# BOOM: Bagging-Based Model Merging for Robust Text Embeddings

Anonymous ACL submission


## Abstract

General-purpose text embeddings support a wide range of NLP and IR applications, but robust generalization across unseen domains and efficient incorporation of new data remain challenging. Although full-corpus training with batch-level shuffling (BLS) is a simple and effective recipe for heterogeneous training corpora, scaling training data does not necessarily guarantee stronger out-of-domain (OOD) robustness, and continual updates typically require costly retraining over accumulated data. We propose **B**agging-based r**O**bust Embedding via m**O**del **M**erging (**BOOM**), which trains multiple embedding models on sampled views of the corpus and merges them into a single deployable model. This design preserves the diversity benefits of sampled-model training while avoiding the inference cost of conventional ensembles. BOOM also supports continual learning by training lightweight update models on new data with a small historical sample and merging them with the existing model. Experiments show that BOOM improves in-domain and OOD performance over strong BLS baselines, and in continual learning, achieves stronger results with substantially lower training cost than full retraining.


## 1 Introduction

General-purpose text embeddings encode diverse linguistic information into dense vectors and serve as a foundation for a wide range of NLP and IR applications, including retrieval, classification, clustering, and semantic textual similarity (Izacard et al., 2021; Wang et al., 2022; Xiao et al., 2024; Muennighoff, 2022; Neelakantan et al., 2022; Li et al., 2023b). Their rapid progress has been driven in part by standardized evaluations such as the Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2023; Enevoldsen et al., 2025), which assesses models across diverse tasks and domains. A central challenge in developing general-

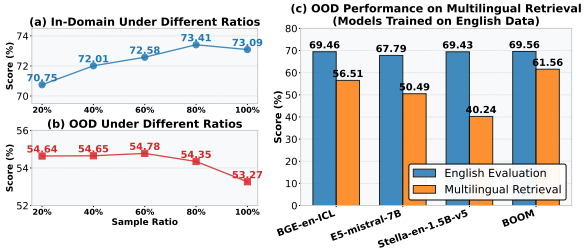


Figure 1: (a)&(b) Average performance (%) of general text embedding models trained with different proportions of the multi-task training set on in-domain and OOD evaluation sets. (c) Average performance (%) of different embedding models trained on the English data.

purpose embeddings is robust generalization: models should perform well not only on familiar benchmark distributions, but also on unseen domains, task mixtures, and data types.

To improve coverage, current embedding models (Li et al., 2024b; Yu et al., 2025) are typically trained on large-scale multi-task corpora that mix retrieval, classification, clustering, and similarity data under contrastive learning objectives (Lee et al., 2024, 2025; Li et al., 2024b; Zhang et al., 2025c). This scaling strategy creates two practical challenges. First, adding more heterogeneous training data does not automatically translate into stronger out-of-domain (OOD) robustness: a model can fit the dominant patterns of a large training corpus and still transfer poorly to shifted distributions. Second, deployed embedding systems are rarely static. New domains such as legal or financial search, as well as new data types such as code retrieval, continually emerge and must be incorporated without repeatedly retraining on all historical data.

Batch-level shuffling (BLS) is the standard single-model recipe for training on such heterogeneous corpora. It constructs each batch from one dataset and shuffles batches across datasets, making it simple, scalable, and effective for exposing the model to diverse supervision throughout training (Li et al., 2024b, 2023b). However, BLS-

based training is still a full-corpus single-model paradigm. As shown in Figure 1 (a)&(b), models trained on the full dataset achieve strong in-domain performance, yet can underperform models trained on substantially smaller subsets (e.g., 20% of the data) on OOD evaluations. Figure 1 (c) further shows that strong performance on standard in-domain benchmarks does not consistently imply strong OOD performance. These observations suggest that the core challenge is not merely relying on a single full-corpus training recipe, but building embedding models that are robust to distribution shift and efficient to update as the training corpus evolves.

These challenges motivate a shift from indiscriminate data scaling toward explicitly improving robustness and update efficiency. A classical approach for improving robustness is bootstrap aggregating, or bagging, which trains multiple models on different sampled subsets to reduce variance (Breiman, 1996). However, conventional bagging is unattractive for embedding services because it requires multi-model inference, increasing latency and deployment cost. Model merging offers a natural alternative: it can consolidate multiple trained models into a single deployable model, potentially preserving the robustness benefits of sampled-model diversity without ensemble-time overhead.

Motivated by this perspective, we propose **B**agging-based r**O**bust Embedding via m**O**del **M**erging (**BOOM**) for general-purpose text embeddings. BOOM first trains multiple embedding models on differently sampled subsets of the training corpus using the same underlying embedding-training recipe, and then merges them into a single embedding model. Although this introduces additional offline training runs, these runs are independent and can be executed in parallel across sampled subsets. More importantly, the final system remains a single deployable embedding model, incurring no additional inference-time latency or serving cost compared with standard single-model baselines. This design retains the diversity induced by subset training while avoiding ensemble inference overhead, leading to improved robustness across diverse unseen domains and task settings. Moreover, BOOM naturally supports continual learning: when new data arrives, we train a lightweight update model on the new data together with a small sampled subset of historical data, and then merge it with the existing model. This enables efficient integration of new knowledge while mitigating catastrophic forgetting, without full retraining over all past data.

We evaluate BOOM against strong BLS-based baselines across multiple embedding benchmarks. Empirically, BOOM consistently outperforms full-corpus single-model training, improving both in-domain and OOD performance. In continual learning settings, BOOM further enables efficient model updates through lightweight training and merging, achieving stronger results while substantially reducing training cost relative to full retraining.

In summary, our contributions are as follows: (1) We identify two practical challenges in general-purpose text embedding training: full-corpus training does not guarantee OOD robustness, and continual updates require costly retraining. (2) We propose BOOM, a bagging-and-merging framework that trains models on sampled corpus views and merges them into one deployable embedding model, improving robustness without ensemble inference cost. (3) We validate BOOM across static and continual learning settings, showing consistent gains over strong BLS baselines and, in continual updates, substantially lower training cost than full retraining.

## 2 Related Work

### 2.1 General Text Embedding

General text embedding represents a critical research direction in information retrieval and natural language processing. The prevailing approach employs a dual-encoder framework, in which queries and documents are encoded independently. The cosine similarity between their embeddings then serves as an estimate of semantic relevance, forming the core methodological principle behind general text embedding.

**PLM-based Embedding.** During the development of pre-trained language models (PLMs) such as BERT and T5, numerous impactful methods have been proposed to advance the use of text embeddings for general tasks, e.g., Contriever (Izacard et al., 2021), E5 (Wang et al., 2022), BGE (Xiao et al., 2024), SGPT (Muennighoff, 2022), Open Text Embedding (Neelakantan et al., 2022), and GTE (Li et al., 2023b). These models typically undergo weakly-supervised contrastive training on large datasets of text pairs collected from diverse web sources (such as citation graphs and Reddit), followed by supervised fine-tuning on labeled

datasets.

**LLM-based Emebdding.** Large language models (LLMs) have demonstrated strong performance across a variety of NLP tasks, and recent studies increasingly explore their potential as backbone encoders for text embedding (Zhang et al., 2025a; Ma et al., 2024; BehnamGhader et al., 2024; Springer et al., 2024; Li et al., 2024a; Zhang et al., 2025b; Li et al., 2025; Lee et al., 2024, 2025; Zhang et al., 2025c). Early work, such as Repllama (Ma et al., 2024), first employed LLMs for embedding generation, showing substantial improvements over traditional pretrained language model (PLM)-based approaches. However, the causal attention mechanism in decoder-only LLMs may constrain their ability to produce highly contextualized embeddings. To address this, LLM2vec(BehnamGhader et al., 2024) replaced causal attention with bidirectional attention and introduced a masked next-token prediction (MNTP) warm-up strategy. LLM-QL (Zhang et al., 2025b) leverages the generative strengths of LLMs through QL maximization with Attention Block and Document Corruption, which acts as a preparation step to better warm up the model for the following contrastive training. Conan-Embedding-v2 (Li et al., 2025) proposes a novel soft masking mechanism combined with dynamic rank reduction. NV-Embed (Lee et al., 2024) introduced latent attention pooling and Two-stage training for state-of-the-art representation quality. Gemini embedding (Lee et al., 2025) employed two-stage training, i.e., pre-finetuning on larger-scale weak-supervised data and fine-tuning on high-quality supervised data. Qwen3-Embedding (Zhang et al., 2025c) integrated large-scale LLM-generated synthetic data, multi-stage fine-tuning to deliver better embeddings.

### 2.2 Ensemble Learning

Ensemble learning (Dong et al., 2020; Sagi and Rokach, 2018) is a powerful machine learning paradigm where multiple models (often referred to as "base learners") are combined to solve a problem and produce a better performance than individual models. Bootstrap Aggregating (Bagging) is one of the most popular ensemble learning techniques that combines multiple models to improve predictive accuracy, especially for high-variance, low-bias models like decision trees. Introduced by (Breiman, 1996), the core idea of bagging is to reduce variance and avoid overfitting by aggregating the predictions of several models trained on different subsets of the data. Each model is trained on a bootstrapped sample, which is a randomly chosen subset of the training data with replacement. However, the high computational and inference cost of bagging remains a significant drawback, especially for tasks requiring real-time predictions using large models, e.g., LLMs.

## 3 Preliminary

### 3.1 Model Merging

Model merging is a powerful technique that combines the strengths of multiple models without the computational cost of ensembling or the need for further training. A central challenge lies in mitigating conflicts among the models to be merged. In this section, we present a simple yet effective method called Multi-SLERP (Goddard et al., 2024a). Additional methods are described in Appendix C.

**Multi-SLERP (Goddard et al., 2024a).** Multi-SLERP extends spherical linear interpolation (SLERP) to $m$ models $\{\mathbf{W}_i\}_{i=1}^m$ with barycentric weights $\{\alpha_i\}_{i=1}^m$ (where $\sum_{i=1}^m \alpha_i = 1$) as follows:

$$\mathbf{W}_{\text{merged}} = \left(\sum_{i=1}^{N} \alpha_i \|\mathbf{W}_i\|\right) \cdot \exp_{\mathbf{M}}\left(\sum_{i=1}^{N} \alpha_i \log_{\mathbf{M}}\left(\frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}\right)\right), \quad (1)$$

where $\mathbf{M}$ is the normalized weighted mean direction,

$$\mathbf{M} = \frac{\sum_{i=1}^{N} \alpha_i \frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}}{\left\|\sum_{i=1}^{N} \alpha_i \frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}\right\|}. \quad (2)$$

Here $\log_{\mathbf{M}}(\cdot)$ and $\exp_{\mathbf{M}}(\cdot)$ denote the logarithmic and exponential maps at $\mathbf{M}$ on the sphere.

### 3.2 Weight Disentanglement

Many works focus on merger-friendly fine-tuning (Yang et al., 2024), which designs training strategies that produce models more amenable to merging than standard fine-tuning, with minimal performance degradation after merging. One necessary condition for effective model merging is weight disentanglement (Ortiz-Jimenez et al., 2023). For a pre-trained model $f(\cdot;\boldsymbol{\theta}_0)$ and task vectors $\{\boldsymbol{\tau}_t\}$ fine-tuned on tasks with disjoint supports $\mathcal{D}_t$, weight disentanglement means that

$$f\Big(\mathbf{x};\boldsymbol{\theta}_0 + \sum_t \alpha_t \boldsymbol{\tau}_t\Big) \approx \sum_t f_t(\mathbf{x};\alpha_t\boldsymbol{\tau}_t) + f_0(\mathbf{x}). \quad (3)$$

### 3.3 General Text Embedding Training

Given the query $q$ from the $K$ training datasets, the general text embedding model mainly uses the InfoNCE loss function:

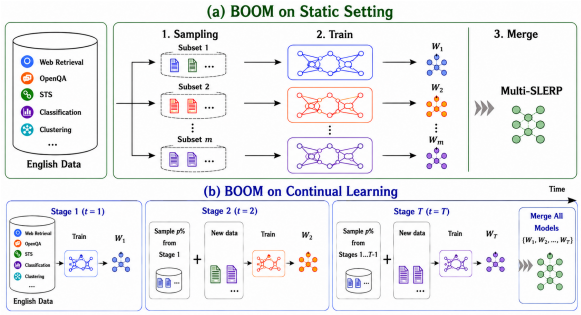


Figure 2: Overall framework of BOOM.

$$L = -\log \frac{\exp(s(q, d^+))}{\exp(s(q, d^+)) + \sum \exp(s(q, D^-))}, \quad (4)$$

where $d^+$ means the positive document, $D^-$ denotes the set of negative documents, and $s(q, d^+)$ is the cosine similarity between the query embedding $e_q$ and document embedding $e_d$. For non-retrieval tasks, e.g., classification or clustering, applying the in-batch negatives approach can mislead the embedding model, as the "document" within a mini-batch may belong to the same class and does not constitute true negatives. So, different negative sampling strategies for different tasks during training. For non-retrieval tasks (e.g., classification and clustering), $D^-$ contains only hard negative documents. For retrieval tasks, $D^-$ includes both hard negatives and in-batch negatives.

## 4 Methodology

General-purpose text embedding models commonly adopt Batch-Level Shuffling (BLS), where each batch is sampled from a single dataset, and batches are shuffled to approximate a diverse training distribution (Li et al., 2024b, 2023b). While effective, BLS often provides limited robustness under distribution shifts and requires expensive retraining over all historical data in continual learning settings. We instead introduce a bagging-based training method for embedding models. The key idea is to train multiple models on different sampled views of the corpus and consolidate them into a single model through parameter merging. Compared with conventional ensembles, our framework preserves ensemble-like robustness without incurring multi-model inference cost.

### 4.1 Static Setting

Let the complete training corpus be $\mathcal{D} = \{D_1, D_2, \ldots, D_N\}$, containing heterogeneous tasks such as retrieval, classification, and clustering. All models share the same architecture, tokenizer, and optimization setup to enable parameter-space fusion.

**Parallel Training with Sampled Data.** We construct $m$ sampled views of the training corpus and train $m$ models in parallel from a common initialization $W_0$. To balance generality and specialization, we consider two sampling strategies.

**Unified Sampling.** We directly sample from the mixed corpus without considering task categories. For model $i$, the sampled corpus is $\mathcal{D}^i = \text{Sample}(\mathcal{D}, k_i)$, where $k_i$ denotes the sampling ratio. Sampling can be performed either with replacement ($k_i$) or without replacement ($k_i'$). Since all models observe similar global distributions but different instances, they converge to nearby yet distinct local optima, introducing moderate diversity while preserving broad task coverage.

**Task-aware Sampling.** To induce stronger specialization, we partition the corpus into retrieval data $\mathcal{D}^R$ and non-retrieval data $\mathcal{D}^N$ and sample them separately:

$$\mathcal{D}^i = \text{Sample}(\mathcal{D}^R, k_i^R) \cup \text{Sample}(\mathcal{D}^N, k_i^N). \quad (5)$$

Different sampling ratios bias models toward different representation behaviors: retrieval-heavy models strengthen semantic matching, while non-retrieval-heavy models favor structures useful for classification and clustering.

**Parameter-space Fusion.** After sampling, we train all models with standard BLS on $\{\mathcal{D}^1, ..., \mathcal{D}^m\}$ and obtain $\{W_1, \ldots, W_m\}$. The trained models are consolidated into a single model:

$$W_{\text{static}} = \mathcal{M}(W_1, W_2, \ldots, W_m), \quad (6)$$

where $\mathcal{M}$ is the model merging method. Unlike conventional model merging, which often combines models trained on disjoint tasks and therefore faces severe parameter conflicts, our models are trained on different views of the same underlying distribution. Consequently, parameter interference is substantially reduced, and merging primarily consolidates compatible local optima. Empirically, different merging techniques yield very similar performance (Appendix C), so we adopt a simple method, i.e., Multi-SLERP as the default strategy. This indicates that models trained on different bootstrap samples exhibit low parameter conflict, allowing even naive merging to be effective.

### 4.2 Continual Learning

We further extend BOOM to a multi-stage continual learning setting, where new datasets arrive sequentially over time. Suppose an initial model

$W_1$ is trained on the original training data $\mathcal{D}_1$. At stage $t \in \{2, \ldots, T\}$, a new dataset collection $\mathcal{D}_t$ becomes available. Our goal is to incorporate new knowledge efficiently without retraining on the accumulated training data $\cup_{i=1}^{t} \mathcal{D}_i$.

**Core Subset Construction and Training.** To preserve historical knowledge with bounded cost, we construct a lightweight replay set by sampling a small subset from all previously observed data: $\mathcal{D}_{\text{core}}^{t} = \bigcup_{i=1}^{t} \text{Sample}(\mathcal{D}_i, k_{\text{core}})$, where $k_{\text{core}}$ denotes the sampling ratio. A stage-specific model is then trained on the union of the new data and replayed samples: $W_t \leftarrow \text{Train}(\mathcal{D}_t \cup \mathcal{D}_{\text{core}}^{t})$.

**Progressive Model Fusion.** After training, the new model is consolidated with the previously $t-1$ accumulated models through parameter-space fusion:

$$W_{\text{final}} = \mathcal{M}(W_1, W_2, \ldots, W_t). \tag{7}$$

Compared with full retraining, this strategy only requires training on newly introduced data and a small replay set. Since all stage-specific models are trained from the same initialization and share partially overlapping data distributions, parameter conflicts remain limited, making simple merging strategies effective.

# 5 Experimental Setup

## 5.1 Training Data

We construct separate training datasets for the static and continual learning settings. Detailed descriptions of all datasets are provided in Appendix A.1.

**Static Setting.** We use **Eng-Text-Data**, a curated collection of 31 English datasets covering a diverse range of tasks, including retrieval, reranking, classification, clustering, and so on.

**Continual Learning Setting.** In the continual learning setting, model training is conducted in three sequential stages with different new datasets:

- **Stage 1** ($T = 1$)**:** Train on Eng-Text-Data.
- **Stage 2** ($T = 2$)**:** Introduce approximately 0.8M additional examples spanning multilingual text retrieval and code retrieval tasks. None of the datasets used in this stage overlap with those in the evaluation benchmark.
- **Stage 3** ($T = 3$)**:** Incorporate another 0.8M examples derived from Chinese retrieval and code retrieval datasets. Notably, the newly added code retrieval datasets are from MTEB(Code, v1) training datasets.

## 5.2 Evaluation Benchamrk

To rigorously assess the model's capabilities, we evaluate its performance on a suite of established benchmarks from the Massive Text Embedding Benchmark (MTEB) framework (Muennighoff et al., 2023): (1) MTEB (English, v2), comprising 41 datasets across 7 task types; (2) MTEB (Code, v1), a code retrieval evaluation covering 12 datasets; and (3) RTEB (beta), featuring 16 datasets spanning specialized domains such as legal, financial, healthcare, multilingual, and code. Full details are provided in Appendix A.2.

## 5.3 BOOM Variants

We instantiate the choices described in Section 4 with the following static-setting variants. Unless otherwise stated, each constituent model is trained from the same backbone initialization and merged with Multi-SLERP.

- **{60, 60, 60}**: three uniformly sampling with identical ratios using different seeds, introducing diversity solely through instance-level variation.
- **{20, 40, 60, 80, 100}**: five uniformly sampled with different ratios using the same seed.
- **{50, 50}$_{\text{split}}$**: two complementary models trained on non-overlapping partitions of the corpus.
- **{80, R60, N60}**: a generalist-specialist design where an $80\%$ uniformly sampled model serves as an anchor, while $R60$ and $N60$ are specialized toward retrieval and non-retrieval data respectively.
- **{80, R60N20, N60R20}**: a softer specialization strategy where task-biased models retain partial exposure to the opposite task type.

These variants allow us to investigate the trade-off between broad task coverage and specialized representation learning.

## 5.4 Implementation Details

We adopted Qwen3-0.4B and Qwen3-4B (Yang et al., 2025) as the backbone for our framework. We fine-tune the Qwen3-0.6B and Qwen3-4B models (Yang et al., 2025) using the contrastive loss for a single epoch on the training datasets, respectively. The training is on a machine with 8× Nvidia A800 (80GB) GPUs. For the larger Qwen3-4B model, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2021) for efficient fine-tuning, with the LoRA rank and alpha both set to 32. The learning rate is set to 5e-5, and the batch size is 128. For the smaller

Table 1: Average performance comparison on different MTEB benchmarks under Eng-Text-Data training datasets. Bold and underline indicate the best performance on each task across all models and within the same model scale, respectively.

| Size | Method | MTEB(Eng, v2) | | | | | | | | | | RTEB | Code |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Class. | Clustering | P-CLS | Reranking | Retrieval | STS | Summ. | Mean | IND | OOD | OOD | OOD |
| 1.5B | Stella-en-1.5B-v5 | **89.38** | 57.06 | 88.02 | **50.19** | 52.42 | 83.27 | **36.91** | 69.43 | - | - | 40.24 | 44.93 |
| 7B | BGE-en-ICL | 88.78 | 57.80 | **85.39** | 48.02 | 55.10 | **82.21** | 32.20 | 69.46 | - | - | 56.51 | 59.67 |
| 7B | NV-Embed-v1 | 84.11 | 49.50 | 87.05 | 49.16 | **60.13** | 82.20 | 31.40 | 68.32 | - | - | - | - |
| 7B | E5-mistral-7B-instruct | 79.85 | 51.44 | 88.42 | 49.78 | 57.62 | 84.32 | 36.57 | 67.97 | - | - | 50.49 | - |
| 0.6B | BLS | 86.01 | 54.68 | 81.20 | 45.81 | 50.60 | 78.34 | 31.70 | 65.94 | 73.09 | 56.80 | 46.61 | 56.12 |
| | Two-Stage | 85.95 | 54.18 | 79.96 | 45.87 | 46.23 | 77.46 | 30.85 | 64.46 | 72.40 | 54.32 | 34.65 | 46.38 |
| | BOOM | | | | | | | | | | | | |
| | $w/$ {80, R60, N60 } | 85.01 | 54.34 | 80.69 | 47.21 | 51.35 | 79.77 | 31.09 | 66.19 | 72.68 | 57.90 | 50.46 | 59.88 |
| | $w/$ {80, R60N20, N60R20} | 84.90 | 54.43 | 81.95 | 46.79 | 52.97 | 80.04 | 32.73 | 66.75 | 73.00 | 58.76 | 51.72 | 61.06 |
| | $w/$ $\{50, 50\}_{\text{split}}$ | 85.34 | 55.07 | 80.97 | 46.84 | 53.07 | 79.19 | 31.11 | 66.69 | 73.27 | 58.29 | 52.52 | 61.10 |
| | $w/$ {60, 60, 60} | 85.88 | 55.35 | 82.08 | 47.37 | 53.16 | 79.52 | 31.31 | 67.06 | 73.63 | 58.67 | 52.62 | 61.10 |
| | $w/$ {20, 40, 60,80, 100} | 86.13 | 55.60 | 81.99 | 47.16 | 53.31 | 80.16 | 32.93 | 67.36 | 73.98 | 58.89 | 53.20 | 61.12 |
| 4B | BLS | 87.28 | **59.05** | 82.69 | 49.46 | 55.03 | 81.10 | 35.29 | 69.10 | 75.78 | 60.57 | 60.52 | 65.19 |
| | Two-Stage | 86.88 | 58.98 | 83.25 | 49.37 | 52.42 | 81.16 | 36.58 | 68.45 | 75.66 | 59.24 | 58.57 | 65.21 |
| | BOOM | | | | | | | | | | | | |
| | $w/$ $\{50, 50\}_{\text{split}}$ | 86.76 | 58.58 | 83.13 | 49.15 | 54.77 | 81.48 | 34.26 | 68.92 | 75.48 | 60.53 | 61.34 | 66.54 |
| | $w/$ {60, 60, 60} | 87.38 | 58.77 | 83.46 | 49.25 | 55.04 | 81.90 | 36.47 | 69.32 | 75.87 | 60.95 | 61.27 | 66.67 |
| | $w/$ {20, 40, 60,80,100} | 87.34 | 59.00 | 83.71 | 49.52 | **55.67** | 81.95 | 36.72 | **69.56** | **75.97** | **61.37** | **61.56** | **66.74** |

Qwen3-0.6B model, we apply full-parameter tuning with a learning rate of 5e-5 and a batch size of 256. The following settings are shared between both models: each dataset incorporates 7 hard negatives, which are provided by Li et al. (2024b) and Suresh et al. (2024), the maximum sequence length is limited to 512 tokens, and the same dataset is used consistently within each training step. During model merging, the weight $\alpha_i$ assigned to model $W_i$ is the ratio of its training data size to the total training data size across all models $\{W_i\}_{i=1}^{m}$.

### 5.5 Evaluation Setting

To rigorously evaluate the adaptability and generalization capabilities of our embedding model, we design two distinct experimental settings: **static** and **continual learning**. Furthermore, our evaluation protocol encompasses two distinct test settings to systematically characterize model generalization: **In-Domain (IND)**: Test data is drawn from the same distribution as the training data. **Out-of-Domain (OOD)**: Test datasets exhibit distribution shifts relative to the training corpus, including shifts in domain (e.g., legal, financial), task composition, language, and modality (e.g., code retrieval). While these benchmarks do not represent arbitrary real-world distribution shifts, they provide a practical proxy for evaluating robustness across substantially different data characteristics.

### 5.6 Baselines

- Batch-Level Shuffling (**BLS**) (Li et al., 2024b, 2023b): Each training batch is constructed from a single dataset. These batches are then shuffled throughout training. The training data and backbone are the same as BOOM.
- **Two-Stage Training (Lee et al., 2024)**: Specifically, the first stage involves retrieval-style datasets using in-batch negatives and curated hard-negative examples. The second stage performs on a mixture of sampled retrieval datasets and non-retrieval datasets, this time without the use of in-batch negatives.
- **Open General Text Embedding Model** Using Similar Training Data Size: (1) BGE-en-ICL (Li et al., 2024b, 2023b), which employs in-context learning with few-shot and zero-shot variants, and to ensure a fair comparison, we only evaluate its zero-shot variant, (2) Stella-en-1.5B-v5 (Zhang et al., 2024), (3) NV-Embed-v1 (Lee et al., 2024), and (4) E5-mistral-7B-instruct (Wang et al., 2024).

**BOOM Without Model Merging.** For bagging, the final prediction is obtained by performing inference over multiple models and then combining their embeddings, i.e., Embedding fusion (**Emb-F**): For each model independently, embeddings are obtained for the query $q$ and the document $d$, i.e., $\mathbf{e}_q$ and $\mathbf{e}_d$. The final embeddings for the query and the document are computed as the weighted sum of the corresponding embeddings from all models. The final similarity is then calculated:

$$s(q,d) = \cos\Big(\sum_{i=1}^{M} \alpha_t \cdot \mathbf{e}_q^{(i)}, \sum_{i=1}^{M} \alpha_t \cdot \mathbf{e}_d^{(i)}\Big), \quad (8)$$

Table 2: Performance (%) of BOOM with model merging and without model merging on different benchmarks.

| Benchmark | 0.6B | | | | | 4B | | |
|---|---|---|---|---|---|---|---|---|
| | | BOOM {60,60,60} | | BOOM {20,40,60, 80,100} | | | BOOM {20,40,60, 80,100} | |
| | BLS | w/ Merge | w/o Merge | w/ Merge | w/o Merge | BLS | w/ Merge | w/o Merge |
| Classification | 86.01 | 85.88 | 86.28 | 86.13 | **86.42** | 87.28 | 87.34 | **87.46** |
| Clustering | 54.68 | 55.35 | 55.41 | 55.60 | **55.96** | 59.05 | 59.00 | **59.46** |
| PairClassification | 81.20 | 82.08 | 81.76 | 81.99 | **82.06** | 82.69 | **83.71** | 83.40 |
| Reranking | 45.81 | **47.37** | 46.97 | 47.16 | 46.80 | 49.46 | **49.52** | **49.52** |
| Retrieval | 50.60 | 53.16 | 53.82 | 53.31 | **53.89** | 55.03 | 55.67 | **55.95** |
| STS | 78.34 | 79.52 | 79.10 | **80.16** | 79.89 | 81.10 | **81.95** | 81.64 |
| Summarization | 31.70 | 31.31 | 31.15 | **32.93** | 32.87 | 35.29 | **36.72** | 36.64 |
| MTEB(Eng, v2) (Mean) | 65.94 | 67.06 | 67.17 | 67.36 | **67.55** | 69.10 | 69.56 | **69.65** |
| MTEB(Eng, v2) (IND) | 73.09 | 73.63 | 73.65 | 73.98 | **74.17** | 75.78 | 75.97 | **76.12** |
| MTEB(Eng, v2) (OOD) | 56.80 | 58.67 | 58.89 | 58.89 | **59.09** | 60.57 | 61.37 | **61.38** |
| RTEB(beta) (OOD) | 46.61 | 52.62 | 51.15 | **53.20** | 51.42 | 60.52 | **61.56** | 61.47 |
| MTEB(Code, v1) (OOD) | 56.12 | 61.10 | 59.86 | **61.12** | 60.22 | 65.19 | **66.74** | 66.70 |

where $\alpha_t$ are the fusion weights.

## 6 Static Setting Results

**Overall Performance.** The results are summarized in Table 1. We can find that: (1) Our BOOM variants consistently outperform the standard BLS baseline across both models on both in-domain performance and OOD generalization performance. Notably, the "$\{50, 50\}_{\text{split}}$" variant, which incurs equivalent total training cost to BLS, delivers better or comparable MTEB(Eng, v2) performance alongside improved OOD scores, demonstrating competitive and superior generalization of our BOOM paradigm. Moreover, the "{20, 40, 60, 80, 100}" variant, which incorporates models trained on five different data partitions, achieves the best overall scores, setting the best performance for both models on all benchmarks. (2) Comparing different sampling strategies, unified sampling generally exhibits stronger and more stable performance than task-aware sampling in our experiments. One possible explanation is that models trained with unified sampling remain closer in parameter space because they are optimized on different views of the same overall distribution. In contrast, stronger task specialization can lead to larger optimization divergence, potentially reducing merge compatibility. (3) Although BGE-en-ICL and stella-en-1.5B-v5 achieve stronger performance on MTEB(Eng v1) compared to batch-level shuffling, it shows weaker out-of-distribution (OOD) generalization on RTEB and MTEB(Code, v1). Our BOOM variants consistently outperform these models across all three benchmarks, further indicating the superior generalization ability of our method.

**BOOM Without Model Merging and With Model Merging.** From the Table 2, we can find that (1) The embedding fusion method has the best performance on the IND benchmark. (2) Our merged model achieves performance highly comparable to its corresponding multi-model ensemble in MTEB(Eng, v2), indicating that the parameter-space aggregation preserves the benefits of output-level fusion. One possible explanation is related to observations from weight disentanglement (Ortiz-Jimenez et al., 2023): the output of the merged model can be decomposed as $f(\mathbf{q}/\mathbf{d}; W_{\text{merged}}) \approx \sum_{t=1}^{m} \alpha_t f(\mathbf{q}/\mathbf{d}; W_t)$, where $\sum_{t=1}^{M} \alpha_t = 1$ and each $W_t$ is trained on an independent bootstrap sample of the same data distribution.

These observations suggest that bagging may naturally produce merger-friendly models. One possible explanation comes from prior studies on weight disentanglement (Goddard et al., 2024a), as shown in Section 3. Since our sampled models are trained on different views of the same underlying data distribution rather than entirely different tasks, they may remain closer in parameter space and thus

Table 3: Average performance of different BOOM variants using Qwen3-0.6B trained on Eng-Text-Data, evaluated on MTEB (English, v2). "TC" means the training cost using GPU hours compared to the batch-level shuffling method.

| Dataset Name | TC | Mean(Task) | IND | OOD |
|---|---|---|---|---|
| BLS | 1.0 | 65.94 | 73.09 | 57.28 |
| BOOM $w/$ $\{50, 50\}_{\text{split}}$ | 1.0 | 66.69 | 73.27 | 58.59 |
| BOOM $w/$ {60, 60} | 1.2 | 66.37 | 73.15 | 57.93 |
| BOOM $w/$ {60, 80} | 1.4 | 66.89 | 73.75 | 58.36 |
| BOOM $w/$ {50, 50, 50} | 1.5 | 66.35 | 73.30 | 57.59 |
| BOOM $w/$ {60, 60, 60} | 1.8 | 67.06 | 73.63 | 58.67 |
| BOOM $w/$ {40, 60, 80} | 1.8 | 67.00 | 73.64 | 58.76 |
| BOOM $w/$ {80, 100} | 1.8 | 67.20 | 73.95 | 58.89 |
| BOOM $w/$ {20, 40, 60, 80} | 2.0 | 66.99 | 73.55 | 58.83 |
| BOOM $w/$ {60, 80, 100} | 2.4 | 67.06 | **74.06** | 58.90 |
| BOOM $w/$ {40, 60, 80, 100} | 2.8 | 67.35 | 73.95 | **59.18** |
| BOOM $w/$ {20, 40, 60, 80, 100} | 3.0 | **67.36** | 73.98 | 59.14 |

Table 4: Average performance (%) of different models upon the Qwen3-0.6B under the continual learning setting. More detailed performance comparison can be found in the Table 7 in the Appendix.

| MTEB | Stage 1 | Stage 2 | | Stage 3 | |
|---|---|---|---|---|---|
| | | Re-training | BOOM | Re-training | BOOM |
| (Eng, v2) | 65.94 | 65.62 | 67.20 | 66.28 | 67.31 |
| RTEB | 46.61 | 52.54 | 54.93 | 53.46 | 58.42 |
| (Code, v1) | 56.12 | 62.29 | 64.46 | 71.04 | 71.12 |
| ALL | 59.75 | 62.01 | 63.88 | 64.14 | 65.91 |

Table 5: Average performance (%) of different sampling ratios (%) from the stage 1 data for stage 2 training under the continual learning setting. "(Eng, v2)" and "(Code, v1)" are the "MTEB(Eng, v2)" and "MTEB(Code, v1)", respectively.

| $k$ | (Eng, v2) | RTEB(beta) | (Code, v1) |
|---|---|---|---|
| BOOM ($k$=20) | 66.92 | 54.45 | 63.50 |
| BOOM ($k$=40) | 67.20 | 54.93 | 64.46 |
| BOOM ($k$=60) | 67.05 | 54.63 | 64.26 |

satisfy such conditions more readily.

**Impact of Sampling Configuration.** To investigate the impact of varying the number and composition of merged models, we evaluate several bagging combinations for BOOM (Table 3). Key findings include: (1) All bagging and merging strategies outperform simple batch-level shuffling on both in-domain and out-of-domain (OOD) evaluations, highlighting the effectiveness and robustness of BOOM in leveraging diverse training subsets. (2) Increasing training cost via merging more models generally improves performance on both benchmarks. However, including a model trained on the higher-ratio sampled dataset (e.g., 80%, 100%) is better. For instance, the configuration 80, 100 (TC=1.8) outperforms 20, 40, 60, 80 (TC=2.0) on both metrics, despite the latter having a higher cost but lacking a fully-trained model.

## 7 Continual Learning Setting Results

**Overall Performance.** Our continual learning results in Table 4 demonstrate that BOOM provides an effective and data-efficient framework for incrementally extending text embedding models. We make the following observations: (1) BOOM consistently outperforms full retraining across all continual learning stages while requiring substantially less historical data. Specifically, our method updates the model using only a small core subset of previous data (40% in our experiments) together with newly introduced data, rather than retraining on the entire accumulated corpus. Despite this reduced training cost, BOOM achieves stronger overall performance. (2) Increasing training data gradually improves performance for both approaches; however, BOOM consistently achieves larger gains than full retraining. From Stage 1 to Stage 3, the overall score of full retraining increases from 59.75 to 64.14, while BOOM reaches 65.91. These results suggest that the performance improvements are not solely due to the increased amount of training data, but also arise from the proposed bagging-and-merging update mechanism, which more effectively consolidates complementary knowledge.

**Impact of Core Subset Size.** The continual learning setting involves sampling $k\%$ old data points and comparing them with all new data to train the new model, which is then merged with the old model to obtain the final model. In this section, we evaluate different sampling ratios, as shown in Table 5. Our observations are as follows: if the ratio is very low (e.g., 20%), the merged performance may degrade due to forgetting of the old task. Conversely, if the sampling ratio is too large, it increases the cost of the training dataset. Therefore, we adopt a sampling ratio of 40% in our experiment.

## 8 Conclusion

We present BOOM, a bagging-and-merging framework for training general-purpose text embeddings that addresses two critical limitations of standard batch-level shuffling (BLS) paradigms: brittle out-of-domain (OOD) generalization and costly continual updates. Rather than training a single model on the full multi-task corpus, BOOM trains multiple embedding models on independently sampled subsets and merges them into a single deployable model. This design preserves the diversity induced by subset training, yielding improved robustness across unseen domains and task settings without introducing any inference-time overhead. We further extend BOOM to continual learning scenarios, where new knowledge is incorporated by training a lightweight update model on newly arrived data together with a small historical sample, and then merging it with the existing model. This strategy enables efficient knowledge integration while mitigating catastrophic forgetting, eliminating the need for expensive full-retraining over all past data. Empirically, BOOM consistently outperforms standard BLS training on the full corpus across multiple embedding benchmarks, improving both in-domain and OOD performance on different settings.

## Limitations

There are three primary limitations that should be acknowledged:

1. BOOM adopts relatively simple sampling and merging strategies for efficiency and generality. While our experiments show that these designs are effective across multiple benchmarks, future work could explore more principled subset construction methods and stronger parameter-fusion strategies to further improve performance and scalability.
2. Our study focuses on general-purpose text embeddings and evaluates robustness mainly through benchmark-based in-domain and distribution-shift settings. Although the evaluation covers diverse shifts across domains, tasks, languages, and modalities (e.g., code retrieval), these benchmarks cannot fully represent the wide spectrum of real-world deployment scenarios, such as rapidly evolving corpora, more extreme distribution shifts, or emerging settings involving multimodal representation learning.
3. Our work primarily provides an empirical study of bagging and model merging for text embeddings. While prior studies on merge-friendly fine-tuning and weight disentanglement offer a plausible perspective for understanding the observed behavior, a deeper analysis of how sampled data views affect parameter geometry and mergeability remains an interesting direction for future work.

## 9 GenAI Usage Disclosure

During the preparation of this work, the authors used ChatGPT (GPT-4o) and DeepSeek solely for writing assistance and language polishing to improve the clarity, grammar, and readability of selected paragraphs. No generative AI tools were used for any research-related activities, including experimental design, data generation, data analysis, or interpretation of results, nor for substantive content creation in the manuscript. The authors remain fully responsible for all scientific content and ethical aspects of this work.

## References


C.J. Adams, Daniel Borkan, Jeffrey Soreson, Lucas Dixon, Lucy Vasserman, and Nithum Thain. 2019. Jigsaw unintended bias in toxicity classification. *URL https://kaggle.com/ competitions/jigsaw-unintended-bias-in-toxicity-classification.*

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. in* sem 2012: The first joint conference on lexical and computational semantics–volume 1: Proceedings of the main conference and the shared task, and volume 2: Proceedings of the sixth international workshop on semantic evaluation (semeval 2012). In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012), Montréal, QC, Canada*, pages 7–8.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Leo Breiman. 1996. Bagging predictors. *Mach. Learn.*, 24(2):123–140.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Xi Chen, Ali Zeynali, Chico Camargo, Fabian Flöck, Devin Gaffney, Przemyslaw Grabowicz, Scott A Hale, David Jurgens, and Mattia Samory. 2022. Semeval-2022 task 8: Multilingual news article similarity. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1094–1106.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. 2020. Specter: Document-level representation learning using citation-informed transformers. In *ACL*.

Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. 2020. A survey on ensemble learning. *Frontiers Comput. Sci.*, 14(2):241–258.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, and 1 others. 2025. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024a. Arcee's mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. 2024b. Arcee's MergeKit: A toolkit for merging large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US. Association for Computational Linguistics.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, and 1 others. 2018. Dureader: a chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the workshop on machine reading for question answering*, pages 37–46.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, and 1 others. 2021. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Junjie Huang, Duyu Tang, Linjun Shou, and 1 others. 2021. Cosqa: 20,000+ web queries for code search and question answering. *arXiv preprint arXiv:2105.13239*.

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

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. 2024. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pages 207–223. Springer.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

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*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995*, pages 331–339. Elsevier.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, and 1 others. 2025. Gemini embedding: Generalizable embeddings from gemini. *arXiv preprint arXiv:2503.07891*.

Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. 2024a. Llama2vec: Unsupervised adaptation of large language models for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3490–3500.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024b. Making text embedders few-shot learners. *arXiv preprint arXiv:2409.15700*.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. Mtop: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962.

Jianquan Li, Xidong Wang, Xiangbo Wu, Zhiyi Zhang, Xiaolong Xu, Jie Fu, Prayag Tiwari, Xiang Wan, and Benyou Wang. 2023a. Huatuo-26m, a large-scale chinese medical qa dataset. *Preprint*, arXiv:2305.01526.

Shiyu Li, Yang Tang, Ruijie Liu, Shi-Zhe Chen, and Xi Chen. 2025. Conan-embedding-v2: Training an llm from scratch for text embeddings. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 15011–15027.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023b. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Xueqing Liu, Chi Wang, Yue Leng, and ChengXiang Zhai. 2018. Linkso: a dataset for learning to retrieve similar question answer pairs on software development forums. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering*, pages 2–5.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2421–2425.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

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

Niklas Muennighoff, Nouamane Tazi, Loïc 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.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, and 1 others. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

James O'Neill, Polina Rozenshtein, Ryuichi Kiryo, Motoko Kubota, and Danushka Bollegala. 2021. I wish i would have loved this one, but i didn't–a multilingual dataset for counterfactual detection in product reviews. *arXiv preprint arXiv:2104.06893*.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2023. Task arithmetic in the tangent space: Improved editing of pre-trained models. *Advances in Neural Information Processing Systems*, 36:66727–66754.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *WIREs Data Mining Knowl. Discov.*, 8(4).

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3687–3697.

Ken Shoemake. 1985. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2024. Repetition improves language model embeddings. *arXiv preprint arXiv:2402.15449*.

Tarun Suresh, Revanth Gangi Reddy, Yifei Xu, Zach Nussbaum, Andriy Mulyar, Brandon Duderstadt, and Heng Ji. 2024. Cornstack: High-quality contrastive data for better code ranking. *arXiv e-prints*, pages arXiv–2412.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251.

Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. 2025. Fusechat: Knowledge fusion of chat models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21629–21653.

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. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916.

Phil Culliton. Wei Chen Maggie. 2020. Tweet sentiment extraction. *URL https://kaggle.com/competitions/tweet-sentiment-extraction*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Xiaohui Xie, Qian Dong, Bingning Wang, Feiyang Lv, Ting Yao, Weinan Gan, Zhijing Wu, Xiangsheng Li, Haitao Li, Yiqun Liu, and 1 others. 2023. T2ranking: A large-scale chinese benchmark for passage ranking. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2681–2690.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115.

Weixiang Yan, Yuchen Tian, Yunzhe Li, and 1 others. 2023. Codetransocean: A comprehensive multilingual benchmark for code translation. *arXiv preprint arXiv:2310.04951*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Peng Yu, En Xu, Bin Chen, Haibiao Chen, and Yinfei Xu. 2025. Qzhou-embedding technical report. *arXiv preprint arXiv:2508.21632*.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2024. Jasper and stella: distillation of sota embedding models. *arXiv preprint arXiv:2412.19048*.

Hengran Zhang, Keping Bi, and Jiafeng Guo. 2025a. A comparative study of specialized llms as dense retrievers. In *China Conference on Information Retrieval*, pages 49–61. Springer.

Hengran Zhang, Keping Bi, Jiafeng Guo, Xiaojie Sun, Shihao Liu, Daiting Shi, Dawei Yin, and Xueqi Cheng. 2025b. Unleashing the power of llms in dense retrieval with query likelihood modeling. *arXiv preprint arXiv:2504.05216*.

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

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. Miracl: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025c. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Tianyu Zheng, Ge Zhang, Tianhao Shen, and 1 others. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*.

# A Datasets

## A.1 Training Datasets

**Training Data for Static Setting.** We adopt the extensively curated datasets, i.e., Eng-Text-Data, which integrate multiple publicly available benchmarks across several key tasks:

- **Retrieval:** ELI5 (Fan et al., 2019), HotpotQA (Yang et al., 2018), FEVER (Thorne et al., 2018), MSMARCO passage and document ranking (Bajaj et al., 2016), NQ (Kwiatkowski et al., 2019), NLI, SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), and FiQA (Maia et al., 2018).
- **Reranking:** StackOverFlowDupQuestions (Liu et al., 2018).
- **Classification:** AmazonReviews-Classification (McAuley and Leskovec, 2013), Banking77-Classification (Casanueva et al., 2020), Emotion-Classification (Saravia et al., 2018), MTOPIntent-Classification (Li et al., 2021), IMDB-Classification (Maas et al., 2011), ToxicConversations-Classification (Adams et al., 2019), TweetSentimentExtraction-Classification (Wei Chen Maggie, 2020), and AmazonCounterfactual-Classification (O'Neill et al., 2021).
- **Clustering:** Arxiv / Biorxiv / Medrxiv / Reddit / StackExchange-Clustering (S2S and P2P), and TwentyNewsgroups-Clustering (Lang, 1995).

- **Semantic Text Similarity (STS):** STS12 (Agirre et al., 2012), STS22 (Chen et al., 2022), and STS-Benchmark (Cer et al., 2017).

The original BGE-en-ICL training data (Li et al., 2024b) includes Quora Duplicate Questions (Thakur et al., 2021), SCIDOCS-RR (Cohan et al., 2020), and ArguAna (Wachsmuth et al., 2018). Because MTEB provides only test and development splits for these datasets (no training data), we exclude them from Eng-Text-Data to avoid potential data contamination.

**Training Data for Incremental Learning.** For the first stage, we employ the Eng-Text-Data as training datasets. For the second stage, we expand the training corpus with multilingual retrieval datasets to enhance cross-lingual generalization. These include DuReader (He et al., 2018), MIRACL (Zhang et al., 2023), Mr. TyDi (Zhang et al., 2021), and T2-Ranking (Xie et al., 2023). Furthermore, to support code retrieval capabilities, we incorporate code-specific training examples sourced from Suresh et al. (2024). Specifically, we sample approximately 10,000 training queries for each of the following programming languages: JavaScript, Java, Python, PHP, and Ruby. This combined dataset ensures coverage across diverse languages, formats, and task types, contributing to a robust and generalizable model training process. For the third stage, we further augment the training data with a Chinese retrieval dataset[1] (Li et al., 2023a) and additional code retrieval data, organized as follows:

1. **Text-to-Code Retrieval (T2C):**
   - APPS (Hendrycks et al., 2021), a diverse collection of problems from programming platforms;
   - CosQA (Huang et al., 2021), containing labeled pairs of textual queries and Python functions;
   - Synthetic-Text2Sql, the largest and most diverse synthetic dataset.
2. **Code-to-Code Retrieval (C2C):**
   - CodeTrans (Yan et al., 2023), which includes code in the same programming language (CodeTrans-Contest) and across different languages (CodeTrans-DL);
   - CodeSearchNet-CCR, a modified version of the original CodeSearchNet.
3. **Code-to-Text Retrieval (C2T):**
   - CodeSearchNet (Husain et al., 2019), consisting of numerous code functions accompanied by code comments.
4. **Hybrid Code Retrieval (Hybrid):**
   - Stackoverflow-QA[2], derived from the original StackOverflow dataset by pairing each question with its most upvoted answer;
   - CodeFeedback (Zheng et al., 2024), a synthesized code instruction dataset generated by LLMs, consisting of CodeFeedback-MT and CodeFeedback-ST.

From these Chinese retrieval and code retrieval datasets, we sample approximately 0.8 million examples for the third stage.

### A.2 Evaluation Benchmarks

**MTEB(English, v2).** This updated English benchmark offers a more realistic assessment of model generalization compared to its predecessor (v1). Its scope includes 41 datasets across 7 task types (i.e., retrieval, classification, clustering, Pair Classification (P-CLS), Reranking, STS, Summarization (Summ.)). MTEB provides two evaluation metrics: Mean (Task) and Mean (Task Type). Mean (Task) is calculated as the average performance across all tasks within the benchmark. Mean (Task Type) is computed by first averaging the results within each task category and then averaging across all categories.

**MTEB (Code, v1).** This specialized benchmark focuses on evaluating embedding models in the context of software engineering. It covers code retrieval tasks across a wide array of popular programming languages, structured into 12 datasets.

**RTEB (Retrieval Text Embedding Benchmark, Beta).** This benchmark focuses on retrieval tasks within high-stakes, specialized domains such as legal, financial, healthcare, multilingual, and code. It includes both open and closed datasets, offering a robust framework for evaluating real-world applicability. We use the open datasets, including 16 datasets, for our evaluation.

Moreover, different tasks within each benchmark use distinct evaluation metrics. For example, retrieval tasks use NDCG@10 (Järvelin and Kekäläinen, 2002), classification tasks use Accuracy, and reranking tasks use MAP@1000. Each dataset may contain multiple subsets, and the score for each dataset is obtained by averaging the scores across its subsets. The final score for each benchmark is

[1] https://huggingface.co/datasets/infgrad/retrieval_data_llm

[2] https://www.kaggle.com/datasets/stackoverflow/stacksample/data

then calculated by averaging the scores across different tasks or datasets, which makes it less suitable for significance testing.

## B Model Merging Method

The following section outlines the key approaches available in MergeKit (Goddard et al., 2024b), which is a comprehensive open-source library for model merging.

### B.0.1 Spherical Interpolation Methods

Spherical interpolation methods are designed to merge or interpolate model weight vectors while respecting their geometric structure on the hypersphere.

- **Spherical Linear Interpolation (SLERP).** SLERP (Shoemake, 1985) interpolates between two model $\mathbf{W}_A$ and $\mathbf{W}_B$ on the hypersphere using

$$\mathbf{W}_{\text{slerp}} = \frac{\sin((1-\alpha)\theta)}{\sin\theta}\mathbf{W}_A + \frac{\sin(\alpha\theta)}{\sin\theta}\mathbf{W}_B, \quad (9)$$

where $\theta = \arccos\left(\frac{\mathbf{W}_A \cdot \mathbf{W}_B}{\|\mathbf{W}_A\|\|\mathbf{W}_B\|}\right)$ and $\alpha \in [0, 1]$. SLERP is inherently a pairwise interpolation method, which means it can directly merge only two models at a time.

- **Multi-SLERP (Goddard et al., 2024a).** Multi-SLERP generalizes Spherical Linear Interpolation (SLERP) to $m$ models $\{\mathbf{W}_i\}_{i=1}^m$ with barycentric weights $\{\alpha_i\}_{i=1}^m$ ($\sum_{i=1}^m \alpha_i = 1$) as follows:

$$\mathbf{W}_{\text{merged}} = \left(\sum_{i=1}^N \alpha_i \|\mathbf{W}_i\|\right) \cdot \exp_{\mathbf{M}}\left(\sum_{i=1}^N \alpha_i \log_{\mathbf{M}}\left(\frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}\right)\right), \quad (10)$$

where $\mathbf{M}$ is the normalized weighted mean direction:

$$\mathbf{M} = \frac{\sum_{i=1}^N \alpha_i \frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}}{\left\|\sum_{i=1}^N \alpha_i \frac{\mathbf{W}_i}{\|\mathbf{W}_i\|}\right\|}. \quad (11)$$

Here, $\log_{\mathbf{M}}(\cdot)$ and $\exp_{\mathbf{M}}(\cdot)$ denote the logarithmic and exponential maps at $\mathbf{M}$ on the sphere.

- **Karcher Mean** (Goddard et al., 2024a). The merged model $\mathbf{W}_{\text{merged}}$ is defined as the point on the hypersphere that minimizes the sum of squared geodesic distances to the input vectors. Formally,

$$\mathbf{W}_{\text{merged}} = \arg\min_{\mathbf{W}\in\mathcal{S}} \sum_{i=1}^N d^2(\mathbf{W}, \mathbf{W}_i), \quad (12)$$

where $\mathcal{S}$ denotes the unit hypersphere and $d(\cdot,\cdot)$ represents the geodesic (angular) distance between two points on the sphere. This approach produces a consensus model that faithfully captures the central tendency of the input models in the underlying spherical geometry.

- **Model Stock**. Model Stock (Jang et al., 2024) moves the merged weights toward the geometric center of a set of fine-tuned checkpoints. Specifically, model stock computes optimal interpolation between a base model $\mathbf{W}_0$ and the average of fine-tuned models $\overline{\mathbf{W}}$. Using the average cosine similarity $\overline{\cos\theta}$ between task vectors:

$$t = \frac{N\overline{\cos\theta}}{1 + (N-1)\overline{\cos\theta}}, \quad \mathbf{W}_{\text{merged}} = t\overline{\mathbf{W}} + (1-t)\mathbf{W}_0. \quad (13)$$

### B.0.2 Task Vector Based Methods

The following methods build on the notion of "task vectors", defined as the difference between a fine-tuned model and its base model.

- **Task Arithmetic**. Task Arithmetic (Ilharco et al., 2022) defines task vectors $\tau_i = \mathbf{W}_i - \mathbf{W}_{\text{base}}$ and combines them with weights $\alpha_i$:

$$\mathbf{W}_{\text{merged}} = \mathbf{W}_{\text{base}} + \sum_{i=1}^N \alpha_i \tau_i. \quad (14)$$

- **TIES Merging**. TIES (Yadav et al., 2023) also uses task vectors, but attempts to mitigate conflicts between merges in weight space. TIES proceeds in three stages: (1)Trim: For each layer, retain only the top-$k\%$ largest-magnitude entries in $\tau_i$, set the rest to zero: $\tau_i' = \text{TopK}(\tau_i, k)$; (2) Sign Selection: Compute the sign consensus for each parameter across all $\tau_i'$. Mask out parameters that disagree with the consensus: $\mathbf{m}_{\text{consensus}} = \text{sign}\left(\sum_{i=1}^N \alpha_i \tau_i'\right)$; (3) Merge:

$$\mathbf{W}_{\text{merged}} = \mathbf{W}_{\text{base}} + \frac{\sum_{i=1}^N \alpha_i(\tau_i' \odot \mathbf{m})}{\sum_{i=1}^N \alpha_i \mathbf{m}} \quad (15)$$

- **SCE**. Unlike TIES, which first performs layer-wise magnitude pruning (TopK) before consensus filtering, SCE (Wan et al., 2025) operates on the global sign consistency directly. For each parameter, retain only entries where all task vectors share the same sign, and mask out conflicting dimensions:

$$\mathbf{m}_{\text{sign}} = \mathbb{I}\left(\text{AllSameSign}\left(\{\boldsymbol{\tau}_i\}_{i=1}^N\right)\right), \boldsymbol{\tau}_i' = \boldsymbol{\tau}_i \odot \mathbf{m}_{\text{sign}}, \quad (16)$$

and then merge: The surviving updates are then aggregated via a weighted sum and normalization:

$$\mathbf{W}_{\text{merged}} = \mathbf{W}_{\text{base}} + \frac{\sum_{i=1}^N \alpha_i \boldsymbol{\tau} i'}{\sum i = 1^N \alpha_i \mathbf{m}_{\text{sign}}} \quad (17)$$

Table 6: Performance (%) of BOOM using different model merging methods on MTEB(Eng, v2).

| Benchmark | TIES Merging | Task Arithmetic | Multi-SLERP | Model Stock | Karcher Mean | SCE |
|---|---|---|---|---|---|---|
| Classification | 87.43 | 87.40 | 87.34 | 86.80 | 87.18 | 87.30 |
| Clustering | 59.08 | 59.30 | 59.00 | 58.28 | 58.69 | 59.24 |
| PairClassification | 83.64 | 83.54 | 83.71 | 82.86 | 83.66 | 83.72 |
| Reranking | 49.57 | 49.59 | 49.52 | 49.21 | 49.57 | 49.73 |
| Retrieval | 55.48 | 55.86 | 55.67 | 55.27 | 55.51 | 55.42 |
| STS | 81.84 | 81.69 | 81.95 | 81.29 | 81.93 | 81.93 |
| Summarization | 35.43 | 36.62 | 36.72 | 38.23 | 36.88 | 35.29 |
| Mean | 69.49 | 69.61 | 69.56 | 69.03 | 69.43 | 69.51 |

## C Different Model Merging Method for BOOM

We evaluate BOOM using several representative model merging methods, as introduced in Section 3. The results are reported in Table 6. Across all methods, the mean scores range only from 69.03% to 69.61%, and the task-level variations are similarly minor. This consistently small gap confirms that alternative merging strategies have little effect on the final performance of BOOM, which is expected because all constituent models are trained on the same data distribution and thus exhibit minimal parameter conflicts. Given this insensitivity, we adopt the simplest Multi-SLERP method throughout our framework for uniformity.

Table 7: Performance (%) of embedding models in different stages.

| Task | Stage1 | Stage2 | | Stage3 | |
|---|---|---|---|---|---|
| | | BLS | BOOM | BLS | BOOM |
| AmazonCounterfactualClassification | 93.51 | 92.72 | 92.25 | 94.28 | 91.60 |
| ArXivHierarchicalClusteringP2P | 63.68 | 63.67 | 64.40 | 63.47 | 63.50 |
| ArXivHierarchicalClusteringS2S | 60.06 | 60.61 | 61.88 | 59.24 | 62.11 |
| ArguAna | 48.62 | 46.92 | 53.15 | 48.38 | 53.62 |
| AskUbuntuDupQuestions | 60.09 | 60.68 | 61.90 | 61.52 | 62.34 |
| BIOSSES | 82.14 | 82.85 | 84.18 | 84.92 | 84.05 |
| Banking77Classification | 86.89 | 86.96 | 87.10 | 87.70 | 86.71 |
| BiorxivClusteringP2P.v2 | 50.12 | 49.89 | 51.23 | 50.16 | 51.06 |
| CQADupstackGamingRetrieval | 50.58 | 52.56 | 54.25 | 52.82 | 55.31 |
| CQADupstackUnixRetrieval | 37.36 | 39.17 | 41.92 | 40.61 | 42.54 |
| ClimateFEVERHardNegatives | 39.74 | 40.15 | 39.32 | 42.14 | 38.25 |
| FEVERHardNegatives | 88.20 | 88.25 | 87.30 | 89.40 | 87.15 |
| FiQA2018 | 43.17 | 43.64 | 45.25 | 42.79 | 45.27 |
| HotpotQAHardNegatives | 72.67 | 72.90 | 73.54 | 73.60 | 73.37 |
| ImdbClassification | 94.64 | 94.90 | 94.57 | 94.62 | 94.43 |
| MTOPDomainClassification | 95.86 | 96.02 | 96.39 | 95.84 | 96.34 |
| MassiveIntentClassification | 73.57 | 73.78 | 74.87 | 73.13 | 75.38 |
| MassiveScenarioClassification | 75.77 | 75.95 | 76.99 | 75.97 | 77.37 |
| MedrxivClusteringP2P.v2 | 45.25 | 45.70 | 44.53 | 45.60 | 43.59 |
| MedrxivClusteringS2S.v2 | 42.42 | 43.28 | 42.31 | 42.96 | 41.51 |
| MindSmallReranking | 31.54 | 30.41 | 32.01 | 31.63 | 31.69 |
| SCIDOCS | 13.39 | 11.63 | 16.15 | 10.27 | 17.40 |
| SICK-R | 78.89 | 76.85 | 79.61 | 78.50 | 79.59 |
| STS12 | 72.93 | 72.25 | 74.68 | 73.02 | 75.62 |
| STS13 | 78.69 | 73.27 | 80.26 | 78.06 | 81.20 |
| STS14 | 74.63 | 71.11 | 75.83 | 74.21 | 76.52 |
| STS15 | 84.31 | 82.88 | 85.60 | 84.20 | 86.37 |
| STS17 | 88.46 | 88.00 | 89.37 | 88.52 | 89.39 |
| STS22.v2 | 63.93 | 61.44 | 64.86 | 62.75 | 64.48 |
| STSBenchmark | 81.13 | 78.71 | 82.85 | 81.23 | 83.59 |
| SprintDuplicateQuestions | 90.48 | 85.24 | 91.15 | 89.00 | 93.41 |
| StackExchangeClustering.v2 | 72.10 | 71.76 | 72.80 | 71.01 | 72.54 |
| StackExchangeClusteringP2P.v2 | 43.87 | 44.31 | 43.63 | 45.27 | 43.91 |
| SummEvalSummarization.v2 | 31.70 | 34.29 | 33.99 | 29.47 | 32.51 |
| TRECCOVID | 61.10 | 63.17 | 66.30 | 66.98 | 67.22 |
| Touche2020Retrieval.v3 | 51.13 | 54.01 | 54.96 | 53.70 | 56.34 |
| ToxicConversationsClassification | 89.75 | 90.52 | 89.66 | 89.74 | 88.24 |
| TweetSentimentExtractionClassification | 78.08 | 78.32 | 77.37 | 77.93 | 76.64 |
| TwentyNewsgroupsClustering.v2 | 59.94 | 61.13 | 61.84 | 60.87 | 61.95 |
| TwitterSemEval2015 | 67.63 | 65.76 | 69.08 | 66.77 | 69.33 |
| TwitterURLCorpus | 85.47 | 84.79 | 85.95 | 85.26 | 86.23 |
| AppsRetrieval | 16.72 | 32.92 | 33.08 | 44.35 | 48.85 |
| COIRCodeSearchNetRetrieval | 56.75 | 74.26 | 72.13 | 83.08 | 80.94 |
| CodeEditSearchRetrieval | 57.18 | 57.25 | 62.38 | 53.57 | 63.12 |
| CodeFeedbackMT | 52.33 | 62.32 | 64.02 | 90.23 | 86.09 |
| CodeFeedbackST | 74.60 | 76.51 | 80.76 | 86.59 | 85.14 |
| CodeSearchNetCCRetrieval | 55.56 | 63.66 | 67.71 | 87.64 | 79.25 |
| CodeSearchNetRetrieval | 81.80 | 88.52 | 88.02 | 89.71 | 89.53 |
| CodeTransOceanContest | 73.88 | 80.34 | 82.30 | 84.31 | 85.45 |
| CodeTransOceanDL | 32.99 | 35.34 | 34.67 | 32.18 | 35.20 |
| CosQA | 30.37 | 26.60 | 32.68 | 35.12 | 35.04 |
| StackOverflowQA | 80.74 | 83.15 | 87.26 | 89.20 | 90.44 |
| SyntheticText2SQL | 60.48 | 66.60 | 68.50 | 76.52 | 74.42 |
| AILACasedocs | 29.86 | 24.15 | 35.73 | 28.88 | 35.93 |
| AILAStatutes | 21.93 | 19.82 | 21.73 | 21.19 | 23.04 |
| AppsRetrieval | 16.72 | 32.92 | 33.08 | 44.35 | 48.85 |
| CUREv1 | 39.10 | 43.62 | 46.15 | 43.84 | 47.86 |
| ChatDoctorRetrieval | 47.58 | 53.30 | 55.05 | 53.56 | 58.41 |
| DS1000Retrieval | 47.70 | 57.94 | 58.15 | 61.99 | 61.71 |
| FinQARetrieval | 49.24 | 48.24 | 53.05 | 47.24 | 52.14 |
| FinanceBenchRetrieval | 67.38 | 67.66 | 72.65 | 68.08 | 74.55 |
| FreshStackRetrieval | 28.27 | 22.04 | 32.21 | 24.33 | 33.09 |
| HC3FinanceRetrieval | 55.54 | 56.31 | 55.94 | 55.68 | 56.07 |
| HumanEvalRetrieval | 84.39 | 90.13 | 91.58 | 85.22 | 95.05 |
| LegalQuAD | 13.22 | 36.45 | 40.22 | 29.17 | 45.53 |
| LegalSummarization | 58.77 | 60.95 | 63.21 | 58.37 | 63.13 |
| MBPPRetrieval | 83.82 | 87.23 | 88.55 | 87.92 | 89.32 |
| MIRACLRetrievalHardNegatives | 25.14 | 56.00 | 48.02 | 55.14 | 58.12 |
| WikiSQLRetrieval | 77.12 | 83.94 | 83.61 | 90.37 | 91.89 |
| ALL | 59.75 | 62.01 | 63.88 | 64.14 | 65.91 |