Towards General Text Embeddings with Multi-stage Contrastive Learning

Zehan Li[1], **Xin Zhang**[1], **Yanzhao Zhang**[1], **Dingkun Long**[1], **Pengjun Xie**[1], **Meishan Zhang**
[1]Alibaba Group
{lizehan.lzh,linzhang.zx,zhangyanzhao.zyz,
dingkun.ldk,pengjun.xpj}@alibaba-inc.com

Abstract

We present GTE, a general-purpose text embedding model trained with multi-stage contrastive learning. In line with recent advancements in unifying various NLP tasks into a single format, we train a unified text embedding model by employing contrastive learning over a diverse mixture of datasets from multiple sources. By significantly increasing the number of training data during both unsupervised pre-training and supervised fine-tuning stages, we achieve substantial performance gains over existing embedding models. Notably, even with a relatively modest parameter count of 110M, GTE_{base} outperforms the black-box embedding API provided by OpenAI and even surpasses 10x larger text embedding models on the massive text embedding benchmark. Furthermore, without additional fine-tuning on each programming language individually, our model outperforms previous best code retrievers of similar size by treating code as text. In summary, our model achieves impressive results by effectively harnessing multi-stage contrastive learning, offering a powerful and efficient text embedding model with broad applicability across various NLP and code-related tasks.[1]

1 Introduction

Text embeddings have became an indispensable component in many natural language processing tasks, such as text classification, text retrieval, question answering and dialogue systems (Karpukhin et al., 2020; Humeau et al., 2020; Choi et al., 2021; Izacard et al., 2022a; Long et al., 2022a; Rajapakse, 2023). These embedding models represent texts using low-dimensional vectors and capture their similarity through vector operations. The emergence of recent large language models (LLMs) (Radford et al., 2018; Touvron et al., 2023; OpenAI, 2023) has generated considerable interest in retrieval-augmented systems based on text embedding models that integrate the reasoning and comprehension capabilities of LLMs (Izacard et al., 2022b; Ram et al., 2023; Shi et al., 2023). Consequently, there has been a growing focus on general text representation in both industry and academia.

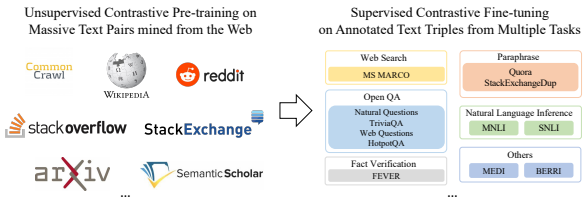


Figure 1: Illustration of the multi-stage contrastive learning pipeline used to train our text embedding model.

The pursuit of developing a unified model to address a multitude of downstream tasks has been long-standing due to the diverse formats, domains and downstream applications of natural language. The emergence of pre-trained language models has further opened up possibilities for training such a universal model. Nonetheless, within the realm of text representation research, previous text embedding models have primarily focused on specific tasks, and their training strategies or models, tailored to a single task, may not perform optimally in other contexts. For example, the text representation model SimCSE (Gao et al., 2021), trained on symmetric text pairs, demonstrates limitations in text retrieval tasks. Similarly, certain text representation models specifically designed for dense retrieval tasks do not exhibit robust performance in sentence textual similarity tasks. Recently, there has been a shift in research focus towards developing more comprehensive models for text representation leveraging large quantities of unlabeled web data through unsupervised contrastive pre-training, coupled with task-specific data, prompts, or instructions to mitigate task conflicts during fine-tuning (Ni et al., 2022a,b; Neelakantan et al., 2022;

[1]The GTE model is publicly available at https://huggingface.co/thenlper/gte-large

Wang et al., 2022b; Su et al., 2023). Additionally, the introduction of benchmarks, such as the Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2023), has established a robust basis for assessing the universality of text representation models. However, a significant limitation in existing research is the reliance on in-house data for pre-training, creating a bottleneck in the utilization of pre-trained model weights or APIs. Furthermore, the formulation of prompts specifically tailored for each task requires extra human effort during implementation (Su et al., 2023).

This work presents a straightforward approach to construct a general text embedding (GTE) model solely using contrastive learning on open-source data, as illustrated in Figure 1. Specifically, we first gather a large-scale dataset comprising unsupervised text pairs extracted from various data sources for contrastive pre-training. Surprisingly, our model, pre-trained on this dataset, exhibits remarkable performance, surpassing BM25 and E5 model (Wang et al., 2022b) in zero-shot text retrieval tasks and surpassing many supervised models in the MTEB benchmark. To further enhance the quality of the learned text representations, we obtain high-quality text pairs with human labels from multiple sources for contrastive fine-tuning. After supervised fine-tuning, our 110M BERT-based (Devlin et al., 2019) model already outperforms the current commercial embedding API of OpenAI and ranks highly in the MTEB benchmark. Furthermore, since our model is trained using code data as well, we evaluate its code search capabilities on the CodeSearchNet benchmark, which encompasses six programming languages. Notably, even without language-specific fine-tuning on each subset, our model significantly outperforms state-of-the-art code retrievers of similar size that have been fine-tuned for each programming language.

In the rest of this paper, we provide a detailed account of the data sources and training configurations employed. Subsequently, we present the evaluation results on widely recognized text embedding benchmarks and compare them with the performance of previous state-of-the-art baselines that were specifically optimized for each individual task. Our model consistently demonstrates superior performance or, at the very least, comparable results to those achieved by larger models, owing to its incorporation of a more diverse mixture of training datasets. We aspire for our model to serve as a robust baseline for the research community investigating text and code embedding.

2 Related Work

Text embeddings serve as low-dimensional vector representations for texts of varying lengths and are essential in numerous natural language processing (NLP) tasks. In contrast to high-dimensional and sparse representations such as TF-IDF, dense text embeddings possess the capacity to address the lexical mismatch problem and enhance the efficiency of text retrieval and matching.

Pre-trained language models, exemplified by BERT (Devlin et al., 2019) and GPT (Radford et al., 2018), have demonstrated remarkable success across various NLP tasks. Nonetheless, extracting a high-quality sentence embedding from pre-trained language models poses a significant challenge due to the presence of anisotropic embedding spaces resulting from the masked language modeling objective. To address this issue, subsequent studies have proposed different approaches, including supervised fine-tuning (Reimers and Gurevych, 2019), normalizing flow (Li et al., 2020), normalizing flow (Li et al., 2020), whitening (Su et al., 2021), or unsupervised contrastive learning (Gao et al., 2021). These investigations primarily concentrate on enhancing performance in semantic textual similarity tasks, wherein two sentences exhibit similar formats.

Another line of research focuses on the text retrieval problem, where the query and document typically exhibit an asymmetric relationship. In this context, the dual-encoder architecture necessitates training with both positive and negative pairs. Lee et al. (2019) propose the Inverse Close Task (ICT) as a self-supervised pre-training approach for generating a dense retriever. The ICT method involves cropping a random sentence from a passage to construct pseudo query-document pairs. Additionally, Chang et al. (2020) leverage the link structure within Wikipedia to introduce further supervision signals in the pre-training data. In a similar vein, REALM (Guu et al., 2020) proposes a joint training approach, wherein a dense retriever and a language model are trained concurrently. The learning signal for the language model is derived from masked language modeling, with backpropagation incorporated through the retrieval step. Recent advancements, such as Contriever (Izacard et al., 2022a) and coCondenser (Gao and Callan,

2022), have demonstrated that constructing positive pairs through random passage cropping yields superior results compared to the ICT task. Building upon the ideas presented in (Chang et al., 2020), some researchers have also put forth methods for constructing higher-quality positive pairs using the web link topology for retriever pre-training (Zhou et al., 2022), a technique that proves effective in zero-shot scenarios. Furthermore, in the field of dense retrieval, significant research is dedicated to enhancing the text representation capabilities of pre-trained language models through the design of auxiliary pre-training tasks (Gao and Callan, 2021; Xiao et al., 2022; Gao and Callan, 2022; Wang et al., 2022a; Long et al., 2022b; Li et al., 2023).

The previous two lines of research can be generalized as learning a vector representation for a piece of text and distinguished by the type of downstream tasks. Recently, several studies have explored the construction of unified text representation models through large-scale contrastive learning and prompt-based learning (Neelakantan et al., 2022; Wang et al., 2022b; Su et al., 2023). Additionally, some research efforts have focused on constructing evaluation datasets to better assess the stability of text representation models across different tasks and domains. BEIR (Benchmarking IR) (Thakur et al., 2021) collects a substantial number of retrieval tasks from various domains to evaluate the robustness of dense retriever models in zero-shot scenarios. Meanwhile, MTEB (Massive Text Embedding Benchmark) (Muennighoff et al., 2023) benchmarks over 56 datasets spanning seven categories, providing a comprehensive evaluation of text embedding models.

This study aims to develop a general text embedding model through a multi-stage training approach. In the initial stage of unsupervised contrastive learning, we generate weak supervised correlation text pairs using publicly available data from various sources. Unlike previous study (Wang et al., 2022b), we exclusively utilized open-source data and did not employ any filtering or cleaning methods. Pre-training on a large-scale text pairs can effectively improve the domain generalization of text representation models and bridge the gap between the MLM training objective and the contrastive learning objective of representation models, making the language model more suitable for text representation tasks. In the supervised fine-tuning stage, the mixture of training data in our approach is more varied to further enhance the model's versatility. Moreover, our model does not incorporate task-specific prompts, which enhances reproducibility and ease of use.

3 Approach

The training process of our model consists of two stages: unsupervised pre-training and supervised fine-tuning. Both stages employ the learning objective of contrastive learning. Firstly, we will introduce the basic framework of the model. Subsequently, we will discuss the sources and construction methods of the training data in the two stages. Finally, we will present some special optimization strategies used to enhance the model's performance during the training process.

3.1 Model Architecture

The backbone of our embedding model is a deep Transformer encoder (Vaswani et al., 2017) which can be initialized with pre-trained language models such as BERT (Devlin et al., 2019). Our model follows the vanilla dual-encoder architecture with mean pooling on top of the contextualized token representations produced by the language model.

Formally, given a piece of text $x = (x_1, \ldots, x_n)$ consisting of n tokens, an embedding model E convert the text into a low-dimensional dense vector $\mathbf{x} = E(x) \in R^d$. To implement E, we first employ a language model to get the deep contextualized token representations

$$\mathbf{h} = \text{LM}(x) \in R^{n \times d}. \tag{1}$$

Then we apply a lightweight mean pooling across the first dimension to get the text representation,

$$\mathbf{x} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{h}_i \in R^d \tag{2}$$

The text representations are learned through the contrastive objective, distinguishing semantic relevant text pairs from irrelevant ones. Such training procedure requires positive and negative pairs, taking the format of (q, d^+, d^-). For a query q, a relevant document d^+, a set of irrelevant documents $\mathcal{D}_- = \{d_1^-, \ldots, d_n^-\}$, one popular contrastive objective is the InfoNCE loss (van den Oord et al., 2018),

$$L_{\text{cl}} = -\log \frac{e^{s(q,d^+)/\tau}}{e^{s(q,d^+)/\tau} + \sum_{i=1}^{n} e^{s(q,d_i^-)/\tau}}, \tag{3}$$

where $s(q, d)$ estimates the similarity between two pieces of text q and d via vector distance between $\mathbf{q} = E(q)$ and $\mathbf{d} = E(d)$.

To acquire text embeddings of superior quality that can be applied across a wide range of scenarios, we compile an extensive text pair dataset from multiple formats and domains. This dataset is then trained using an improved contrastive loss method in a multi-stage fashion.

3.2 Unsupervised Pre-training Data

Weakly supervised text relevance data is readily available in publicly accessible web sources, such as the inherent connection between queries and answers on QA forums. These data can be extensively collected without the need for manual annotation, thereby efficiently aiding in training text representation models. Inspired by previous work (Ni et al., 2022a,b; Neelakantan et al., 2022; Wang et al., 2022b), our model is initially pre-trained on naturally occurring text pairs extracted from diverse sources. To ensure the versatility of the embedding model, we explore a range of resources for text pair extraction, including web pages (*e.g.*, CommonCrawl, ClueWeb), scientific papers (*e.g.*, arXiv, SemanticScholar), community QA forums (*e.g.*, StackExchange), social media (*e.g.*, Reddit), knowledge bases (*e.g.*, Wikipedia, DBPedia), and code repositories (*e.g.*, StackOverflow, GitHub). Additionally, we harness the presence of hyperlinks in certain datasets to facilitate text pair extraction. Table 2 demonstrates some examples of text pair format from different sources. Further details regarding the data collection process can be found in Appendix A. In total, we utilized ∼800M text pairs text pairs for the unsupervised pre-training stage. Simple statistics and data distributions are illustrated in Table 1.

3.3 Supervised Fine-tuning Data

In the supervised fine-tuning stage, we use relatively lower-sized datasets with human annotation of the relevance between two pieces of text and optional hard negatives mined by an extra retriever to form text triples. To handle both symmetric tasks (*e.g.*, semantic textual similarity) and asymmetric tasks (*e.g.*, passage retrieval), we collect data from a large variety of tasks and domains, including web search (*e.g.*, MS MARCO), open-domain QA (*e.g.*, NQ), NLI (*e.g.*, SNLI), fact verification (*e.g.*, FEVER), paraphrases (*e.g.*, Quora). We totally used ∼3M pairs for fine-tuning, which is a combination of training data used by previous research (Gao et al., 2021; Gao and Callan, 2022; Asai et al., 2023; Su et al., 2023; Li et al., 2023). More details can be found in Appendix A.

Source	Datasets	Prop.	Size
Web Page	3	18.7%	147M
Academic Paper	5	5.7%	45M
Hyperlink	4	13.4%	106M
Social Media	2	41.5%	327M
Knowledge Base	2	4.8%	38M
Community QA	7	1.5%	12M
News	5	0.4%	3M
Code	2	2.5%	20M
Others	3	11.6%	91M
Total	33	100%	788M

Table 1: Statistics of pre-training data.

3.4 Training Details

Data Sampling In the initial stage of unsupervised pre-training, data sources often differ significantly in terms of the number of training instances. To address this imbalance, we employ a multinomial distribution to sample data batches from different data sources, taking into account their respective sizes. Suppose the whole pre-training dataset D consists of m different subsets $\{D_1, \ldots, D_m\}$ and denote the size of each subset as $n_i = |D_i|$, at each training iteration, the probability of sampling data from the i-th subset D_i can be represented by:

$$p_i = \frac{n_i^\alpha}{\sum_{j=1}^{m} n_j^\alpha}, \tag{4}$$

where we set $\alpha = 0.5$ in this work. Furthermore, to prevent the model from solely learning task-specific shortcuts for discrimination, we ensure that all training instances within a batch originate from the same task.

Improved Contrastive Loss When using the contrastive objective, people usually reuse in-batch documents as negative candidates to improve training efficiency (Karpukhin et al., 2020). This paper uses an improved contrastive learning objective which is bidirectional and enlarges the negative samples with both in-batched queries and documents. This can be viewd as a combination of loss variants proposed by Radford et al. (2021); Ren et al. (2021); Moiseev et al. (2023).

Task Type	Text Pair Format	Query	Doc
Web Page	(title, body)	Providence Real Estate | Providence Homes for Sale	Founded by Roger Williams in 1636, Providence is recognized as one of the country's oldest cities...
Academic Paper	(title, abstract)	Polymer Quantum Mechanics and its Continuum Limit	A rather non-standard quantum representation of the canonical commutation relations of quantum mechanics...
Hyperlink	(citation, reference)	After the championship in 1996, the PGA of America raised its stake to 50% and announced that ...	Pebble Beach Golf Links The largest margin of victory ever in a major championship, surpassing the 13-shot ...
Social Media	(post, comment)	Pretty sure any team with Lebron James will be a playoff contender. Considering UNC would be in the East...	I was being sarcastic and making fun of the East, but honestly I was really in deep thought about this ...
Knowledge Base	(entity, description)	Animation	Animation is the process of creating the illusion of motion and shape change by means of the rapid display of ...
Community QA	(question, answer)	How the human species evolved?	A tough question as it overlaps science and theology. Since you asked "how the human species evolved?" I'll assume ...
News	(summary, content)	Nepalese Opposition Welcomes Return of Parliament	Nepal's opposition alliance formally calls off weeks of pro-democracy protests after King Gyenandra reinstates ...
Code	(text, code)	SetMaxRecords sets the MaxRecords field's value.	func (s *DescribeSnapshotCopyGrantsInput) SetMaxRecords (v int64) *DescribeSnapshotCopyGrantsInput { s.MaxRecords

Table 2: Examples of mined (query, document) pairs in the pre-training data.

Consider a batch of positive text pair samples

$$B = \{(q_1, d_1), (q_2, d_2), ..., (q_n, d_n)\},$$

we use an improved contrastive loss which takes the form

$$L_{\text{icl}} = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{e^{s(q_i,d_i)/\tau}}{Z} \tag{5}$$

with the partition function being

$$\begin{aligned} Z = &\sum_{j} e^{s(q_i,d_j)/\tau} + \sum_{j\neq i} e^{s(q_i,q_j)/\tau} \\ &+ \sum_{j} e^{s(q_j,d_i)/\tau} + \sum_{j\neq i} e^{s(d_j,d_i)/\tau} \end{aligned} \tag{6}$$

in which the first two terms are used for query to document contrast, where as the last two terms are used for the inverse. In this work, we use the cosine similarity as the distance metric

$$s(q, d) = \frac{\mathbf{q} \cdot \mathbf{d}}{||\mathbf{q}||_2 \cdot ||\mathbf{d}||_2}. \tag{7}$$

The temperature τ is fixed to 0.01 in this work.

Training and Evaluation The training of our embedding model consists of two stages. In the first stage of contrastive pre-training with only in-batch negatives, using a large batch size is crucial to better model performance by reducing the gap between training and inference with more negatives included and providing a better approximation to the underlying learning objective. To facilitate this, we limit the maximum sequence length to 128 during pre-training and distribute the use of negatives across all GPUs. Popular techniques such as automatic mixed precision training (Micikevicius et al., 2018) with fp16, deepspeed ZeRO (Rajbhandari et al., 2020) stage 1 and gradient checkpointing (Chen et al., 2016) are also jointly used to reduce memory cost and scale up batch size to over ten thousands. We run the pre-training for $50,000$ steps, which roughly corresponds to one epoch on the whole pre-training data. We only tuned the learning rate to ensure the convergence of larger models. we employ the AdamW optimizer with linear learning rate decay and a warm-up period during the initial 5% of training steps. We conducted experiments on three distinct model scales: small, base, and large. These models were initialized using the small-sized MiniLM (Wang et al., 2020) model and the base and large models of the BERT (Devlin et al., 2019) model. Further details can be found in Table 3.

In the second stage of contrastive fine-tuning with supervised data and hard negatives, a large batch size is unnecessary since hard negatives can already provide a reliable gradient estimation of the learning objective (Xiong et al., 2021; Li et al., 2023). Therefore, a global batch size of 128 and a train group size of 16 are utilized, with one positive example and the remaining being either hard negatives or random negatives. Instead we increase the max sequence length to 512 to better handle texts with longer lengths. The learning rate is decreased by a factor of ten during fine-tuning. The model is fine-tuned on the collected dataset for a single epoch. In-batch texts are also incorporated as negative candidates using the enhanced contrastive loss described in Equation 5.

After training, we directly take the last checkpoint for evaluation. We run model training on up to 8 NVIDIA A100 GPUs with 80GB memory and model evaluation on up to 8 NVIDIA Tesla V100 GPUs with 32GB memory. Models are trained with mixed precision using fp16 and evaluated with half precision fp16 as well.

Model	Params	LR	GPUs	BS	Base LM
GTE_{small}	30M	3×10^{-4}	2	16384	`microsoft/MiniLM-L12-H384-uncased`
GTE_{base}	110M	2×10^{-4}	4	16384	`bert-base-uncased`
GTE_{large}	330M	5×10^{-5}	8	16384	`bert-large-uncased`

Table 3: Pre-training configurations of models of different sizes.

4 Experiments

In this section, we provide an extensive evaluation of our embedding model, comparing to state-of-the-art models for each task. Note that an apple-to-apple comparison is hardly possible since different models used different in-house data for pre-training and the base language models vary a lot. We mainly use the number of model parameters as a criterion for performance comparison since it is closely related to the inference speed.

4.1 Zero-shot Text Classification

Model	Params	Prompting	Accuracy
$E5_{base}$	110M	✓	81.3
$E5_{large}$	330M	✓	85.3
`cpt-text`	6B		88.1
`cpt-text`	6B	✓	89.1
GTE_{base}	110M		85.1
GTE_{base}	110M	✓	87.2

Table 4: Zero shot text classification performance on SST-2. All compared models are the fine-tuned ones.

One method to assess the quality of learned representation is through zero-shot classification. (Radford et al., 2021; Neelakantan et al., 2022; Wang et al., 2022b). We recast text classification into an embedding-based similarity matching problem. In this setting, inputs texts are converted into embeddings directly and labels are verbalized to corresponding text to get label embeddings. Distances between input embeddings and label embeddings are measured by their inner product and label with the most close embedding distance to the input text is regarded as the classification result. An example is SST-2 binary sentiment classification task. We consider two types of label verbalizers for evaluation. The vanilla version uses the sentiment word 'positive' or 'negative' to denote the corresponding labels. Prompted version uses fuzzy prompt template, such as 'this is an example of positive/negative movie review'.

Zero-shot text classification accuracy on SST-2 is shown in Table 4. In the vanilla setting, our 110M model already matches the performance of prompted $E5_{large}$ with 330M parameters. Using prompting strategy further improves results significantly and closes the gap with large models. Even without explicit prompt or instruction during training, our model can somewhat understand the label context better when formatted as a natural language text.

4.2 Unsupervised Text Retrieval

Text retrieval requires retrieving most relevant documents from a large-scale candidate sets. We use BEIR (Thakur et al., 2021) as our evaluation benchmark for zero-shot unsupervised text retrieval. BEIR is a heterogeneous information retrieval benchmark which contains retrieval tasks of different formats and from different domains. We use the open available 15 datasets for evaluation.

We compare our unsupervised pre-trained checkpoint to recent unsupervised dense retrievers such as Contriever (Izacard et al., 2022a) and E5 (Wang et al., 2022b). According to Table 5, we find that our base size model significantly outperforms the models with comparable size, like SimCSE, Contriever and E5. Our base model is comparable to $E5_{large}$ without using human supervision.

4.3 Massive Text Embedding Benchmark

Massive Text Embedding Benchmark (MTEB) is a comprehensive semi-supervised benchmark that incorporates a limited amount of supervision data for evaluation. In this paper, we evaluate the English subsets which encompasses 56 English datasets across seven distinct tasks, including text classification (Class.), text clustering (Clust.), pairwise classification (Pair.), text reranking (Rerank.), text retreival (Retr.), semantic textual similarity (STS) and summarization (Summ.). The evaluation metrics employed in MTEB are accuracy, v-measure, average precision, MAP, nDCG@10, and Spearman coefficients, respectively. For further details

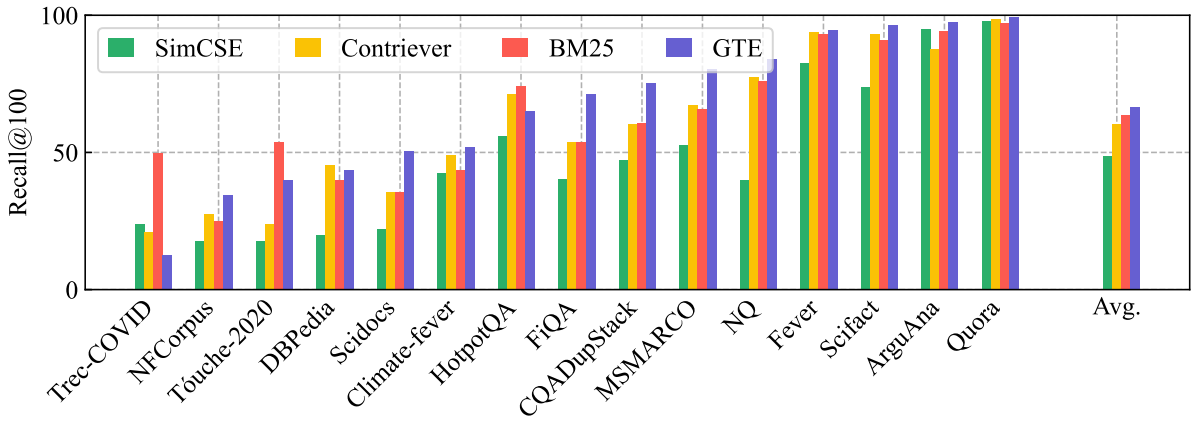


Figure 2: Recall@100 of unsupervised text retrieval methods on BEIR benchmark (Thakur et al., 2021). We compare our model GTE_{base} (based on $\text{BERT}_{\text{base}}$) without using any annotated data to SimCSE (Gao et al., 2021) (based on $\text{RoBERTa}_{\text{large}}$), Contriever (Izacard et al., 2022a) (based on $\text{BERT}_{\text{base}}$) and BM25. Baseline results are borrowed from the Contriever paper (Izacard et al., 2022a) with dot product being the similarity function.

Dataset	BM25	SimCSE	Contriever	CPT-S	E5_{small}	E5_{base}	E5_{large}	$\text{GTE}_{\text{small}}$	GTE_{base}	$\text{GTE}_{\text{large}}$
MS MARCO	22.8	9.4	20.6	19.9	25.4	26.0	26.2	31.3	31.8	31.7
Trec-Covid	65.6	26.2	27.4	52.9	52.0	61.0	61.8	61.8	64.0	64.8
NFCorpus	32.5	9.9	31.7	32.0	29.3	35.8	33.7	34.9	36.2	38.1
NQ	32.9	11.7	25.4	-	37.3	39.0	41.7	32.0	35.3	34.5
HotpotQA	60.3	19.8	48.1	51.5	46.0	52.4	52.2	49.3	50.8	49.2
FiQA	23.6	9.8	24.5	34.1	38.3	40.0	43.2	37.0	36.9	40.6
ArguAna	31.5	38.3	37.9	38.7	42.5	42.2	44.4	41.6	41.0	41.3
Touche-2020	36.7	8.9	19.3	21.0	19.9	16.9	19.8	17.7	18.2	18.5
CQADupStack	29.9	13.2	28.4	-	35.0	35.4	38.9	38.1	39.9	39.8
Quora	78.9	78.0	83.5	68.1	85.8	85.7	86.1	86.1	85.0	84.8
DBPedia	31.3	15.0	29.2	27.2	34.5	35.4	37.1	33.5	33.2	33.6
Scidocs	15.8	5.5	14.9	-	19.9	21.1	21.8	21.5	22.5	22.7
Fever	75.3	21.1	68.2	57.1	62.5	63.4	68.6	71.3	72.7	70.5
Climate-Fever	21.3	11.8	15.5	15.8	14.5	15.4	15.7	21.4	21.0	25.4
Scifact	66.5	25.7	64.9	65.4	68.5	73.7	72.3	72.7	74.1	74.1
Average	41.7	20.3	36.0	-	40.8	42.9	44.2	43.4	44.2	44.6

Table 5: nDCG@10 of different unsupervised methods on the BEIR benchmark (Thakur et al., 2021). SimCSE is based on $\text{BERT}_{\text{base}}$ backbone. CPT-S (Neelakantan et al., 2022) is of similar size to $\text{BERT}_{\text{large}}$. Baseline results are borrowed from E5 paper (Wang et al., 2022b). Note that Contriever uses dot product as the similarity metric while other models uses cosine similarity.

on the tasks covered in the MTEB benchmark, please refer to the Appendix B.

Two settings are considered for comparison: the unsupervised setting and the supervised setting. In the unsupervised setting, models are trained using unlabeled data, while supervised models are fine-tuned using high-quality datasets with human labels. The results of strong baseline models are presented in Table 6.

In the unsupervised setting, our model outperforms the previous best model, E5, by a significant margin across all considered tasks, without the use of task-specific prompts. This improvement can be attributed to the inclusion of more training data formats and various sources of self-supervision signals. Furthermore, it is worth noting that our unsupervised pre-trained model narrows the gap even further with larger supervised baselines, such as GTR and Sentence-T5. In the supervised setting, our model surpasses OpenAI results

# of datasets →	Params	Class. 12	Clust. 11	Pair. 3	Rerank 4	Retr. 15	STS 10	Summ. 1	Avg 56
Unsupervised models									
Glove	120M	57.3	27.7	70.9	43.3	21.6	61.9	28.9	42.0
BERT	110M	61.7	30.1	56.3	43.4	10.6	54.4	29.8	38.3
SimCSE	110M	62.5	29.0	70.3	46.5	20.3	74.3	31.2	45.5
$E5_{small}$	30M	67.0	41.7	78.2	53.1	40.8	68.8	25.2	54.2
$E5_{base}$	110M	67.9	43.4	79.2	53.5	42.9	69.5	24.3	55.5
$E5_{large}$	330M	69.0	44.3	80.3	54.4	44.2	69.9	24.8	56.4
GTE_{small}	30M	71.0	44.9	82.4	57.5	43.4	77.2	30.4	58.5
GTE_{base}	110M	71.5	46.0	83.3	58.4	44.2	76.5	29.5	59.0
GTE_{large}	330M	71.8	46.4	83.3	58.8	44.6	76.3	30.1	59.3
Supervised models									
SimCSE	110M	67.3	33.4	73.7	47.5	21.8	79.1	23.3	48.7
Contriever	110M	66.7	41.1	82.5	53.1	41.9	76.5	30.4	56.0
GTR_{large}	330M	67.1	41.6	85.3	55.4	47.4	78.2	29.5	58.3
Sentence-$T5_{large}$	330M	72.3	41.7	85.0	54.0	36.7	81.8	29.6	57.1
$E5_{small}$	30M	71.7	39.5	85.1	54.5	46.0	80.9	31.4	58.9
$E5_{base}$	110M	72.6	42.1	85.1	55.7	48.7	81.0	31.0	60.4
$E5_{large}$	330M	73.1	43.3	85.9	56.5	50.0	82.1	31.0	61.4
$InstructOR_{base}$	110M	72.6	42.1	85.1	55.7	48.8	81.0	31.0	60.4
$InstructOR_{large}$	330M	73.9	45.3	85.9	57.5	47.6	83.2	31.8	61.6
$OpenAI_{ada-001}$	n.a.	70.4	37.5	76.9	49.0	18.4	78.6	26.9	49.5
$OpenAI_{ada-002}$	n.a.	70.9	45.9	84.9	56.3	49.3	81.0	30.8	61.0
GTE_{small}	30M	72.3	44.9	83.5	57.7	49.5	82.1	30.4	61.4
GTE_{base}	110M	73.0	46.1	84.3	58.6	51.2	82.3	30.7	62.4
GTE_{large}	330M	73.3	46.8	85.0	59.1	52.2	83.4	31.7	63.1
Larger models									
$InstructOR_{xl}$	1.5B	73.1	44.7	86.6	57.3	49.3	83.1	32.3	61.8
GTR_{xxl}	4.5B	67.4	42.4	86.1	56.7	48.5	78.4	30.6	59.0
Sentence-$T5_{xxl}$	4.5B	73.4	43.7	85.1	56.4	42.2	82.6	30.1	59.5

Table 6: Results on the MTEB (Muennighoff et al., 2023) (56 datasets in English subset). Compared models include SimCSE (Gao et al., 2021), Sentence-T5 (Ni et al., 2022a), GTR (Ni et al., 2022b), Contriever (Izacard et al., 2022a), OpenAI text embedding API (Neelakantan et al., 2022), E5 (Wang et al., 2022b) and InstructOR (Su et al., 2023). Exact parameter amount of OpenAI ada model is not available, but is suspected to be ∼300M, comparable to the BERT large size model.

by a large margin despite using a modest model size. GTE_{small} is comparable to $E5_{large}$ while being 10× smaller. GTE_{large} establishes new state-of-the-art performance on the MTEB benchmark, outperforming the multi-task instruction-finetuned embedding model, $InstructOR_{large}$, by 1.5 points on average.

4.4 Code Search

Programming languages can be regarded as a distinct form of text. To assess the effectiveness of our approach in code search, we conduct a comparative analysis with other code-based language models, such as CodeBERT (Guo et al., 2021) and GraphCodeBERT (Guo et al., 2021). We also compare our approach with a more recent code language model called UniXcoder (Guo et al., 2022), which aims to integrate various pre-training tasks into a unified model. CodeRetriever (Li et al., 2022) is initialized from GraphCodeBERT and pre-trained on large-scale multi-modal code-text pairs mined and cleaned by heuristics. It is important to note that while the baseline models are individually trained and evaluated for each programming language, our model is directly evaluated across all the languages.

In line with recent work (Guo et al., 2021, 2022;

Model	Params	Ruby	JS	Go	Python	Java	PHP	Avg.
CodeBERT	110M×6	67.9	62.0	88.2	67.2	67.6	62.8	69.3
GraphCodeBERT	110M×6	70.3	64.4	89.7	69.2	69.1	64.9	71.3
UniXcoder	110M×6	74.0	68.4	91.5	72.0	72.6	67.6	74.4
CodeRetriever	110M×6	77.1	71.9	92.4	75.8	76.5	70.8	77.4
GTE_{base}	110M	76.1	73.6	88.1	95.9	80.1	85.3	83.2

Table 7: Results on CodeSearchNet. Comparison on code search across 6 programming languages (Husain et al., 2019) with CodeBERT (Feng et al., 2020), GraphCodeBERT (Guo et al., 2021), UniXcoder (Guo et al., 2022) and CodeRetriever (Li et al., 2022). This setting requires finding the corresponding code candidates from all candidates from dev and test set.

Li et al., 2022), we mainly evaluate on the challenging settings where the code corpus include all codes from dev and test set instead of 1k randomly sampled code.[2] The results are presented in Table 7. Surprisingly, our model surpasses models that are pre-trained on code and then fine-tuned for each programming language separately. This finding demonstrates that, by scaling the amount of data and computational resources, the language model can acquire high-quality code representations directly from sequences of code tokens, without the need for incorporating human knowledge about the structural information of code (Guo et al., 2021). We observe a significant improvement in Python, likely due to its resemblance to natural language. Our model, pre-trained on an extensive text pairs spanning various domains, demonstrates effective cross-task knowledge transfer from text retrieval to code retrieval.

5 Analysis

In this section, we analyze the crucial factors influencing model performance and present a series of ablation experiments. Unless otherwise stated, the experiments are performed using a BERT-base scale model with 110M parameters. The training steps and epochs remain consistent across all ablation experiments.

5.1 Impact of Scaling

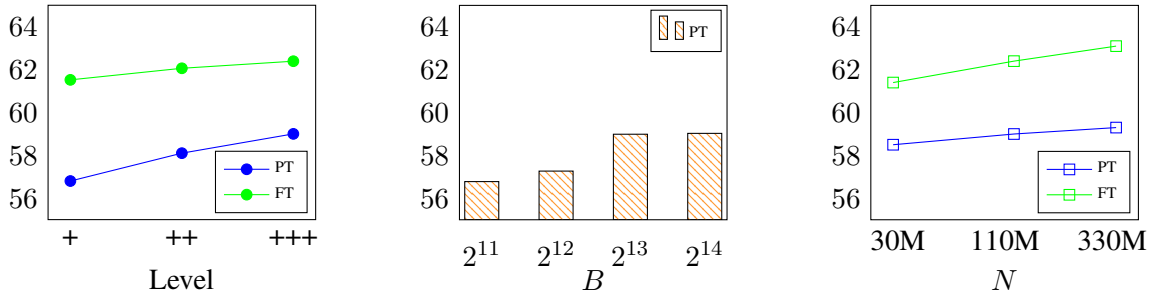
We investigate the impact of scaling the number of data sources, batch size, and model parameters on the quality of learned text embeddings. The evaluation is conducted on the MTEB benchmark.

Number of Training Datasets First, we conducted an ablation study on the number of training datasets used in pre-training. Model training was carried out by randomly sampling a subset from all available datasets. In the pre-training stage, the first group consisted of only the five largest datasets, ranked by size. The second group included an additional 10 randomly sampled datasets, resulting in a mixture of 15 datasets. The third group utilized all 33 datasets in the pre-training process. For fine-tuning, we initially started with the three datasets used in E5 (Wang et al., 2022b) fine-tuning and gradually incorporated datasets from MEDI (Su et al., 2023) and BERRI (Asai et al., 2023) to investigate the potential benefits. The results presented in Figure 3a demonstrate that the inclusion of more diverse data sources consistently enhances model performance during both the pre-training and fine-tuning stages.

Pre-training Batch Size We gradually increase the batch size by a factor of 2 while keeping the training steps fixed to study the influence of batch size used in embedding model pre-training. According to Figure 3b, model performance saturates at around a batch size of ten thousands. No performance gain is observed when further scaling up batch size.

Number of Model Parameters We investigate the scaling behavior by training language models of various sizes, including 30M, 110M, and 330M, which correspond to the small, base, and large scales of the BERT model. Figure 3c illustrates the performance of the pre-trained and fine-tuned models. It can be observed that as the model size grows exponentially, the model performance also improves linearly.

[2]Evaluation on the original settings can be found in Appendix C.



(a) Number of training datasets. (b) Batch size. (c) Number of model parameters.

Figure 3: Scaling analysis of different factors during contrastive pre-training and fine-tuning. Model performance is mesured by average performance on MTEB.

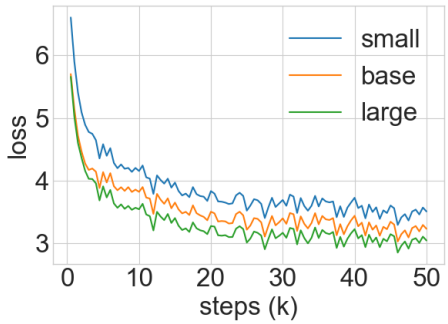


Figure 4: Loss during contrastive pre-training for models of different size.

5.2 Training Behavior

We plot the training loss of different sized models during contrastive pre-training in Figure 4. Larger models have better ablity at learning to distinguish positive pairs from negative ones. The training loss experiences minor fluctuations consistently across all model scales, which suggests variations in the quality and difficulty of data per batch.[3]

We also evaluate model performance at different training steps. It's shown that model performance saturates at 20k steps roughly corresponds to training convergence.

Steps	10k	20k	30k	40k	50k
MTEB	56.4	59.0	57.8	57.7	59.0

Table 8: Model performance at different training steps during unsupervised contrastive pre-training.

[3]We use a fixed random seed for data sampling during model training, ensuring that each model encounters the data batches in the same order.

5.3 Influence of Different Training Stages

To examine the efficacy of multi-stage contrastive learning, we conducted an analysis on the training strategies. We compared three settings: a) solely pre-training on unsupervised text pairs extracted from diverse sources; b) solely fine-tuning on supervised datasets; c) contrastive pre-training followed by fine-tuning. All models were initialized from the original BERT base model.

Setting	PT	FT	Full
MTEB	59.0	57.8	62.4

Table 9: Model performance at different training stages. PT denotes run only unsupervised pre-training. FT only use supervised data for model trainining. Full apply two stages in a sequential manner.

It can be observed from Table 9 that relying solely on supervised data for fine-tuning is insufficient to achieve a high-quality text embedding model, likely due to its limited size. Conversely, unsupervised pre-training using web-scale text pairs yields superior text embeddings compared to solely relying on labeled data for fine-tuning. Nevertheless, the incorporation of supervised data in a multi-stage fashion with unsupervised pre-training can still contribute to the refinement of the acquired text embeddings.

5.4 Training Data Mixture

We study the influence of mixing ratio used in sampling distribution on pre-training data to model performance.

The performance on two task categories, retrieval and STS, as well as the average performance on MTEB is reported in Table 10. We observe that neither uniformly sampling from each pre-

α	Retrieval	STS	MTEB
0	36.7	73.2	55.4
0.3	44.6	75.9	58.9
0.5	44.2	76.5	59.0
1	42.0	75.5	58.3

Table 10: Influence of ratio α used in pre-training data sampling.

training task ($\alpha = 0$) nor directly combining all data sources ($\alpha = 1$) is the best choice. Setting α to 0.5 can improve results on all tasks.

5.5 Ablation of the Contrastive Objective

This work uses an improved contrastive objective which can efficiently enlarges the negative pool under fixed batch size. We compare it against the vanilla contrastive loss with only in-batch negatives in both pre-training and fine-tuning stages.

Setting	PT	FT
Vanilla	57.3	61.8
Improved	57.8	62.4

Table 11: Comparison of the vanilla contrastive loss with in-batch negatives, and the improved contrastive loss with enlarged negative pool. For ablation we run the pre-training (PT) for 30k steps to reduce computational cost. We report average score on MTEB.

According to Table 11, using improved contrastive loss consistently improves model performance in both pre-training and fine-tuning stages.

6 Discussion

Despite the strong performance on English tasks, our current model can only handle text with a length of less than 512, as it is initialized from BERT and lacks multilingual capabilities. Consequently, longer texts must be truncated or split for encoding. However, with more data engineering and compute resources, the described training approach could easily be extended to a multilingual version and accommodate longer contexts.

Another issue is the problem of data contamination resulting from large-scale pre-training on Internet data. Currently, we only conduct deduplication based on exact matching of text pairs, which is an overly strict filter. This issue has also been highlighted by Brown et al. (2020) during the training of large-scale generative language models. We suspect that this is a common problem that other models also suffer from, but quantifying it without details about the training data sources is even more challenging (Neelakantan et al., 2022).

Furthermore, the models trained in this study are based on a non-causal architecture with bidirectional context attention. It would be intriguing to explore similar pre-training methods for causal or prefix language models, as these models could optimize generation and retrieval jointly and unify them within a single model.

7 Conclusion

This paper presents a multi-stage contrastive learning approach to develop text embedding model that can be applied to various tasks. Our model benefits from a diverse training data mixture, enabling it to achieve good generalization performance for single vector embedding. Through extensive evaluation on multiple benchmarks, we demonstrate the effectiveness and versatility of our text embedding model. Our future work will focus on scaling the model to support longer context, extending it to support multilingual and multi-modal applications, as well as exploring the benefits of prompts and instructions.

References

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2023. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3650–3675, Toronto, Canada. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost.

Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. 2021. Evaluation of bert and albert sentence embedding performance on downstream nlp tasks. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5482–5487.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Conference on Empirical Methods in Natural Language Processing*.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2843–2853, Dublin, Ireland. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. 2022. UniXcoder: Unified cross-modal pre-training for code representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7212–7225, Dublin, Ireland. Association for Computational Linguistics.

Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie LIU, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, and Ming Zhou. 2021. Graphcode{bert}: Pre-training code representations with data flow. In *International Conference on Learning Representations*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

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 *International Conference on Learning Representations*.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *CoRR*, abs/1909.09436.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022a. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022b. Few-shot Learning with Retrieval Augmented Language Models.

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.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online. Association for Computational Linguistics.

Xiaonan Li, Yeyun Gong, Yelong Shen, Xipeng Qiu, Hang Zhang, Bolun Yao, Weizhen Qi, Daxin Jiang, Weizhu Chen, and Nan Duan. 2022. CodeRetriever: A large scale contrastive pre-training method for code search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2898–2910, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zehan Li, Yanzhao Zhang, Dingkun Long, and Pengjun Xie. 2023. Challenging decoder helps in masked auto-encoder pre-training for dense passage retrieval. *CoRR*, abs/2305.13197.

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.

Dingkun Long, Qiong Gao, Kuan Zou, Guangwei Xu, Pengjun Xie, Ruijie Guo, Jianfeng Xu, Guanjun Jiang, Luxi Xing, and Ping Yang. 2022a. Multi-cpr: A multi domain chinese dataset for passage retrieval. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Dingkun Long, Yanzhao Zhang, Guangwei Xu, and Pengjun Xie. 2022b. Retrieval oriented masking pre-training language model for dense passage retrieval. *ArXiv*, abs/2210.15133.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations*.

Fedor Moiseev, Gustavo Hernandez Abrego, Peter Dornbach, Imed Zitouni, Enrique Alfonseca, and Zhe Dong. 2023. SamToNe: Improving contrastive loss for dual encoder retrieval models with same tower negatives. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12028–12037, Toronto, Canada. Association for Computational Linguistics.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training. *CoRR*, abs/2201.10005.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022a. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. 2022b. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Barlas Oguz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Scott Yih, Sonal Gupta, and Yashar Mehdad. 2022. Domain-matched pre-training tasks for dense retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1524–1534, Seattle, United States. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Thilina C. Rajapakse. 2023. Dense passage retrieval: Architectures and augmentation methods. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *ArXiv*, abs/2302.00083.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2173–2183, Online. Association for Computational Linguistics.

Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *ArXiv*, abs/2301.12652.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada. Association for Computational Linguistics.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022a. Simlm: Pre-training with representation bottleneck for dense passage retrieval. In *Annual Meeting of the Association for Computational Linguistics*.

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

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606, Online. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. Retromae: Pre-training retrieval-oriented language models via masked auto-encoder. In *Conference on Empirical Methods in Natural Language Processing*.

Yiqing Xie, Xiao Liu, and Chenyan Xiong. 2023. Unsupervised dense retrieval training with web anchors. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2476–2480, New York, NY, USA. Association for Computing Machinery.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Jiawei Zhou, Xiaoguang Li, Lifeng Shang, Lan Luo, Ke Zhan, Enrui Hu, Xinyu Zhang, Hao Jiang, Zhao Cao, Fan Yu, Xin Jiang, Qun Liu, and Lei Chen. 2022. Hyperlink-induced pre-training for passage retrieval in open-domain question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7135–7146, Dublin, Ireland. Association for Computational Linguistics.

A More Details about Training Data

A.1 Pre-training Data

Web Page Within a web page, we use title as query and the body text as document. Some resources include Common Crawl, Clue Webs, MS MARCO documents. The task can be formatted as given a short title, find the most relevant body texts from a set of randomly sampled texts.

Academic Paper The scientific articles usually have a higher quality due to its formal nature. For each paper, we use the title as query and its abstract as document for constructing text pairs. The articles are mined from different websites (such as arXiv, bioRxiv, medRxiv, PubMed and Semantic Scholar) to cover a wide range of topics.

Hyperlink Another important information present on the internet is the hyperlink with text in it, also know as web anchors. The hyperlink can provide necessary references for current arguments. We use the citation argument and the text from reference as relevant text pairs for contrast. This type of task is more challenging as it usually involves multi-hop reasoning. We used three resources to incorporate the link information: ClueWeb, Wikipedia and Semantic Scholar paper citations.

Community QA We also used many data from community QA websites. The UI design of such websites usually follows a structured format, where the user can write their questions in the format of a summaritive title and a descriptive body. These two fields are usually semantically consistent. In addition, we also consider the question answer pairs from this type of websites. The data sources we used include StackExchange, Yahoo Answers, WikiHow and Amazon QA. Simple heuristics such as text lengths and voting numbers are used to filter out low-quality data.

Social Media The social media websites such as Twitter and Reddit usually involves people publishing posts about one event, and many internauts leave their comments. The post is also structured with title and body in it, which we consider as positive pairs. Similar to Community QA, post comment are also regared as positive pairs for data mining. We mine data from Reddit.

News News are structured as title body pairs. Some news has highlighted sentences in it. We use these information to construct (query,doc) pairs. We used data from CCNews, MicrosoftNews, NPR, CNNDaily.

Knowledge Base Knowledge base usually stores textual descriptions knowledge about an entity or event. The (entity, description) pairs are mined. We use WikiPedia and DBPedia for text pair mining in this work.

Code Code can be viewed as another form of text. The naturally paired text-code can be repurposed as positive pairs. We use GitHub and StackOverflow as two data sources. We reuse training set from CodeSearchNet which is mined from GitHub.

Others In addition, we also use data from various websites such as Amazon reviews about the goods, debate websites about one argument, googaq q,a pairs by prompting google search box with search log queries.

A.2 Fine-tuning Data

Web Search We used MS MARCO passage retrieval benchmarks. Hard negatives are mined by sampling from high-ranked documents retrieval system, excluding positive ones.

Open QA We consider Natural Questions, Trivia QA, Web Questions, HotpotQA, etc. In the open domain QA datasets, a question and its supporting evidence passages are provided as positive pairs. Top ranked passage by retrieval system which do

not include answer to the question is regared as hard negatives.

Natural Language Inference Prior work (Conneau et al., 2017) has shown that high-quality sentence embeddings can be learned from a supervised natural language inference task. We use entailment as positive pairs and contradiction as negative pairs to construct training triples. The combination of MNLI and SNLI is used in this work.

Fact Verification One argument and its supporting source (a Wikipedia document) is positive pairs. We use training set from FEVER as data source for this task.

Paraphrase Two sentences with similar meanings are labeled as positive pairs. This type of data includes Quora and StackExchangeDupquestion.

Others In addition to previous datasets, we also used miscellaneous datasets from different NLP tasks and domains released in MEDI (Su et al., 2023) and BERRI (Asai et al., 2023). By doing so, a sub-sampled version of pre-training data is also included in fine-tuning to avoid catastrophe forgetting.

A.3 Data Sources

The pre-training data comes mostly from language corpus released by previous work. We use CommomCrawl preprocessed by CCNet at 2019 snapshot due to large computaional cost of processing (Wenzek et al., 2020). Since Reddit data is no longer free available, we use two pre-processed version by sentence-transformers [4] and Oguz et al. (2022) for pair mining. Text pairs mined from hyperlinks come from Zhou et al. (2022) and Xie et al. (2023). We also include citation pairs from the S2ORC dataset (Lo et al., 2020). We reuse DBPedia, debating arguments and PubMed corpus from BEIR (Thakur et al., 2021). Wikipedia data is taken from Izacard et al. (2022b). Microsoft News data comes from Wu et al. (2020). Arxiv data is downloaded from Kaggle, medRxiv and bioRxiv are mined via requesting public API from year 2013 to 2022. The StackExchange and StackOverflow data comes from the pre-processed version maintained by sentence-transformers team.[5] The remaining data comes from embedding-training-data.[6] The training data is keep as it was without any specific filtering, except that we use text pair exact-match for training data de-duplication for some datasets.

The fine-tuning data is basically a combination of previous research. For the MS MARCO dataset, we use mined hard negative by the second stage retriever from Li et al. (2023). For NQ dataset, we reuse the training data released by coCondenser (Gao and Callan, 2022). We use NLI data released by SimCSE (Gao et al., 2021). Other data comes from MEDI and BERRI (Su et al., 2023; Asai et al., 2023), but we discard the instructions written for each task and only use training triples. Some randomly sampled examples can be found in Table 12.

B Massive Text Embedding Benchmark

Classification This task is evaluated in the linear probing setting. The embedding model is kept frozen and used to extract text embeddings for each example from train and test set. The train set embeddings are used as input features to train a logistic regression classifier with 100 maximum iterations. The accuracy on test set is reported as the main evaluation metric. In this setting, different classification tasks only need to train an extra classification head with a few labeled training data.

Clustering A high-quality embedding model should embed semantically similar texts close in the embedding space. This property is evaluated by running a k-means algorithm on the embeddings produced for each sentence of the test set. A mini-batch k-means model is used with batch size 32 and k being the number of labels. Texts are partitioned into k clusters. The clustering performance is measured by the v-measure (Rosenberg and Hirschberg, 2007) which is invariant to the permutation of clustering labels.

Reranking Given a query and a list of relevant and irrelevant reference texts, reranking needs to rank the list of reference texts based on their similarity to the query. The embedding model is invoked to obtain embeddings for each query and reference text and cosine similarity is used as the ranking score. This inference setting is quite similar to text retrieval with the reference set being

[4] https://huggingface.co/datasets/sentence-transformers/reddit-title-body

[5] https://huggingface.co/flax-sentence-embeddings

[6] https://huggingface.co/datasets/sentence-transformers/embedding-training-data

Task Type	Text Triple Format	query	doc	hard neg
Web Search	(query, passage, negative)	finger cellulitis symptoms	The following are the most common symptoms of cellulitis. However…	Cellulitis usually begins as a small area of pain and …
Open QA	(question, passage, negative)	big little lies season 2 how many episodes	Big Little Lies (TV series). series garnered several accolades…	Little People, Big World. final minutes of the season two…
Natural Language Inference	(sentence, entailment, contradiction)	(Read for Slate 's take on Jackson's findings.)	Slate had an opinion on Jackson's findings.	Slate did not hold any opinion on Jackson's findings.
Fact Verification	(argument, evidence, others)	Roman Atwood is a content creator.	Roman Bernard Atwood (born May 28, 1983) is an American YouTube personality…	6th Streamy Awards Casey Neistat and Jesse Wellens, PrankvsPrank …
Paraphrase	(sentence, paraphrase, others)	Lexapro taken with crestor any reaction?	Can dayquil be taken with Lexapro?	Can stopping lexapro cause a longer period?

Table 12: Examples of (query, positive, negative) text triples in fine-tuning data.

smaller and harder to distinguish. In line with previous work, the main evaluation metric is MAP (mean average precision).

Retrieval We omit the text retrieval evaluation since it's similar to that introduced in previous section.

Pair Classification This task needs to assign a label for a pair of texts. Popular tasks include duplicate or paraphrase identification, where the label is binary. The similarity score is the cosine similarity between the embeddings of two texts. The average precision score is reported as the main evaluation metric using the best binary threshold.

Semantic Textual Similarity To determine the similarity between a given pair of sentences, continuous scores are assigned, with higher values indicating greater similarity. The embedding model is employed to embed the sentences, and their similarity is computed using cosine similarity. The estimated similarity scores is compared against human labeled scores ranging from 1 to 5. We report Spearman's correlation, which measures the rankings instead of the actual scores and better suits the need of evaluating sentence embeddings.

Summarization This is a text generation evaluation task which aims to automatically evaluate the quality of generated text. For the summarization task, the quality of each generated summary is computed by measuring the cosine similarity between its embedding and the embedding of the ground truth references. In the case of multiple gold references, the closest one with highest similarity score is used for quality estimation. Similar to STS task, we use the Spearman correlation between the ranking produced by the text embedding model and the human assessments for evaluation.

C Original CodeSearchNet Results

We list the results of the original setting on CodeSearchNet in Table 13, where the retrieval corpus contains 1k randomly sampled code snippets. Compared to previous open-source code language models with similar architecture and size (CodeBERT (Feng et al., 2020) and GraphCodeBERT (Guo et al., 2021)), our model is superior in most programming languages. There is still a performance gap to the code embedding model trained by Neelakantan et al. (2022), which used Codex (Chen et al., 2021) as backbone and trained on a large-scale (code, text) pairs extracted from open-source code. It is worthwhile to explore how to further close this gap.

Model	Params	Ruby	JS	Go	Python	Java	PHP	Avg.
CodeBERT	110M × 6	69.3	70.6	84.0	86.8	74.8	70.6	76.0
GraphCodeBERT	110M × 6	84.1	73.2	87.9	75.7	71.1	72.5	77.4
`cpt-code` S	300M	86.3	86.0	97.7	99.8	94.0	96.7	93.4
`cpt-code` M	1.2B	85.5	86.5	97.5	99.9	94.4	97.2	93.5
GTE_{base}	110M	79.6	79.4	84.2	98.8	86.8	86.8	85.9

Table 13: Results on CodeSearchNet (Husain et al., 2019). We compare with CodeBERT (Feng et al., 2020), GraphCodeBERT (Guo et al., 2021) and `cpt-code` (Neelakantan et al., 2022). This setting requires finding the relevant code block among 1K candidates for a given natural language query.