Improving General Text Embedding Model: Tackling Task Conflict and Data Imbalance through Model Merging

Mingxin Li[1], **Zhijie Nie**[1], **Yanzhao Zhang, Dingkun Long, Richong Zhang**[1], **Pengjun Xie**
[1]CCSE, School of Computer Science and Engineering, Beihang University
{limx,niezj,zhangrc}@act.buaa.edu.cn

Abstract

Text embeddings are vital for tasks such as text retrieval and semantic textual similarity (STS). Recently, the advent of pretrained language models, along with unified benchmarks like the Massive Text Embedding Benchmark (MTEB), has facilitated the development of versatile general-purpose text embedding models. Advanced embedding models are typically developed using large-scale multi-task data and joint training across multiple tasks. However, our experimental analysis reveals two significant drawbacks of joint training: 1) **Task Conflict**: Gradients from different tasks interfere with each other, leading to negative transfer. 2) **Data Imbalance**: Disproportionate data distribution introduces biases that negatively impact performance across tasks. To overcome these challenges, we explore model merging—a technique that combines independently trained models to mitigate gradient conflicts and balance data distribution. We introduce a novel method, **Self Positioning**, which efficiently searches for optimal model combinations within the interpolation space of task vectors using stochastic gradient descent. Our experiments demonstrate that Self Positioning significantly enhances multi-task performance on the MTEB dataset, achieving an absolute improvement of 0.7 points. It outperforms traditional resampling methods while reducing computational costs. This work offers a robust approach to building generalized text embedding models with superior performance across diverse embedding-related tasks.

1 Introduction

Text embedding is a crucial aspect in the fields of Information Retrieval (IR) and Natural Language Processing (NLP). Embedding models represent text in high-dimensional spaces to capture semantic information, making them applicable to various downstream tasks, including text retrieval [1], semantic textual similarity (STS) [2], and text classification [3], among others. Initially, text embedding models for different tasks were mostly trained independently. Recently, with the emergence of pretrained language models, such as BERT [4] and GPT3 [5], and unified multi-task evaluation benchmarks like the Massive Text Embedding Benchmark (MTEB) [6], there has been significant progress in training general text embedding models [7–10]. These general text embeddings offer greater usability and broader applicability compared to models previously trained for specific tasks.

Recently, State-of-the-Art general text embedding models predominately adopt a dual-tower training architecture [1]. This architecture encodes the query and document of a text pair into continuous vectors using pretrained language models, while the training objective employs contrastive learning. To ensure that the resulting embedding models possess strong domain generalization and multi-task adaptability, training typically involves collecting large-scale, multi-task relevant text pairs (often exceeding millions in quantity). Additionally, some approaches gather billions of weakly supervised text pairs from public web resources (e.g., query-answer pairs from Q&A communities) to enhance

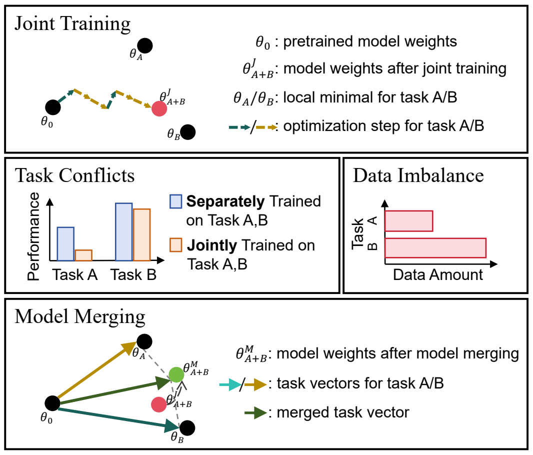

Joint Training
θ_0 : pretrained model weights
θ^J_{A+B}: model weights after joint training
θ_A/θ_B: local minimal for task A/B
optimization step for task A/B
Task Conflicts
Performance
Task A
Task B
Separately Trained on Task A,B
Jointly Trained on Task A,B
Data Imbalance
Task
A
B
Data Amount
Model Merging
θ^M_{A+B}: model weights after model merging
: task vectors for task A/B
: merged task vector

Figure 1: During joint training across multiple tasks, model weights fluctuate towards local minima for each task, which leads to two primary issues: 1) Gradient conflicts reduce performance; 2) Data imbalances result in unequal optimization frequencies, biasing the final weights towards more prevalent tasks. Model merging mitigates these challenges by isolating task training and achieving a more balanced weight position.

domain generalization and robustness. Despite the effectiveness of this mixed multi-task contrastive learning approach, we have identified two significant issues that can impact model performance. The first is **Task Conflict**. Previous study [11] on multi-task learning suggests that when concurrently training on multiple tasks, gradients from different tasks may conflict, leading to the Negative Transfer Phenomenon, where a jointly trained model performs worse than models trained separately on each task. The second issue is **Data Imbalance**. Significant differences in the data amount collected from various sources (e.g., a ratio of 50:1 between Retrieval and Classification tasks) can create inductive bias, severely impacting the modelś performance across different tasks.

In this work, we examine a simplified yet representative multi-task joint training setting involving text retrieval and semantic textual similarity (STS) tasks. Our empirical analysis confirms that task conflict arises in the commonly used joint training approach, leading to poorer performance in jointly trained models compared to those trained separately on individual tasks. Furthermore, we utilize task vectors, as described by [12], to analyze the impact of data imbalance on model performance. Task vectors assess the shift in model weights across tasks, helping us understand performance disparities. Our findings indicate that the amount of training data significantly influences the positioning of task vectors (see Figure 2), thereby affecting model performance on different tasks.

To address the limitations of directly joint-training general text embedding models while ensuring their generalization and robustness, we explore model merging, a technique gaining popularity in multi-task learning scenarios. This method combines parameters from different models, allowing the unified model to handle multiple tasks without the gradient conflicts and data imbalance issues inherent in joint training. We evaluate five common model merging strategies in the context of general text embedding. The results show significant improvements in multi-task performance. We also analyze the relationship between merged models and the interpolation space of task vectors, focusing on how the direction and norm of merged models influence performance. Our findings indicate that ***searching within the interpolation space of task vectors is sufficient to identify the optimal model merge strategy***.

In model merging method, each sub-model is trained independently, and the hyperparameters increase linearly with the number of sub-models, complicating parameter searches. We find that the performance of merged models correlates with their loss on multi-task training data. Therefore, we propose an efficient method called **Self Positioning** to search for optimal merging positions within the interpolation space of task vectors. This technique uses stochastic gradient descent to find low-loss hyperparameter combinations for a specified target dataset. Our experiments confirm that **Self Positioning** identifies high-performance merging results effectively. Compared to resampling methods for handling data imbalance, Self Positioning achieves better performance with significantly less time cost. To fully validate the effectiveness of Self Positioning method, we apply Self Positioning to a process involving 330 tasks collected by Instructor [13]. Experiments demonstrate that our model merging strategy significantly improved performance on the MTEB dataset (0.7 point increasement in average performance).

Briefly, our main contributions are as follows: 1) We identified two key problems affecting the performance of general text embeddings in joint training: **Task Conflict** and **Data Imbalance**, through detailed experimentation and analysis. 2) We explore using model merging to address these limitations. Additionally, we discovered that high-performing merged models can be found within the interpolation space of task vectors. This inspired us to propose a **Self Positioning** approach as an automatic search method for model merging. 3) We conduct extensive experiments and analysis to validate the effectiveness of Self Positioning, significantly enhancing the performance of general text embedding models.

2 Background

2.1 General Text Embeddings

Text embeddings have shown significant potential in applications such as web retrieval, text similarity, and text classification. Initially, research focused on creating task-specific embeddings. However, with the emergence of pretrained language models, the emphasis has shifted toward developing general text embedding models capable of handling various tasks. Studies like GTR [14], E5 [8], GTE [15], BGE [16], and JinaEmbedding [17] have fine-tuned embedding models using large amounts of multi-task relevance data. Additionally, there are studies like TART [18] and InstructOR [13] that introduce natural language prompts to guide embedding models in generating task-specific outputs. Leveraging the powerful text modeling capabilities of pretrained Large Language Models (LLMs), models like SGPT [19], UDVER [20], E5Mistral [21], RepLLaMA [22], GTE-Qwen [15], and NV-Embed [23] utilize LLMs as their foundations, refining them with multi-task data and instructional guidance.

To enhance the performance of general text embedding models, recent research primarily focuses on optimizing training data and pretrained language models. In terms of training data, efforts concentrate on collecting large-scale, high-quality, multi-domain, and multi-task relevant text pairs from public benchmarks or web resources [14, 8, 15, 16]. Synthetic data is also generated using large language models to expand the dataset further [21]. For pretrained models, researchers aim to adapt models designed for general natural language understanding (NLU) or generation tasks to better suit embedding tasks [24, 25]. During model training, most approaches employ contrastive learning as the optimization objective [26]. Consider the training data $\mathcal{D}$ consists of N datasets: $\mathcal{D} = \{D_i\}_{i=1}^{N}$, each containing N_i instances: $D_i = \{I_j\}_{j=1}^{N_i}$. Each instance I_j includes a query text q, a similar text p, and a set of dissimilar texts $P^{\mathrm{n}} = \{p_k^{\mathrm{n}}\}_{k=1}^{K}$. A model M with parameters θ maps each text to an L2-normalized vector $M(p)$. For each instance I_j, we optimize the following loss function:

$$\mathcal{L}_{I_j}^{\mathrm{CL}} = -\log \frac{\exp(M(p)^\top M(q)/\tau)}{\exp(M(p)^\top M(q)/\tau) + \sum_{k=1}^{K} \exp(M(p)^\top M(q_k^{\mathrm{n}})/\tau)},$$

where τ is a temperature parameter.

Unlike previous studies, this work focuses on the training process of models, particularly identifying and addressing the limitations of joint training with multi-task large-scale datasets to enhance the performance of the final model.

2.2 Task Vectors

The concept of task vectors was introduced by [12]. Given an initial model M_0 with weights θ_0, and a trained model M_i with weights θ_i after learning on a specific task i, the task vector for the corresponding task is defined as:

$$V_i = \theta_i - \theta_0. \tag{1}$$

[12] discovered that performing arithmetic operations on task vectors directly affects the model's performance on the corresponding tasks. For instance, adding or subtracting the task vector V_i to the weights θ_j of another model M_j will respectively increase or decrease M_j's performance on task i. Therefore, task vectors effectively represent the capabilities a model has acquired for specific tasks.

This study investigates how the norm and direction of task vectors impact task performance. The norm of the task vector, $\|V_i\|$, measures its length in the weight space. The direction of the task vector pertains to the relative orientation between the combined task vector V_{i+j}, which can perform tasks i and j simultaneously, and the individual task vectors V_i and V_j, which can perform tasks i and j independently. Specifically, we define direction as:

$$\frac{\alpha_{(i+j,i)}}{\alpha_{(i+j,i)} + \alpha_{(i+j,j)}}, \tag{2}$$

V_{i+j} = V_{i} + V_{j}

where $\alpha_{(i+j,i)}$ is the angle between V_{i+j} and V_i, and $\alpha_{(i+j,j)}$ is the angle between V_{i+j} and V_j (see Figure 2c for illustration). Equation 2 quantifies the extent to which the task vector V_{i+j} aligns more closely with V_i relative to V_j.

2.3 Model Merging

Model merging [27–30] refers to the process of integrating the parameters of multiple models with the same architecture into a single unified model, enabling the merged model to possess the capabilities of multiple models. Model merging has been applied to many machine learning subfields, such as continual learning to mitigate catastrophic forgetting of old tasks, and multi-task/multi-domain learning to facilitate knowledge transfer [30].

This work involves five popular model merging methods. Specifically, given N datasets $\{D_i\}_{i=1}^N$ and the N models $\{M_i\}_{i=1}^N$ trained on them, with corresponding model weights $\{\theta_i\}_{i=1}^N$ and task vectors $\{V_i\}_{i=1}^N$, the following five methods are used to obtain the merged task vector V_{m}: (1) **Average** Merging [27] performs linear interpolation between different task vectors to obtain V_{m}:

$$V_{\mathrm{m}} = \sum_{i=1}^{N} w_i V_i \Big/ \sum_{i=1}^{N} w_i \ , \tag{3}$$

where $\{w_i\}_{i=1}^N$ ($w_i \in \mathbb{R}$) are hyperparameters; (2) **SLERP** [1] performs **S**pherical **L**inear int**ERP**olation successively between two task vectors for $(N-1)$ times to obtain V_{m}:

$$V_{i+j} = \frac{\sin\left(\frac{a_j}{a_i+a_j}\alpha_{(i,j)}\right) V_i + \sin\left(\frac{a_i}{a_i+a_j}\alpha_{(i,j)}\right) V_j}{\sin(\alpha_{(i,j)})}, \tag{4}$$

where $a_i, a_i \in \mathbb{R}$ are hyperparameters; (3) **TIES** Merging tackles the task conflicts in task vectors by trimming low-magnitude model weights, resolving sign disagreements, and disjointly merging parameters with consistent signs to obtain V^{TIES}, which is scaled by hyperparameter λ to obtain V_{m}:

$$V_{\mathrm{m}} = \lambda V^{\mathrm{TIES}}; \tag{5}$$

(4) **Fisher** Merging [31] calculates merging coefficient $\hat{F}_{\theta_i} \in \mathbb{R}^{\|\theta_0\|}$ based on Fisher information matrix:

$$\hat{F}_{\theta_i} = \frac{1}{N_i} \sum_{j=1}^{N_i} (\nabla_{\theta_i} \mathcal{L}^{\mathrm{CL}}(\theta_i; I_j))^2, \tag{6}$$

where $I_j \in D_i$ is a training sample. $\hat{F}_{\theta_i}$ is then used to obtain Vm:

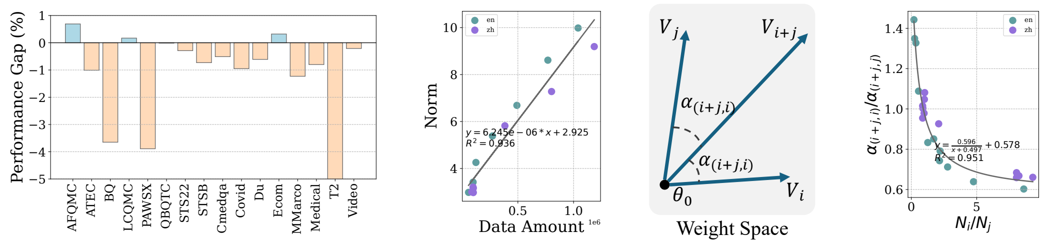
$$V_{\mathrm{m}}^{(k)} = \frac{\sum_{i=1}^{N} \lambda_i F_{\theta_i}^{(k)} V_i^{(k)}}{\sum_{i=1}^{N} \lambda_i F_{\theta_i}^{(k)}}, \tag{7}$$

where $\{\lambda_i\}_{i=1}^N$ ($\lambda_i \in \mathbb{R}$) are hyperparameters, and $k = 1, \cdots, \|\theta_0\|$; (5) **RegMean** Merging [32] performs model merging by optimizing a linear regression problem with the following closed-form solutions:

$$W_{\mathrm{m}}^k = (\sum_{i=1}^{N} X_i^k (X_i^k)^\top)^{-1} \sum_{i=1}^{N} (X_i^k (X_i^k)^\top W_i^k) \tag{8}$$

to obtain V_{m}, where $\{W_i^k\}_{k=1}^K$ are linear layers weights in θ_i, each corresponding to an input matrix $\{X_i^k\}_{k=1}^K$. We implement these methods based on the implementations from [29].

[1] `https://github.com/Digitous/LLM-SLERP-Merge`



(a) The difference in model performance obtained from joint training compared to the best performance achieved when trained separately on STS and Retrieval tasks.

(b) The Impact of Data Amount on Norm

(c) Subjects of statistics

(d) The Impact of Data Amount on Direction

Figure 2: Experimental Results on Task Conflict and Data Imbalance Problems.

Table 1: The number of instances in training datasets.

	STS	Retrieval			
en	ALLNLI 271,612	FEVER 123,140	HotpotQA 97,825	MS MARCO 491,007	NQ 57,110
zh	SimCLUE 802,291	DuReader 86,395	Ecom 99,763	Medical 97,509	Video 99,719

3 Problems Analysis

3.1 Training and Evaluation Setup

To investigate the problem of **task conflict** and **data imbalance** in the joint-training process of embedding models, we first conduct experiments in a controlled setting. Specifically, we utilize ten datasets for training across two tasks: Semantic Textual Similarity (STS) and Retrieval, in both English and Chinese languages. These two tasks were selected due to their widespread application in real-world scenarios and their differing data requirements—STS involves matching symmetric texts, while Retrieval often requires matching asymmetric texts. This selection naturally highlights potential conflicts between the tasks.

Dataset For the English STS task, we use the AllNLI [2] dataset, and the Retrieval task incorporates four datasets: FEVER [33], HotpotQA [34], MS MARCO [35], and NQ [36]. In the Chinese STS task, we utilize the SimCLUE dataset[2]. The Retrieval task for Chinese includes DuReaderRetrieval [37], Ecom/Medical/Video-Retrieval [38]. The size of each dataset is detailed in Table 1.

Training Details We train the $\text{BERT}_{\text{base}}$ model [4] separately for English and Chinese using the aforementioned datasets. Following the training approach proposed by [15], each batch contains data from a single task. The hyperparameters for all experiments are set as follows: 3 epochs, a batch size of 256, a learning rate of 1×10^{-5}, a temperature τ of 2×10^{-2}, and a warmup ratio of 0.1.

Evaluation The model's performance is evaluated using the STS and Retrieval evaluation tasks from MTEB [6], which include 25 tasks in English and 16 tasks in Chinese.

3.2 Task Conflict

Research on multi-task learning has shown that jointly training multiple tasks can lead to the *Negative Transfer Phenomenon* [39], where the model's performance deteriorates compared to models trained separately on each task. We observe a similar effect in embedding model training. Specifically, we trained models separately on English and Chinese STS and Retrieval datasets and then evaluated

[2]https://github.com/CLUEbenchmark/SimCLUE

their performance on each task, as illustrated in Figure 2a. The results demonstrate that, in the Chinese setting, the jointly trained model underperforms in 13 out of 16 evaluation tasks compared to task-specific models. Similarly, in the English setting, performance declines in 12 of 25 tasks (see Figure 9 in the appendix). Previous studies [11] suggest that this may be due to detrimental gradient interference between tasks, where optimizing for one task conflicts with the gradients of others. This observation motivates the solutions we propose in Section 4.

3.3 Data Imbalance

As shown in Table 1, the sizes of different datasets can vary by nearly a factor of 10. However, current methods do not account for this imbalance and instead, combine all datasets in their entirety for training. To thoroughly understand the issue of data imbalance, we examine how differences in training data size affect the performance of embedding models in this section.

3.3.1 Impact on Norm

First, we explore the relationship between the amount of training data and the norm of the task vector in the trained model. We train the model on individual datasets (e.g., DuReaderRetrieval training data), on single-task datasets (e.g., all Retrieval training data), and on all training datasets combined. We then analyze the relationship between the norm of the task vectors ($\|V_i\|$) and the amount of training data (N_i). The results, shown in Figure 2b, reveal a clear linear relationship, indicating that the norm of the task vectors increases proportionally with the amount of training data.

3.3.2 Impact on Direction

Next, we investigate the relationship between the amount of training data and the direction of the task vectors. For a pair of training datasets (D_i, D_j) with respective data sizes N_i and N_j, we perform the following steps: 1) Train jointly on both datasets to obtain model M_{i+j} and its task vector V_{i+j}. 2) Train separately on each dataset to obtain models M_i and M_j, with task vectors V_i and V_j, respectively. 3) Measure the angles $\alpha_{(i+j,i)}$ between V_{i+j} and V_i, and $\alpha_{(i+j,j)}$ between V_{i+j} and V_j. Figure 2c illustrates the experimental setup. The dataset pairs (D_i, D_j) include individual pairs (e.g., SimCLUE and DuReaderRetrieval) as well as a combination of all STS and Retrieval training data. To clearly demonstrate the relationship between the angle ratios $\frac{\alpha_{(i+j,i)}}{\alpha_{(i+j,j)}}$ and the data size ratios $\frac{N_i}{N_j}$, we plot these values in Figure 2d. The figure shows that as $\frac{N_i}{N_j}$ increases, the ratio $\frac{\alpha_{(i+j,i)}}{\alpha_{(i+j,j)}}$ decreases significantly. This indicates that the task vector obtained from joint training becomes more aligned with the task vector of the dataset that has a larger amount of training data.

These experiments reveal that differences in dataset sizes introduce varying inductive biases, significantly affecting both the norm and direction of the task vectors. This suggests that the properties of the task vectors are heavily influenced by the amount of training data, serving as an inductive bias that does not directly correlate with the model's performance. Consequently, it is possible that task vectors with better performance exist at different positions.

4 Solutions via Model Merging

According to the experiments in Section 3, joint training presents two main challenges: 1) Negative Transfer: Gradient conflicts between different tasks during optimization; 2) Data Imbalance: The amount of training data significantly affects the positioning of task vectors. We propose that model merging can address these issues effectively because: 1) Independent Training: Model merging ensures that different tasks are trained independently, thereby avoiding gradient conflicts during optimization. 2) Adjustable Task Vectors: Task vectors can be scaled and weighted during the merging process, which helps mitigate or eliminate the effects of data imbalance. In this section, we investigate the effectiveness of model merging in training embeddings.

4.1 Model Merging Pipelines

Based on model merging, we come up with the two training pipelines, both of which have potential applications in training embedding models: (1) **Separate Merging**: Using distinct datasets $\{D_i\}_{i=1}^{N}$, we train separate models $\{M_i\}_{i=1}^{N}$ from a common backbone. These models are then merged to form the final model $M_{\mathrm{m}}^{\mathrm{sep}}$. This approach enables the transformation of an existing backbone (e.g., a language model) into an embedding model. (2) **Iterative Merging**: We iteratively train the backbone on the datasets $(D_1, \dots, D_N)$ to obtain a sequence of models $(M_1, M_{(1,2)}, \dots, M_{(1,\dots,N)})$. These models are subsequently merged to create the final model $M_{\mathrm{m}}^{\mathrm{iter}}$. This method is particularly effective for enhancing the performance of existing embedding models on specific tasks.

4.2 Effectiveness of Model Merging

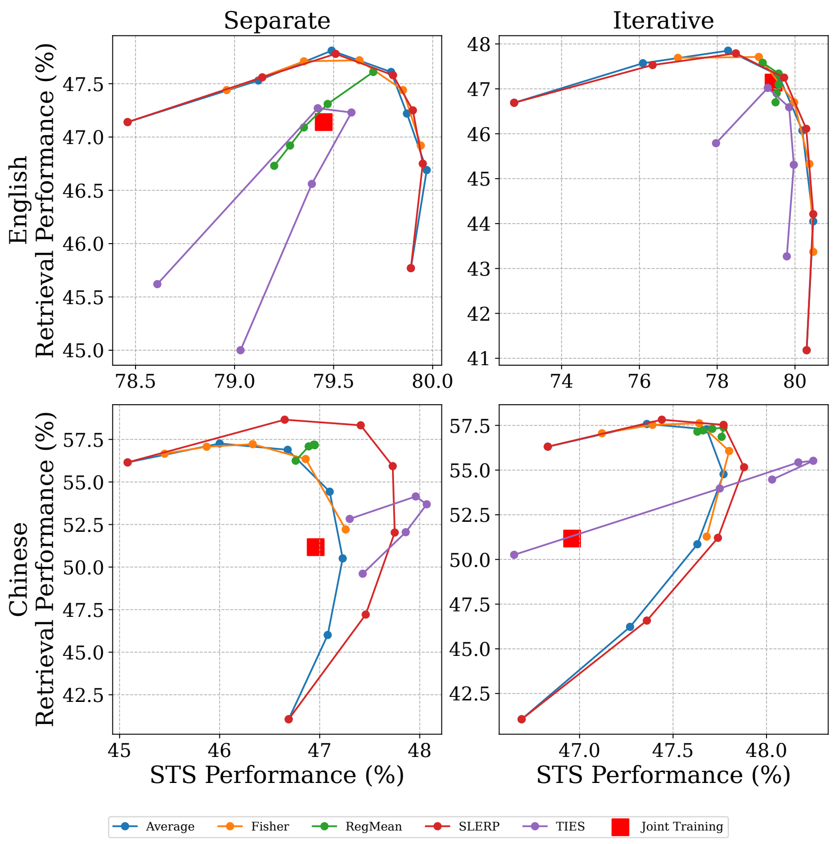


Figure 3: Performance of models under different languages, merging settings, and model merging methods during hyperparameter tuning.

To validate the training pipelines described in Section 4.1, we evaluate them using the model merging methods outlined in Section 2.3 within the simplified settings proposed in Section 3. Specifically, we use training data from STS and Retrieval to obtain the task vectors V_{s} and V_{r} for their respective tasks and then derive the merged task vector V_{m} through various model merging techniques. The training configurations align with those detailed in Section 3.1. For each model merging method, we optimize performance through hyperparameter tuning, with results presented in Figure 3 and Table 2. Additionally, we include results from joint training in the figure. In the figure, models positioned closer to the top right corner demonstrate stronger average multi-task performance. Our comparison reveals that, across both English and Chinese settings and the two training pipelines, model merging methods significantly outperform joint training in terms of average performance.

The improvement in average performance from model merging demonstrates its effectiveness in the application and highlights its capability to alleviate task conflict during training.

4.3 Key Factors in Model Merging

In this section, we explore the key factors influencing the merged performance. From the results of Section 4.2, it is evident that the performance improvements brought by different model merging methods vary, with SLERP performing well in all settings and having the highest upper-performance limit in three out of four settings. Therefore, we mainly study the SLERP method and use the Averaging method, whose merging results both lie in the interpolation space of task vectors, for comparison.

4.3.1 Direction of V_{m}

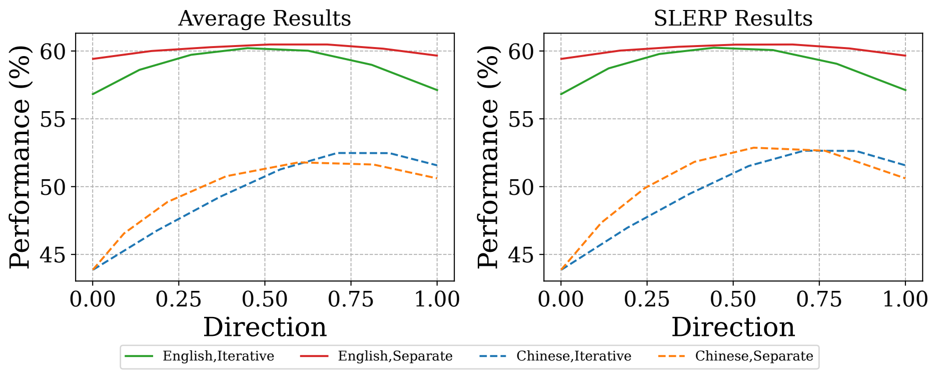


Figure 4: The relationship between the direction of task vectors and the average performance on STS and Retrieval tasks in different settings.

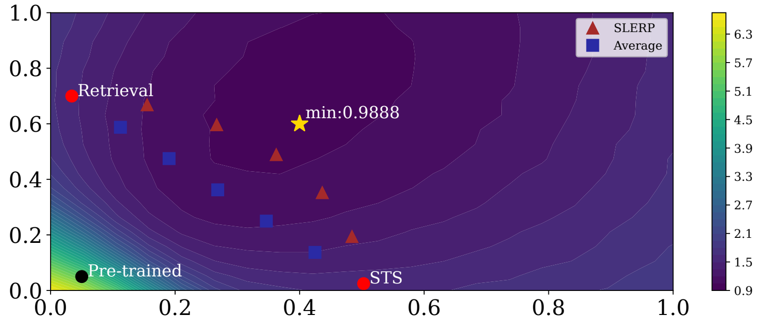


Figure 5: The loss landscape on the interpolation space of Retrieval and STS task vectors, under English Separate Merging. Other settings yield similar results.

We examine the link between model performance and the direction of task vectors using the Average and SLERP methods. The results are plotted in Figure 4, where we can observe that direction significantly impacts the performance of the task vector. Therefore, *direction is a key factor when seeking the optimal task vector*.

4.3.2 Norm of V_{m}

Though both Average and SLERP can explore many directions, there is a performance gap between them (see Figure 3). To investigate this gap, we plot the loss landscape under multi-task training data on the interpolation space $P_{(s,r)}$ in Figure 5 (refer to Appendix C for statistical details), and plot the merging results V_{m} from Average and SLERP methods in it. From the figure, we can see: 1) The norm of V_{m} from SLERP is consistently longer than that from Average; 2) V_{m} from SLERP consistently stays closer to regions with lower loss in the loss landscape. These findings reflect the impact of the norm of V_{m} on its properties (i.e., the loss on multi-task training data). Due to the differences in norm between the Average and SLERP methods leading to variations in performance, this indicates that *norm is a key factor when seeking the optimal task vector*.

5 Self Positioning in the Interpolation Space

As shown in Figure 3, the performance upper bound (The point closest to the top right corner) is sensitive to both the direction and norm of the task vector, which is only reached under a few merging weight settings. When the number of models is small (e.g., $N = 2$), different combinations of directions and norms can be explored using a grid search method to achieve high performance. However, as the number of parameters describing directions increases linearly with the number of models N (referring to the SLERP merging process), the time complexity of the grid search exponentially increases with the number of models N. This makes the method of finding the merging result $V\mathrm{m}$ through exhaustive search impractical. Therefore, in this section, we propose a method to find a near-optimal V_m within the interpolation space.

5.1 Methodology

Our approach is essentially a hyperparameter search method based on a probe dataset, therefore, it can be applied to any model merging method. Given SLERP's superior performance, we use the SLERP method for all equations and experiments. For brevity, we rewrite Equation 4 in functional form:

$$f^{\mathrm{SLERP}}(V_i, V_j, a_i, a_j) = \frac{\sin\left(\frac{a_j}{a_i+a_j}\alpha_{(i,j)}\right) V_i + \sin\left(\frac{a_i}{a_i+a_j}\alpha_{(i,j)}\right) V_j}{\sin(\alpha_{(i,j)})},$$

Given N task vectors $\{V_i\}_{i=1}^N$ and a specific model merging method, we try to control the direction and norm of the merged task vector using different learnable parameters.

Direction Control Corresponding merging weights $\{a_i\}_{i=1}^N$, we merge them in an iterative form:

$$V_{(1,\cdots,N)} = f^{\mathrm{SLERP}}\left(V_{(1,\cdots,N-1)}, V_N, \frac{\sum_{i=1}^{N-1} a_i}{N-1}, a_N\right), \tag{9}$$

Norm Control By applying a scaling factor λ, we can represents all positions in the interpolation space:

$$V_\mathrm{p} = \lambda V_{(1,\cdots,N)} \tag{10}$$

As our goal is to search for positions with low loss, we need to construct a target dataset D_t, and form the optimization problem:

$$(\{\hat{a_i}\}_{i=1}^N, \hat{\lambda}) = \underset{(\{a_i\}_{i=1}^N, \lambda)}{\arg\min} \left(\frac{1}{|D_\mathrm{t}|}\sum_{I \in D_\mathrm{t}} \mathcal{L}^{\mathrm{CL}}(I; \theta_0 + V_\mathrm{p}) + \mu\lambda\right) \tag{11}$$

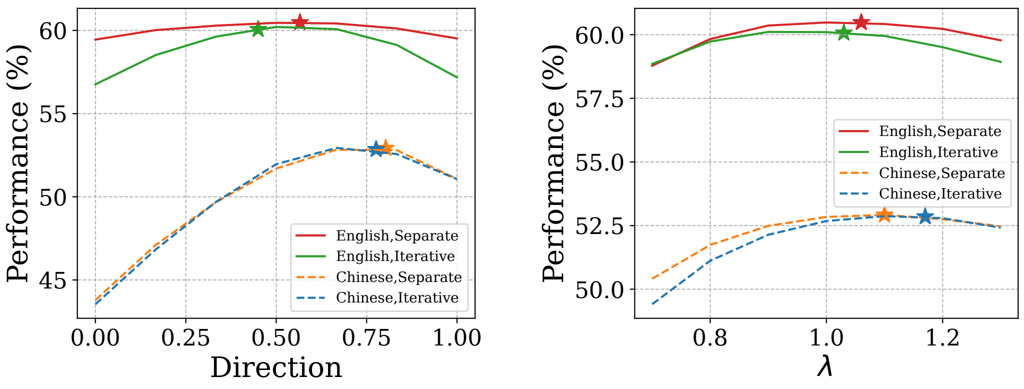
where θ_0 represents the init parameters of the embedding model, and μ is a hyperparameter used to prevent overfitting in this optimization problem. For the target dataset D_t being used, we note that: 1) Since the number of parameters to be optimized is small, the amount of data required for the target dataset is much less than that required to train the model; 2) The target dataset can be obtained either by sampling from the training dataset or by choosing to construct it from other data sources. Overall, we can find the final task vector by solving this optimization problem using stochastic gradient descent algorithm [40]. We refer to this method of finding the merging results within the interpolation space of task vectors as **Self Positioning**.

5.2 Experiments

We apply Self Positioning in the model merging settings described in Section 4.2, which includes four merging processes: separate merging and sequential merging for both English and Chinese. To construct the target dataset D_t, we sample data from the training datasets for the STS task and the Retrieval task in equal proportions, resulting in a final D_t that contains $32,000$ instances. The optimization process uses the following settings: the optimizer is `Adam` [41], the batch size is 32, the learning rate is 5×10^{-3}, and the number of training steps is $1,000$. The initial values for the weight parameters $\{a_i\}_{i=1}^N$ and the scaling factor λ are both set to 1. We conduct experiments under

Table 2: Average performance (%) on all STS and Retrieval tasks. "JT" represents results from joint training, "SP" represents results from Self Positioning, and other model merging methods report the best results after hyperparameter tuning.

	JT	Merging Pipeline	Fisher	RegMean	TIES	Average	SLERP	SP
en	60.1	Separate	**60.5**	60.4	60.2	**60.5**	**60.5**	**60.5**
		Iterative	**60.3**	60.2	59.9	60.2	60.2	60.1
zh	51.2	Separate	51.7	52.1	51	51.8	**52.9**	**52.9**
		Iterative	52.6	52.6	51.9	52.5	52.6	**52.9**



(a) Results when fixing λ

(b) Results when fixing direction

Figure 6: Experimental results for validating the effectiveness of the search results of Self Positioning.

$\mu \in \{0.00, 0.05, 0.10\}$, with the best results presented in Table 2. The results in the table show that Self Positioning achieved the best results in three of the four merging settings. Although it does not achieve the best results in the iterative merging for English, it performs comparably to joint training. It is important to note that the results of other model merging methods in the table are the best results obtained after hyperparameter tuning. Therefore, the performance comparison in the table demonstrates the effectiveness of Self Positioning in searching for merging results with good performance.

5.3 Performance Analysis

This section provides analyses of the merging results obtained through Self Positioning.

5.3.1 Effectiveness of the Search Results

To determine whether the search results are positioned in areas of high performance within the interpolation space, we examine the performance surrounding these results. Specifically, for a search result $(\{\hat{a}_1, \hat{a}_2\}, \hat{\lambda})$, we conducted two experiments: (1) Holding $\hat{\lambda}$ constant, we adjuste the ratio $\frac{a_1}{a_1+a_2}$ to take values in $\{\frac{0}{6}, \frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}, \frac{6}{6}\}$ and evaluate the performance of the merging results. (2) Keeping $\{\hat{a}_1, \hat{a}_2\}$ fixed, we vary λ across the values $\{0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$ and assessed the performance of the merging outcomes. The results of Experiment (1) & (2) are illustrated in Figure 6a and Figure 6b, respectively. Additionally, the results identified by Self Positioning $(\{\hat{a}_1, \hat{a}_2\}, \hat{\lambda})$ are highlighted. Our findings indicate that the proposed method successfully identifies optimal results in most scenarios. In instances such as the English Separate Merging setting, where the optimal result was not achieved, the merging result remained closely aligned with the optimal position. This proximity underscores the effectiveness of the search results derived from Self Positioning.

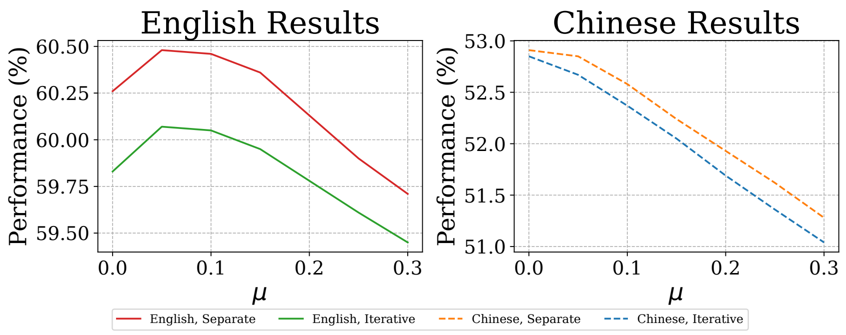
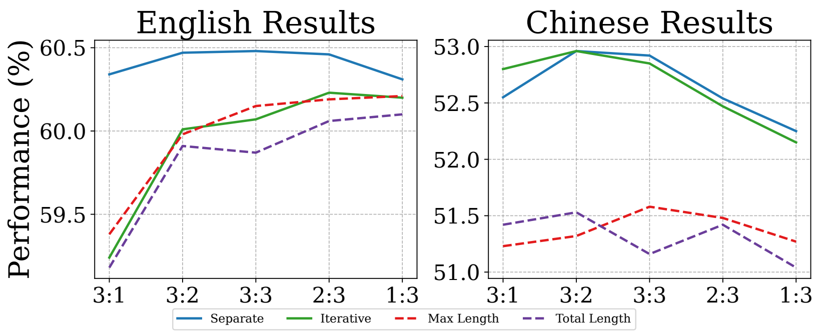
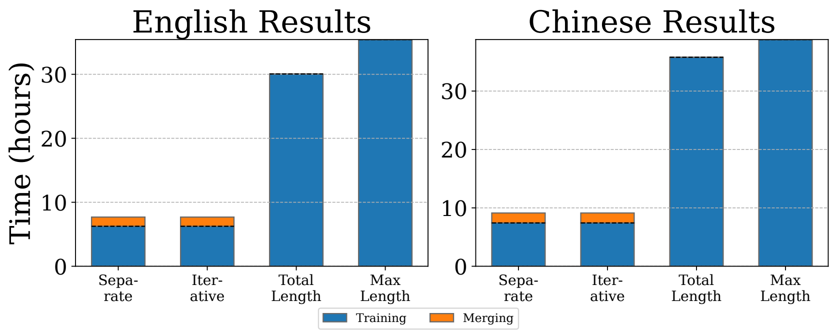


Figure 7: The impact of the hyperparameter μ on the performance of Self Positioning search results.



(a) Performance Comparison



(b) Time Cost Comparison

Figure 8: Comparison of Self Positioning (labeled as "Separate" and "Iterative") and Resampling Methods ("Total Length" and "Max Length") in Tackling Data Imbalance.

5.3.2 The Influence of μ

Equation 11 aims to prevent overfitting during optimization by constraining the scaling factor λ. In this section, we maintain the same settings as described in Section 5.2 and conduct experiments with $\mu \in \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$. Figure 7 visualizes the changes in the Self Positioning search results and model performance. Firstly, the changes in the search result positions indicate that μ significantly affects the magnitude but only slightly influences the direction. Secondly, the model performance results show that optimal performance is typically achieved when μ is set to a relatively low value (i.e., $\mu \leq 0.05$).

5.3.3 Efficiency in Addressing Data Imbalance

In constructing the target dataset $D_{\rm t}$, we aim to address the problem of data imbalance by sampling proportionally from both the STS and Retrieval training data. In this section, we compare the approaches of Self Positioning and resampling to tackle data imbalance, focusing on two main dimensions: model performance and time cost. Specifically, we experiment with the ratios of STS

Table 3: Results on MTEB [6] for Separate Merging(Section 6.1) and Iterative Merging(Sectuion 6.2). Retri., Pair., Class., Sum. refer to retrieval, pair classification, classification, and summarization, respectively. All models utilize the T5-large encoder [42] as their backbone. Results of GTR and Instructor are sourced from [13]. For methods requiring sample data (Fisher, RegMean, and our Self Positioning), the first result in each group uses training data, while the second uses alternative data sources.

	Avg. (56)	Class. (12)	Cluster. (11)	Pair. (3)	ReRank. (4)	Retri. (15)	STS (10)	Sum. (1)
GTR [14]	58.3	67.1	41.6	85.3	55.4	47.4	78.2	29.5
Instructor [13]	61.6	73.9	45.3	85.9	57.5	47.6	83.2	31.8
Separate Merging								
TIES	62.2	72.5	45.4	86.3	57.7	51.1	82.8	31.2
Fisher	62.1 61.7	72.3 72.2	45.2 44.6	86.4 86.7	57.5 56.9	51.0 50.2	82.7 82.4	30.9 31.6
RegMean	61.5 61.4	71.9 72.0	44.7 44.5	86.6 86.7	56.9 56.8	49.7 49.6	82.4 82.4	31.6 31.5
Self Positioning	62.3 62.2	72.6 72.6	45.6 45.4	86.0 86.3	58.0 57.7	51.0 51.1	82.9 82.7	31.4 31.3
Iterative Merging (with extra classification training data)								
TIES	62.8	78.3	44.8	85.1	57.7	49.6	82.2	31.0
Fisher	62.5	79.2	44.5	84.5	57.6	48.7	81.7	29.9
RegMean	62.7	78.8	44.8	84.7	57.4	49.2	82.1	30.4
Self Positioning	63.0	79.1	44.6	85.3	57.6	49.9	82.2	30.5

training data amount N_{s} to Retrieval training data amount N_{r}, where $\frac{N_{\mathrm{s}}}{N_{\mathrm{r}}} \in \{\frac{3}{1}, \frac{3}{2}, \frac{3}{3}, \frac{2}{3}, \frac{1}{3}\}$. For Self Positioning, we ensure the total data amount remains constant (at 32,000), and adjust the ratios when constructing the target dataset D_{t}, while keeping other settings consistent with Section 5.2. We apply Self Positioning in both two merging settings (denoted as 'Separate' and 'Iterative' respectively). For the resampling method, we consider two scenarios: 1) ensuring the total data amount remains constant (English at 1,040,694, Chinese at 1,185,677), denoted as 'Total Length'; 2) keeping the larger dataset's amount constant (English at 769,082, Chinese at 802,291), denoted as 'Max Length'. Other settings are maintained as in Section 3.1. The results are illustrated in Figure 8. The results show that the Self Positioning method significantly improves both performance and time efficiency.

6 Applications on Instructor

Instructor [13] is an instruction-following embedding model that has been jointly trained on 330 tasks. Our objective is to further validate the proposed merging method by integrating more models. In this section, we apply Self Positioning and the two merging pipelines introduced in Section 4.1 to Instructor.

6.1 Separate Merging

In this section, we employ the Separate Merging pipeline to enhance its embedding performance. As there are not all tasks present conflicts, we adopt a clustering approach similar to that in [43] to divide the tasks into N clusters. For each task cluster, we fine-tune a `gtr-t5-large` [14] model using the same settings as in [13]. Subsequently, we apply Self Positioning to explore the merging results. During the search process, we set the batch size to 16 and $\mu \in \{0.00, 0.05\}$, while keeping other settings consistent with those described in Section 5.2. To assess the effectiveness of our method beyond the training data, we consider two target dataset settings: 1) constructed by sampling from the Instructor's training data; 2) using the target dataset composed of STS and Retrieval data mentioned in Section 5.2. The merged models are evaluated with 56 evaluation tasks in MTEB [6] and the results of 3-cluster are presented in Table 3. Additional results for different cluster numbers and detailed implementation specifics are available in Appendix D. By comparing the average performance in the table, we can draw the following conclusions: 1) Separate Merging consistently yields significant improvements over the original Instructor model across various settings. GTR v.s. Instructor This highlights the superiority of Separate Merging compared to joint training. 2) Analysis of different data sources reveals that Fisher and RegMean rely solely on training data and exhibit substantial performance drops when applied to data from other sources. Conversely, Self Positioning maintains strong performance across diverse data sources, demonstrating its adaptability and flexibility.

6.2 Iterative Merging

This section attempts to use the Iterative Merging to specifically enhance the performance of the optimal model M^{o} in Section 6.1 on particular tasks. Specifically, we collect a training dataset for Classification tasks consisting of 209,375 instances (detailed in Appendix E) to fine-tune M^{o} under the same training setting as in Section 6.1, with the merged model noted as M^{c}. Then, we use the search settings from Section 6.1 to search for merging results of M^{o} and M^{c}. Notably, the target dataset D_{t} is obtained by sampling from Instructor's training dataset and the training dataset used in this section at a 2:1 ratio. The results of our proposed method are compared with other model merging methods in Table 3. By comparing the average and classification performance in the table, we can draw the following conclusions: 1) The use of additional classification data combined with Iterative Merging leads to significant improvements in both average and classification performance, demonstrating that Iterative Merging can enhance model performance on certain tasks while maintaining average performance. 2) Self Positioning achieves superior average performance compared to other model merging methods, reflecting the advantages of the Self Positioning method in enhancing performance.

7 Conclusion

In this work, we identify two challenges in jointly training general text embeddings that hinder their performance: **Task Conflict** and **Data Imbalance**. To address these issues, we propose a model merging-based solution and validate its effectiveness through experiments. Additionally, by analyzing five existing model merging methods used in our experiments, we discover that high-performing merging results can be achieved solely within the interpolation space of task vectors. To mitigate the high time complexity associated with searching for merging results in this space, we introduce an automatic search method called **Self Positioning**. Through comprehensive experiments and analysis, we demonstrate that Self Positioning effectively searches for optimal merging results and efficiently resolves Data Imbalance. Finally, by applying our proposed method to the Instructor model, we significantly enhance its performance, illustrating the applicability of our approach to mainstream methods.

References

[1] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics, 2020.

[2] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics, 2021.

[3] Liliane Soares da Costa, Italo Lopes Oliveira, and Renato Fileto. Text classification using embeddings: a survey. *Knowl. Inf. Syst.*, 65(7):2761–2803, 2023.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec

Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[6] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive Text Embedding Benchmark. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2006–2029. Association for Computational Linguistics, 2023.

[7] 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. Text and code embeddings by contrastive pre-training. *CoRR*, abs/2201.10005, 2022.

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

[9] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *CoRR*, abs/2402.03216, 2024.

[10] Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *CoRR*, abs/2407.19669, 2024.

[11] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient Surgery for Multi-Task Learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[12] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[13] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One Embedder, Any Task: Instruction-Finetuned Text Embeddings. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1102–1121. Association for Computational Linguistics, 2023.

[14] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. Large Dual Encoders Are Generalizable Retrievers. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 9844–9855. Association for Computational Linguistics, 2022.

[15] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards General Text Embeddings with Multi-stage Contrastive Learning. *CoRR*, abs/2308.03281, 2023. arXiv: 2308.03281.

[16] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-Pack: Packaged Resources To Advance General Chinese Embedding. *CoRR*, abs/2309.07597, 2023. arXiv: 2309.07597.

[17] Michael Günther, Louis Milliken, Jonathan Geuter, Georgios Mastrapas, Bo Wang, and Han Xiao. Jina embeddings: A novel set of high-performance sentence embedding models. *arXiv preprint arXiv:2307.11224*, 2023.

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

[19] Niklas Muennighoff. Sgpt: Gpt sentence embeddings for semantic search, 2022.

[20] Xin Zhang, Zehan Li, Yanzhao Zhang, Dingkun Long, Pengjun Xie, Meishan Zhang, and Min Zhang. Language models are universal embedders. *ArXiv*, abs/2310.08232, 2023.

[21] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving Text Embeddings with Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[22] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 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, 2024.

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

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

[25] Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. Llama2Vec: Unsupervised adaptation of large language models for dense retrieval. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3490–3500, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[26] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *CoRR*, abs/1807.03748, 2018. arXiv: 1807.03748.

[27] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 2022.

[28] Shitao Xiao, Zheng Liu, Peitian Zhang, and Xingrun Xing. LM-Cocktail: Resilient Tuning of Language Models via Model Merging. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 2474–2488, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

[29] Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[30] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications and Opportunities, August 2024.

[31] Michael Matena and Colin Raffel. Merging Models with Fisher-Weighted Averaging. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[32] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless Knowledge Fusion by Merging Weights of Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[33] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics, 2018.

[34] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics, 2018.

[35] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[36] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: a Benchmark for Question Answering Research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019.

[37] Yifu Qiu, Hongyu Li, Yingqi Qu, Ying Chen, Qiaoqiao She, Jing Liu, Hua Wu, and Haifeng Wang. DuReader-Retrieval: A Large-scale Chinese Benchmark for Passage Retrieval from Web Search Engine. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5326–5338. Association for Computational Linguistics, 2022.

[38] Dingkun Long, Qiong Gao, Kuan Zou, Guangwei Xu, Pengjun Xie, Ruijie Guo, Jian Xu, Guanjun Jiang, Luxi Xing, and Ping Yang. Multi-CPR: A Multi Domain Chinese Dataset for Passage Retrieval. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai, editors, *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pages 3046–3056. ACM, 2022.

[39] Yu Zhang and Qiang Yang. A Survey on Multi-Task Learning. *IEEE Trans. Knowl. Data Eng.*, 34(12):5586–5609, 2022.

[40] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3981–3989, 2016.

[41] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[42] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[43] Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T. Kwok, and Yu Zhang. Mixture of Cluster-conditional LoRA Experts for Vision-language Instruction Tuning. *CoRR*, abs/2312.12379, 2023. arXiv: 2312.12379.

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas

Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.

[45] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics, 2020.

[46] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[47] Julian J. McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In Qiang Yang, Irwin King, Qing Li, Pearl Pu, and George Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 165–172. ACM, 2013.

[48] James O'Neill, Polina Rozenshtein, Ryuichi Kiryo, Motoko Kubota, and Danushka Bollegala. I wish I would have loved this one, but I didn't - A multilingual dataset for counterfactual detection in product review. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7092–7108. Association for Computational Linguistics, 2021.

[49] Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. Efficient intent detection with dual sentence encoders. *CoRR*, abs/2003.04807, 2020.

[50] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: contextualized affect representations for emotion recognition. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3687–3697. Association for Computational Linguistics, 2018.

[51] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics, 2011.

[52] Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 2950–2962. Association for Computational Linguistics, 2021.

A Experimental Setup

A.1 Experimental Environment

Our experimental environment consists of: `python` 3.10.14, `pytorch` 2.3.0 [44], `transformers` [45] 4.37.2, `mteb` [6] 1.2.0. Experiments with BERT$_{\text{base}}$ [4] are conducted on a single A100 GPU, while experiments on Instructor use two A100 GPUs.

A.2 Details of Evaluation

Our evaluations utilize MTEB benchmark [6], which consists of 7 tasks and a total of 56 datasets. The description of each task and its main evaluation matric are as follows:

- **Classification (12 datasets):** Embedding sentences for tasks like sentiment analysis, intent classification. *Main metric: Accuracy.*
- **Clustering (11 datasets):** Clustering sentences or paragraphs using embedded representations. *Main metric: V-measure.*
- **Pair Classification (3 datasets):** Classifying relationships between pairs of texts, such as determining if they are duplicates or paraphrases. *Main metric: Average Precision.*
- **Reranking (4 datasets):** Reranking lists of relevant and irrelevant texts in relation to a query based on embedded similarities. *Main metric: MAP (Mean Average Precision).*
- **Retrieval (15 datasets):** Retrieving relevant documents for queries from a corpus using embedded representations. *Main metric: nDCG@10 (Normalized Discounted Cumulative Gain).*
- **Semantic Textual Similarity (STS) (10 datasets):** Determining the similarity between pairs of sentences. *Main metric: Spearman Correlation.*
- **Summarization (1 dataset:** Evaluating the quality of machine-generated summaries compared to human-generated references. *Main metric: Spearman Correlation.*

A.3 Hyperparameter Tuning for Model Merging Methods

In Section 4.2, we conducted a hyperparameter search for each model merging method. Here we present the range of hyperparameter searches for each method:

- Average: $(w_{\text{s}}, w_{\text{r}}) \in \{(1,5),(2,4),(3,3),(4,2),(5,1)\}$;
- SLERP: $(a_{\text{s}}, a_{\text{r}}) \in \{(1,5),(2,4),(3,3),(4,2),(5,1)\}$;
- TIES: $\lambda \in \{0.8, 1.0, 1.2, 1.4, 1.6\}$;
- Fisher: $(\lambda_{\text{s}}, \lambda_{\text{r}}) \in \{(1,5),(2,4),(3,3),(4,2),(5,1)\}$;
- RegMean: The ratio for reducing non-diagonal elements $\sigma \in \{0.0, 0.1, 0.2, 0.3, 0.4\}$.

B Task Conflict

In the main text, we present the results of task conflict in Chinese; here, we provide the results of task conflict in English in Figure 9. From these results, it is evident that although the Negative Transfer Phenomenon is less pronounced in English than in Chinese, there is still a decline in performance in 12 out of the 25 tasks (nearly half).

C Loss Landscape

When computing the loss landscape, we first measure the angle α between the task vectors V_{s} and V_{r}. We then represent the backbone model parameters in the bottom left corner of the image and symmetrically distribute V_{s} and V_{s} on either side of the image diagonal based on the magnitude of α. Next, we measure the magnitudes of the two task vectors, $|V_{\text{s}}|$ and $|V_{\text{r}}|$, and set the longer task vector's value on the x-axis (or y-axis) of the image to 0.7, scaling the shorter vector proportionally,

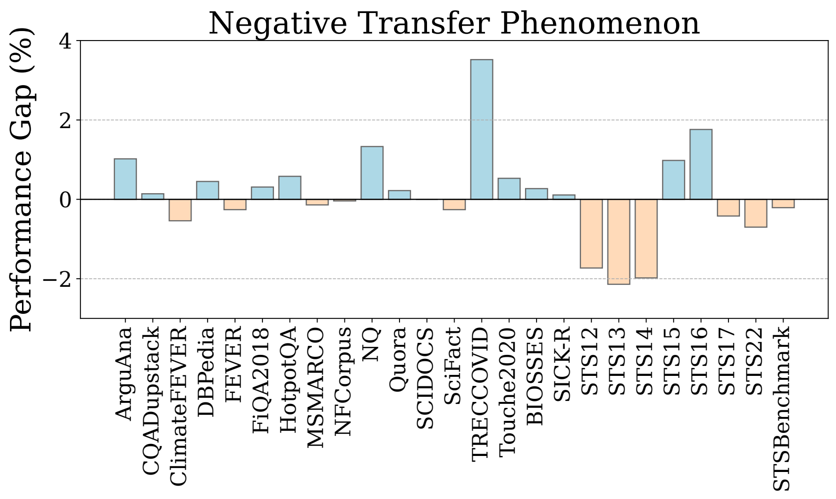


Figure 9: The difference in model performance obtained from joint training compared to the best performance achieved when trained separately on STS and Retrieval tasks in English.

Table 4: Statistics for the clustering results of 330 tasks.

	Task Count for each cluster	**Instance Count** for each cluster
2 clusters	30/300	1,255,000/18,000
3 clusters	30/121/179	1,255,000/72,600/107,400
4 clusters	30/92/97/111	1,255,000/55,200/58,200/66,600

to obtain the positions p_s and p_r of the two task vectors in the image. We then divide the image into a 20x20 grid, and each grid cell calculates its interpolation weight from the p_s and p_r, which is used as the model merging weight. We then measure the merged model's loss on a multi-task dataset, which represents the loss of one point on the loss landscape. For computing the loss, we sample data proportionally from the STS and Retrieval training data, calculate the loss with a batch size of 32, and ultimately compile the average result over 200 steps.

D Implementation of Clustering

In Section 6.1, we divide 330 tasks using a clustering algorithm. Specifically, we sample 100 instances each of q and p from every dataset, and then we obtain their embeddings using the backbone (i.e., `gtr-t5-large`). Next, we perform average pooling on the embeddings of q and p separately and concatenate them to form the dataset's corresponding embedding. We then use sklearn's [46] KMeans algorithm with the following parameters to perform clustering: `'init'='k-means++'`, `'n_init'=10`, `'max_iter'=300`. The final clustering statistics are shown in Table 4. The results of varying numbers of clusters are shown in Table 5.

E Extra Classification Training Data

We use the English training portions of various classification datasets from the MTEB [6]. The specific datasets used include: AmazonReviews Classification [47], AmazonCounterfactual Classification [48], Banking77-Classification [49], Emotion-Classification [50], IMDB- Classification [51], MTOPIntent-Classification [52], ToxicConversations Classification, and TweetSentimentExtraction Classification. We limit the maximum instance number of all datasets to 25,000.

Table 5: Results on MTEB [6]. Retri., Pair., Class., Sum. refer to retrieval, pair classification, classification, and summarization, respectively. All models utilize the T5-large encoder [42] as their backbone. Results of GTR and Instructor are sourced from [13]. Results from Separate Merging (Section 6.1) are categorized under “2 clusters”, “3 clusters”, and “4 clusters”. For methods requiring sample data (Fisher, RegMean, and our Self Positioning), the first result in each group uses training data, while the second uses alternative data sources.

	Avg. (56)		**Class. (12)**		**Cluster. (11)**		**Pair. (3)**		**ReRank. (4)**		**Retri. (15)**		**STS (10)**		**Sum. (1)**	
GTR [14]	58.3		67.1		41.6		85.3		55.4		47.4		78.2		29.5	
Instructor [13]	61.6		73.9		45.3		85.9		57.5		47.6		83.2		31.8	
2 clusters																
TIES	62.3		72.8		45.7		86.2		57.8		50.7		83.1		31.3	
Fisher	62.3	62.1	72.8	72.7	45.4	45.1	86.1	86.4	57.8	57.6	51.2	50.8	82.9	82.9	30.8	31.0
RegMean	61.9	61.9	72.4	72.5	45.2	45.2	86.5	86.6	57.3	57.3	50.1	50.0	83.0	83.0	31.1	31.0
Self Positioning	62.2	62.3	72.8	72.8	45.7	45.5	85.9	86.3	57.8	57.7	50.3	50.8	83.1	83.0	31.3	30.9
3 clusters																
TIES	62.2		72.5		45.4		86.3		57.7		51.1		82.8		31.2	
Fisher	62.1	61.7	72.3	72.2	45.2	44.6	86.4	86.7	57.5	56.9	51.0	50.2	82.7	82.4	30.9	31.6
RegMean	61.5	61.4	71.9	72.0	44.7	44.5	86.6	86.7	56.9	56.8	49.7	49.6	82.4	82.4	31.6	31.5
Self Positioning	62.3	62.2	72.6	72.6	45.6	45.4	86.0	86.3	58.0	57.7	51.0	51.1	82.9	82.7	31.4	31.3
4 clusters																
TIES	62.1		72.3		45.2		86.4		57.6		51.2		82.6		31.3	
Fisher	62.1	61.7	72.3	72.2	45.2	44.6	86.4	86.7	57.5	56.9	51.0	50.2	82.7	82.4	30.9	31.6
RegMean	61.5	61.4	71.9	72.0	44.7	44.5	86.6	86.7	56.9	56.8	49.7	49.6	82.4	82.4	31.6	31.5
Self Positioning	62.3	62.2	72.6	72.6	45.6	45.4	86.0	86.3	58.0	57.7	51.0	51.1	82.9	82.7	31.4	31.3
Iterative Merging (with extra classification training data)																
TIES	62.8		78.3		44.8		85.1		57.7		49.6		82.2		31.0	
Fisher	62.5		79.2		44.5		84.5		57.6		48.7		81.7		29.9	
RegMean	62.7		78.8		44.8		84.7		57.4		49.2		82.1		30.4	
Self Positioning	63.0		79.1		44.6		85.3		57.6		49.9		82.2		30.5	