Jeronymo, Hugo Queiroz Abonizio, Israel Campiotti, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. 2021. mmarco: A multilingual version of the ms marco passage ranking dataset. *arXiv preprint arXiv:2108.13897*.

Peter Bühlmann. 2012. Bagging, boosting and ensemble methods. *Handbook of computational statistics: Concepts and methods*, pages 985–1022.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–

8451, Online. Association for Computational Linguistics.

Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1533–1536.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021a. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. *arXiv preprint arXiv:2104.07186*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. 2024. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–122.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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

Mi-Young Kim, Juliano Rabelo, Randy Goebel, Masaharu Yoshioka, Yoshinobu Kano, and Ken Satoh. 2022. Coliee 2022 summary: Methods for legal document retrieval and entailment. In *JSAI International Symposium on Artificial Intelligence*, pages 51–67. Springer.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Haitao Li, Yunqiu Shao, Yueyue Wu, Qingyao Ai, Yixiao Ma, and Yiqun Liu. 2023. Lecardv2: A large-scale chinese legal case retrieval dataset. *arXiv preprint arXiv:2310.17609*.

Jimmy Lin and Xueguang Ma. 2021. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *arXiv preprint arXiv:2106.14807*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, pages 2356–2362.

Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2022. *Pretrained transformers for text ranking: Bert and beyond*. Springer Nature.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics, pages 4969–4983, Online. Association for Computational Linguistics.

Shayne Longpre, Yi Lu, and Joachim Daiber. 2021. MKQA: A linguistically diverse benchmark for multilingual open domain question answering. *Trans. Assoc. Comput. Linguistics*, 9:1389–1406.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *choice*, 2640:660.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia-Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. 2022. No language left behind: Scaling human-centered machine translation.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2020. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2010.08191*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. *arXiv preprint arXiv:2110.07367*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Tomáš Ko ˇ ciský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, TBD:TBD.

Holger Schwenk, Guillaume Wenzek, Sergey Edunov, Edouard Grave, Armand Joulin, and Angela Fan. 2021. CCMatrix: Mining billions of high-quality parallel sentences on the web. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6490–6500, Online. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 538–548, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.

Xiaohui Xie, Qian Dong, Bingning Wang, Feiyang Lv, Ting Yao, Weinan Gan, Zhijing Wu, Xiangsheng Li, Haitao Li, Yiqun Liu, et al. 2023. T2ranking: A large-scale chinese benchmark for passage ranking. *arXiv preprint arXiv:2304.03679*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

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

Sha Yuan, Hanyu Zhao, Zhengxiao Du, Ming Ding, Xiao Liu, Yukuo Cen, Xu Zou, Zhilin Yang, and Jie Tang. 2021. Wudaocorpora: A super large-scale chinese corpora for pre-training language models. *AI Open*, 2:65–68.

Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021a. Adversarial retriever-ranker for dense text retrieval. *arXiv preprint arXiv:2110.03611*.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023a. Retrieve anything to augment large language models.

S. Zhang, X. Zhang, H. Wang, L. Guo, and S. Liu. 2018. Multi-scale attentive interaction networks for chinese medical question answer selection. *IEEE Access*, 6:74061–74071.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. 2021b. Mr. tydi: A multi-lingual benchmark for dense retrieval. *arXiv preprint arXiv:2108.08787*.

Xinyu Zhang, Kelechi Ogueji, Xueguang Ma, and Jimmy Lin. 2023b. Toward best practices for training multilingual dense retrieval models. *ACM Transactions on Information Systems*, 42(2):1–33.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023c. MIRACL: A Multilingual Retrieval Dataset Covering 18 Diverse Languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131.

A Details of Datasets

A.1 Collected Data

The language and length distribution (the number of tokens) of the unsupervised data are illustrated in Figure 5.

We observed that for long texts (e.g., the news in cc-news), the initial sentences tend to be summarizing statements, and the model can rely solely on the information presented in these initial sentences to establish relevant relationships. To prevent the model from focusing solely on these starting sentences, we implemented a strategy of randomly shuffling the order of segments within entire texts. Specifically, we divided the text into three segments, shuffled their order randomly, and recombined them. This approach allows relevant text segments to appear randomly at any position within the long sequence. During training, we applied this operation to passages with a probability of 0.2%.

A.2 Synthetic Data

The prompt for GPT3.5 is "You are a curious AI assistant, please generate one specific and valuable question based on the following text. The generated question should revolve around the core content of this text, and avoid using pronouns (e.g., "this"). Note that you should generate only one question, without including additional content:". The details of generated dataset are shown in Table 8.

B Implementation Details

B.1 Experimental Hyperparameters

We adopt a further pre-trained XLM-RoBERTa[12] as the foundational model. We extend the max position to 8192 and update the model via the RetroMAE (Xiao et al., 2022) method. The data comprises Pile (Gao et al., 2020), Wudao (Yuan et al., 2021), and mC4 (Raffel et al., 2019) datasets. We sampled a total of 184 million text samples from these sources, covering 105 languages. The maximum sequence length is 8192 and the learning rate is 7×10^{-5}. The batch size is set to 32 and we accumulate the gradient over 16 steps. Pre-training is conducted on 32 A100(40GB) GPUs for 20,000 steps.

For the pre-training with the massive unsupervised data, the max length of query and passage is set to 512 and 8192, respectively. The learning rate is 5×10^{-5}, the warmup ratio is 0.1 and the weight decay is 0.01. This training process takes 25,000 steps. For training data with different sequence length ranges (e.g., 0-500, 500-1000, etc.), we use different batch sizes. The details are represented in Table 9. The second stage is conducted on 96 A800(80GB) GPUs.

In the fine-tuning stage, we sample 7 negatives for each query. Refer to Table 9 for the batch size. In the initial phase, we employed approximately 6000 steps to perform warm-up on dense embedding, sparse embedding and multi-vectors. Subsequently, we conducted unified training with self-knowledge distillation. These experiments were carried out on 24 A800(80GB) GPUs.

B.2 MCLS Method

The fine-tuning using long text can be constrained due to the absence of long text data or computation resources. In this situation, we propose a simple but effective method: MCLS(Multiple CLS) to enhance the model's ability without fine-tuning on long text. The MCLS method aims to utilize multiple CLS tokens to jointly capture the semantics of long texts. Specifically, we insert a CLS token for every fixed number of tokens (in our experiments, we insert a "[CLS]" for each 256 tokens), and each CLS token can capture semantic information from its neighboring tokens. Ultimately, the final text embedding is obtained by averaging the last hidden states of all CLS tokens.

B.3 Split-batch Method

Algorithm 1 Pseudocode of split-batch.

```
# enable gradient-checkpointing
BGE-M3.gradient_checkpointing_enable()

embs = []
for batch_data in loader:
    # split the large batch into multiple sub-batch
    for sub_batch_data in batch_data:
        sub_emb = BGE-M3(sub_batch_data)
        # only collect the embs
        embs.append(sub_emb)

# concatenate the outputs to get final embeddings
embs = cat(embs)
```

Algorithm 1 provides the pseudo-code of the split-batch strategy. For the current batch, we partition it into multiple smaller sub-batches. For each sub-batch we utilize the model to generate embeddings, discarding all intermediate activations via gradient checkpointing during the forward pass. Finally, we gather the encoded results from all sub-batch, and obtain the embeddings for the current batch. It is crucial to enable the gradient-checkpointing

12. https://huggingface.co/FacebookAI/xlm-roberta-large

Language	Source	#train	#dev	#test	#cropus	Avg. Length of Docs
ar	Wikipedia	1,817	200	200	7,607	9,428
de	Wikipedia, mC4	1,847	200	200	10,000	9,039
en	Wikipedia	10,000	200	800	200,000	3,308
es	Wikipedia, mC4	2,254	200	200	9,551	8,771
fr	Wikipedia	1,608	200	200	10,000	9,659
hi	Wikipedia	1,618	200	200	3,806	5,555
it	Wikipedia	2,151	200	200	10,000	9,195
ja	Wikipedia	2,262	200	200	10,000	9,297
ko	Wikipedia	2,198	200	200	6,176	7,832
pt	Wikipedia	1,845	200	200	6,569	7,922
ru	Wikipedia	1,864	200	200	10,000	9,723
th	mC4	1,970	200	200	10,000	8,089
zh	Wikipedia, Wudao	10,000	200	800	200,000	4,249
Total	–	41,434	2,600	3,800	493,709	4,737

Table 8: Specifications of MultiLongDoc dataset.

Length Range	Batch Size	
	Unsupervised	Fine-tuning
0-500	67,200	1,152
500-1000	54,720	768
1000-2000	37,248	480
2000-3000	27,648	432
3000-4000	21,504	336
4000-5000	17,280	336
5000-6000	15,072	288
6000-7000	12,288	240
7000-8192	9,984	192

Table 9: Detailed total batch size used in training for data with different sequence length ranges.

strategy; otherwise, the intermediate activations for each sub-batch will continuously accumulate, ultimately occupying the same amount of GPU memory as traditional methods.

In Table 10, we investigate the impact of split-batch on batch size. It can be observed that, with the split-batch enabled, there is a significant increase in batch size. Simultaneously, the increase becomes more pronounced with longer text lengths, and in the case of a length of 8192, enabling split-batch results in a growth of batch size by over 20 times.

Use Split-batch	Max Length		
	1024	4096	8192
×	262	25	6
√	855	258	130

Table 10: Maximum batch size per device under different experimental settings.

C More Results

C.1 Additional Resutls

In this section, we present additional evaluation results on the MIRACL and MKQA benchmarks. As shown in Table 12 and 13, M3-Embedding outperforms all baselines on average.

C.2 Different Tokenizer for BM25

We investigate the impact of different tokenizers on the BM25 method, and the results are shown in Table 11. We can observe that:

- Using the Analyzer from Lucene[13] can significantly enhance the effectiveness of BM25. Lucene analyzer includes multiple steps typically including tokenization, stemming, stopword removal, etc, achieving better results than directly using the tokenzier of xlm-roberta. Additionally, it's worth noting that the vocabulary size of the tokenizer from xlm-roberta is limited, resulting in fewer unique tokens after encoding documents (for example, on the MLDR dataset, the tokenizer of xlm-roberta produces 1056 unique terms per article, while Lucene's analyzer generates 1451 unique terms, which is over 37% more and will increase retrieval latency).

- M3 outperforms BM25 models using the same tokenizer on all datasets, indicating that the learned weights are significantly better than the weights calculated by BM25.

- The sparse retrieval of M3 outperforms BM25 on Miracl and MKQA datasets. In long document retrieval (MLDR), M3's sparse doesn't

13. https://github.com/apache/lucene/tree/main/lucene/analysis/

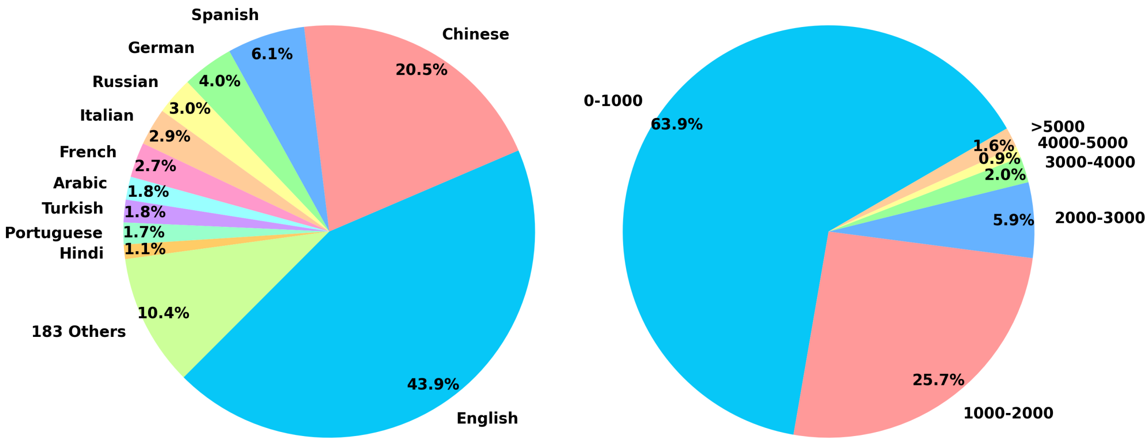

Figure 5: Language and sequence length distribution of unsupervised data

method	tokenizer	Miracl	MKQA	MLDR
BM25	Analyzer	38.5	40.9	64.1
BM25	xlm-roberta	31.9	39.9	53.6
M3.Sparse	xlm-roberta	53.9	45.3	62.2
M3.All	xlm-roberta	70.0	75.5	65.0

Table 11: Comparison with the BM25 methods using different tokenizers.

surpass BM25 but achieves competitive performance. This suggests that BM25 remains a highly competitive baseline model. Exploring tokenizers that perform better for sparse representation is a worthwhile topic for future research.

Model	Avg	ar	bn	en	es	fa	fi	fr	hi	id	ja	ko	ru	sw	te	th	zh	de	yo
Baselines (*Prior Work*)																			
BM25	67.3	78.7	90.0	63.6	25.4	68.1	81.2	50.2	73.8	71.8	73.6	70.1	56.4	69.9	73.3	87.5	55.1	42.8	80.1
mDPR	79.0	84.1	81.9	76.8	86.4	89.8	78.8	91.5	77.6	57.3	82.5	73.7	79.7	61.6	76.2	67.8	94.4	89.8	71.5
mContriever	84.9	92.5	92.1	79.7	84.1	65.4	95.3	82.4	64.6	80.2	87.8	87.5	85.0	91.1	96.1	93.6	90.3	84.1	77.0
$\text{mE5}_{\text{large}}$	92.8	97.3	98.2	87.6	89.1	**92.9**	98.1	90.6	93.9	87.9	97.1	93.4	95.5	96.7	99.2	98.9	93.3	90.6	69.4
$\text{E5}_{\text{mistral-7b}}$	91.3	96.0	96.0	90.2	87.5	88.0	96.7	92.8	89.9	88.4	95.1	89.4	95.0	95.5	95.1	96.5	90.1	88.6	73.3
M3-Embedding (*Our Work*)																			
Dense	93.8	97.6	98.7	90.7	90.1	89.6	97.9	93.1	94.2	90.5	97.5	95.5	95.9	97.2	**99.4**	99.1	95.7	90.8	74.2
Sparse	85.6	92.0	96.6	81.5	71.9	87.0	91.5	73.2	87.2	84.9	92.3	91.8	77.0	85.1	98.0	95.2	72.8	69.0	**92.9**
Multi-vec	94.5	97.8	**98.9**	91.6	91.3	90.5	98.2	**95.4**	94.9	92.5	**98.0**	95.9	96.5	97.3	**99.4**	**99.2**	96.2	**92.3**	74.6
Dense+Sparse	94.4	**98.0**	**98.9**	**92.4**	91.5	91.0	**98.4**	93.9	**95.3**	**92.6**	97.5	95.6	96.6	**97.6**	99.1	99.0	95.7	90.8	75.4
All	**94.6**	**98.0**	**98.9**	92.1	**91.8**	91.2	**98.4**	95.0	94.9	92.4	97.9	**96.0**	**96.7**	97.2	**99.4**	**99.2**	**96.5**	92.2	74.6

Table 12: Recall@100 on the dev set of the MIRACL dataset for multilingual retrieval in all 18 languages.

	Baselines (*Prior Work*)						M3-Embedding (*Our Work*)				
	BM25	mDPR	mContriever	$\text{mE5}_{\text{large}}$	$\text{E5}_{\text{mistral-7b}}$	OpenAI-3	Dense	Sparse	Multi-vec	Dense+Sparse	All
ar	13.4	33.8	43.8	59.7	47.6	55.1	61.9	19.5	62.6	61.9	**63.0**
da	36.2	55.7	63.3	71.7	**72.3**	67.6	71.2	45.1	71.7	71.3	72.0
de	23.3	53.2	60.2	**71.2**	70.8	67.6	69.8	33.2	69.6	70.2	70.4
es	29.8	55.4	62.3	70.8	**71.6**	68.0	69.8	40.3	70.3	70.2	70.7
fi	33.2	42.8	58.7	67.7	63.6	65.5	67.8	41.2	68.3	68.4	**68.9**
fr	30.3	56.5	62.6	69.5	**72.7**	68.2	69.6	43.2	70.1	70.1	70.8
he	16.1	34.0	50.5	61.4	32.4	46.3	63.4	24.5	64.4	63.5	**64.6**
hu	26.1	46.1	57.1	68.0	**68.3**	64.0	67.1	34.5	67.3	67.7	67.9
it	31.5	53.8	62.0	71.2	**71.3**	67.6	69.7	41.5	69.9	69.9	70.3
ja	14.5	46.3	50.7	63.1	57.6	64.2	67.0	23.3	67.8	67.1	**67.9**
km	20.7	20.6	18.7	18.3	23.3	25.7	58.5	24.4	59.2	58.9	**59.5**
ko	18.3	36.8	44.9	58.9	49.4	53.9	61.9	24.3	63.2	62.1	**63.3**
ms	42.3	53.8	63.7	70.2	71.1	66.1	71.6	52.5	72.1	71.8	**72.3**
nl	42.5	56.9	63.9	**73.0**	74.5	68.8	71.3	52.9	71.8	71.7	72.3
no	38.5	55.2	63.0	71.1	70.8	67.0	70.7	47.0	71.4	71.1	**71.6**
pl	28.7	50.4	60.9	70.5	**71.5**	66.1	69.4	36.4	70.0	69.9	70.4
pt	31.8	52.5	61.0	66.8	**71.6**	67.7	69.3	40.2	70.0	69.8	70.6
ru	21.8	49.8	57.9	**70.6**	68.7	65.1	69.4	29.2	70.0	69.4	70.0
sv	41.1	54.9	62.7	72.0	**73.3**	67.8	70.5	49.8	71.3	71.5	71.5
th	28.4	40.9	54.4	69.7	57.1	55.2	69.6	34.7	70.5	69.8	**70.8**
tr	33.5	45.5	59.9	67.3	65.5	64.9	68.2	40.9	69.0	69.1	**69.6**
vi	33.6	51.3	59.9	68.7	62.3	63.5	69.6	42.2	70.5	70.2	**70.9**
zh_cn	19.4	50.1	55.9	44.3	61.2	62.7	66.4	26.9	66.7	66.6	**67.3**
zh_hk	23.9	50.2	55.5	46.4	55.9	61.4	65.8	31.2	66.4	65.9	**66.7**
zh_tw	22.5	50.6	55.2	45.9	56.5	61.6	64.8	29.8	65.3	64.9	**65.6**
Avg	28.1	47.9	56.3	63.5	62.4	62.1	67.8	36.3	68.4	68.1	**68.8**

Table 13: Recall@20 on MKQA dataset for cross-lingual retrieval in all 25 languages.