SAME: Stability-Aware Embedding Extraction in Mixture-of-Experts Language Models

Shufan Yang* **Zifeng Cheng*** † **Zhiwei Jiang**† **Hao Wang**
Miao Xue **Changhui Sun** **Cong Wang** **Ao Zhou** **Qing Gu**
State Key Laboratory for Novel Software Technology, Nanjing University
sfyang@smail.nju.edu.cn {chengzf,jzw}@nju.edu.cn
{522025720019,miaoxue,502024330042}@smail.nju.edu.cn
wang.c@nju.edu.cn za@smail.nju.edu.cn guq@nju.edu.cn

Abstract

Extracting sentence embeddings from MoE-based large language models is a promising direction, as they provide greater model capacity than dense models at comparable computational cost. Existing works perform the standard forward propagation process to extract embeddings, overlooking a key stability requirement in the MoE encoding process: semantically similar texts should be encoded into similar representations. In this work, we identify two key stability-related phenomena in MoE models: (1) routing stability varies across layers, and (2) shared and routed experts exhibit different levels of stability. To address these issues, we propose SAME, a Stability-Aware MoE Embedding extraction framework that dynamically allocates activated experts across layers and adjusts expert outputs to mitigate component instability and improve embedding quality. Specifically, SAME assigns fewer activated experts to layers with less stable routing, where layer-wise routing stability is measured by the average overlap between the experts activated before and after injecting Gaussian noise into each layer's router inputs. Additionally, SAME reduces the contributions of routed experts to alleviate the impact of their instability. Notably, our method is training-free, seamlessly integrates with existing approaches, and incurs no additional inference overhead. Experiments on semantic textual similarity benchmarks demonstrate that SAME consistently improves the quality of extracted embeddings across multiple MoE backbones.

1 Introduction

Sentence embeddings play a fundamental role in a wide range of natural language processing tasks, including information retrieval (Zamani and Bendersky, 2023; Che et al., 2024; Oosterhuis et al., 2024), semantic matching (Nigam et al., 2019), and retrieval-augmented generation (Jiang et al., 2025b,a). Traditionally, high-quality sentence embeddings are obtained by training embedding models, which often requires substantial labeled data and computational resources. Recently, motivated by the success of Large Language Models (LLM), including those with Mixture-of-Experts (MoE) architectures, across diverse tasks, extracting sentence embeddings directly from general LLMs *without any training or data* has emerged as an appealing paradigm. This paradigm eliminates the need for additional data and costly fine-tuning, making it particularly suitable for resource-constrained scenarios. It also enhances the embedding capability of LLMs while preserving their general-purpose abilities.

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Prior work (Jiang et al., 2024; Springer et al., 2025; Fu et al., 2025; Zhang et al., 2024; Cheng et al., 2025) has demonstrated that carefully designed prompts can encourage LLMs to encode the semantics of an entire sentence into the representation of the final token, which can subsequently serve as a sentence embedding. As a representative example, PromptEOL (Jiang et al., 2024) employs a concise yet effective semantic compression prompt: `This sentence: ''[TEXT]'' means in one word : ''`, where `[TEXT]` denotes a placeholder for the input sentence. In this paradigm, the hidden state of the final token can be treated as the sentence embedding. Subsequently, several training-free methods (Springer et al., 2025; Fu et al., 2025) were proposed to overcome the limitations of unidirectional attention in LLMs. These training-free strategies substantially enhance embedding quality while avoiding the need for additional fine-tuning.

Compared to dense models, MoE models achieve greater capacity at comparable computational cost through sparse expert activation, making them a promising backbone for embedding extraction. MoEE (Li and Zhou, 2025) first demonstrates that MoE LLMs can serve as effective embedding models and proposes a post-processing method that incorporates routing weights to enhance embeddings. However, this approach extracts sentence embeddings through a standard forward pass, overlooking the stability of the MoE encoding process, which requires semantically similar texts to be encoded into similar representations. In particular, we observe that several components within MoE architectures exhibit poor stability during the encoding process. First, **routing stability varies substantially across MoE layers**, causing expert selection in unstable layers to be highly sensitive to small input perturbations. As a result, similar inputs may be routed to different experts, leading to inconsistent representations and degraded embedding quality. Second, **shared experts generally exhibit higher stability than routed experts**. Unstable experts may generate markedly different outputs for similar inputs, thereby disrupting representation consistency and harming sentence embedding quality. Furthermore, in light of these instability issues, we revisit the conventional practice of extracting embeddings from the final MoE sub-layer and find that it is suboptimal. In light of these instability issues, we investigate the embedding extraction position in MoE models and find that the commonly adopted last MoE sublayer is not optimal.

To this end, we propose **SAME, a Stability-Aware MoE Embedding extraction framework**, consisting of three components: Routed Expert Budget Reallocation, Routed-Shared Fusion Reweighting, and Embedding Extraction. For *expert reallocation*, we introduce a routing stability score to compute layer-wise expert activation budgets, reducing the number of activated experts in layers with lower routing stability. Specifically, we compute a routing stability score for each layer using a calibration set by measuring the average overlap of activated experts before and after injecting Gaussian noise into the router inputs. Notably, the total expert activation budget is kept fixed, and the strategy merely reallocates the number of activated experts at each layer. For *routed-shared fusion reweighting*, we down-weight the outputs of routed experts, while leaving shared experts unchanged, thereby alleviating the negative effects of instability in routed expert outputs on embedding quality. Finally, motivated by the instability of MoE components, we investigate different embedding extraction points and find that, within the same layer, embeddings extracted from the attention sub-layer often achieve better quality than those extracted from the MoE sub-layer. Therefore, we adopt embeddings extracted from the final attention sub-layer. Importantly, our framework introduces no additional inference overhead and can be seamlessly integrated with existing embedding extraction methods.

Our main contributions are summarized as follows:

- We revisit the components of MoE models and find that instability in both routers and experts can undermine the stability of the MoE encoding process.
- We propose SAME, a Stability-Aware MoE Embedding extraction framework that improves embedding quality by reallocating routed expert budgets across layers, reweighting routed-shared expert fusion, and extracting embeddings from the final attention sub-layer.
- Extensive experiments on semantic textual similarity benchmarks show that our method consistently outperforms existing training-free embedding extraction approaches across multiple MoE LLMs, demonstrating strong effectiveness and generalization.

2 Related Work

Sentence Embeddings Sentence embedding represents the semantics of a sentence as a fixed-length vector and plays a crucial role in various downstream tasks (Wang et al., 2024). More recently,

contrastive learning has become a dominant paradigm for sentence embedding, where models are fine-tuned using augmented views of the same input. Prominent approaches include SimCSE (Gao et al., 2021), and a series of augmentation- or prompt-based methods such as PromptBERT (Jiang et al., 2022), DiffCSE (Chuang et al., 2022), and other augmentation strategies (Su et al., 2023; Chanchani and Huang, 2023). With the remarkable capabilities demonstrated by LLMs, LLMs have also been employed as embedding models (Li and Li, 2024; BehnamGhader et al., 2024; Lee et al., 2024; Muennighoff et al., 2024b) through direct fine-tuning or instruction tuning. Although such methods lead to substantial improvements in embedding quality, they depend on large-scale training data and incur significant fine-tuning costs. In contrast, this paper focuses on extracting sentence embeddings from MoE models in a training-free manner.

Extracting Embeddings from LLMs Existing work on extracting embeddings from LLMs primarily relies on prompt-based strategies, where the hidden state of a specific token is used as the text representation. PromptEOL (Jiang et al., 2024) serves as a representative baseline, which carefully designs a prompt of the form: *This sentence: "[TEXT]" means in one word:"*, and directly uses the hidden state of the final token as the embedding. Building upon this approach, Echo (Springer et al., 2025) repeats the input sequence twice so that earlier tokens can attend to the semantic information of later tokens, and extracts embeddings from the second occurrence of the input. Furthermore, to avoid the additional computational overhead introduced by input repetition, TP (Fu et al., 2025) appends the sentence embedding from the previous layer as an auxiliary token at the beginning of the sequence. To improve conditional text embeddings, SCS (Cheng et al., 2026a) employs self-contrastive steering to refine conditional representations using unconditional text embeddings. To improve long-context embeddings, SCSP (Cheng et al., 2026b) leverages semantic compression to identify informative tokens and selectively aggregate their representations.

Extracting Embeddings from MoE Models MoEE (Li and Zhou, 2025) is the first to propose using the routing weights of MoE models as a complementary signal to token hidden states, thereby further improving embedding quality. However, this approach still preserves the original MoE architecture and does not account for the stability requirements of sentence embedding extraction. Our method is fundamentally different: we conduct an in-depth investigation of the unstable components within the MoE architecture and optimize these factors to improve embedding quality. Concurrently, AEA (Yang et al., 2026) proposes a training-free approach that dynamically allocates expert budgets based on token importance to improve sentence embeddings.

3 Preliminary Analysis

In this chapter, we revisit the Mixture-of-Experts (MoE) module and analyze the potential impact of its components on sentence embedding extraction. Our analysis focuses on two aspects. First, we show that routing stability varies across different MoE layers. Second, we demonstrate that stability differs between shared and routed experts. All experiments in this chapter are conducted on the STS-B validation set.

3.1 Mixture-of-Experts (MoE) Module

MoE architectures extend standard Transformer blocks by replacing the feed-forward network (FFN) sublayer with a set of expert networks and a routing mechanism that dynamically selects a subset of experts for each token. Formally, given the attention sublayer output $\mathbf{h}_{\text{att},i}^{(\ell)}$ for the i-th token at layer ℓ, the router computes a score vector $\mathbf{r}_i^{(\ell)} \in \mathbb{R}^M$ over M experts, which is then normalized into routing weights:

$$g_{i,e}^{(\ell)} = \frac{\exp\left(\mathbf{r}_{i,e}^{(\ell)}\right)}{\sum_{j=1}^{M} \exp\left(\mathbf{r}_{i,j}^{(\ell)}\right)}, \quad e \in \{1, \ldots, M\}. \tag{1}$$

Finally, the output of the MoE sublayer is computed as the sum of the routed expert outputs and the shared expert outputs:

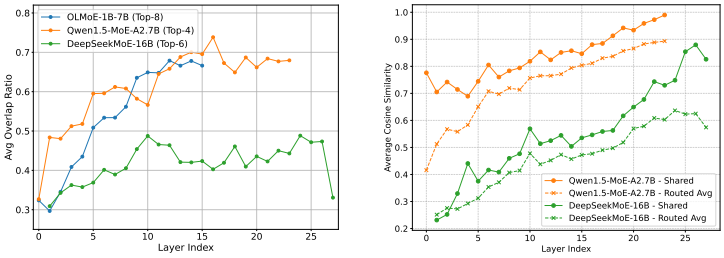
$$\begin{aligned}\mathrm{MoE}^{(\ell)}(\mathbf{h}^{(\ell)}_{\mathrm{att},i}) &= \sum_{s\in\mathcal{S}^{(\ell)}} \mathrm{E}^{(\ell)}_s(\mathbf{h}^{(\ell)}_{\mathrm{att},i}) \;+\; \sum_{e\in\mathcal{E}^{(\ell)}(\mathbf{h}^{(\ell)}_{\mathrm{att},i})} g^{(\ell)}_{i,e}\cdot \mathrm{E}^{(\ell)}_e(\mathbf{h}^{(\ell)}_{\mathrm{att},i}) \\ &= \mathbf{h}^{(\ell)}_{\mathrm{shared},i} + \mathbf{h}^{(\ell)}_{\mathrm{routed},i},\end{aligned} \tag{2}$$

where $\mathcal{S}^{(\ell)}$ denotes the set of shared experts at layer ℓ, $\mathcal{E}^{(\ell)}(\cdot)$ denotes the top-K routed experts with the highest routing weights, $\mathrm{E}^{(\ell)}_e(\cdot)$ and $\mathrm{E}^{(\ell)}_s(\cdot)$ denote the e-th routed expert and the s-th shared expert at layer ℓ, respectively, and $g^{(\ell)}_{i,e}$ is the routing weight assigned to expert e for token i.

In LLM-based embedding extraction, the sentence embedding is typically taken from the final-token representation at the last layer. In an MoE layer, this representation is first contextualized by the attention sub-layer and then passed through the MoE sub-layer. As a result, the conventional embedding $\mathbf{h}^{(L)}_T$ is inevitably affected by the final MoE transformation, whose instability may compromise embedding quality.

3.2 Motivation of Our Method

Existing embedding extraction methods typically keep the original MoE architecture unchanged. However, MoE is originally designed for next-token prediction, not for stable sentence representation learning. Since embedding extraction requires semantically similar inputs to produce similar embeddings, this raises a fundamental question: **are the key components of MoE sufficiently stable to support high-quality embedding extraction?**



(a) (b)

Figure 1: (a) The overlap of activated experts before and after Gaussian noise perturbation at each layer, reflecting routing stability. (b) The cosine similarity between expert outputs before and after Gaussian perturbation at each layer, reflecting the stability of the experts.

Layer-wise Routing Stability We first study layer-wise routing stability on three representative MoE models: DeepSeekMoE-16B (Dai et al., 2024), Qwen1.5-MoE-A2.7B (Team, 2024), and OLMoE-1B-7B (Muennighoff et al., 2024a). Specifically, we inject standard Gaussian noise into the router inputs at each layer and measure the overlap between the activated expert sets before and after the perturbation, averaged over all tokens in the STS-B validation set.

As shown in Figure 1a, expert overlap varies substantially across layers, indicating that some layers in MoE models exhibit poor routing stability. Under such conditions, when passing through layers with low routing stability, similar inputs may be routed to different experts, thereby amplifying output discrepancies. As a result, semantically similar texts can produce highly divergent embeddings, which severely undermines embedding consistency.

Inspiration This motivates us to **allocate fewer activated experts to layers with unstable routing, and reassign the freed expert budget to layers with more stable routing**, thereby improving embedding stability.

Shared vs. Routed Expert Stability We further investigate the stability of shared and routed experts. Specifically, we also inject Gaussian noise into the input of each expert and measure the similarity between the corresponding outputs before and after perturbation. As shown in Figure 1b, shared experts are consistently more stable than routed experts in nearly all layers.

Inspiration. This motivates us to **adjust the relative contribution of routed expert outputs**, thereby improving the stability of the extracted embeddings.

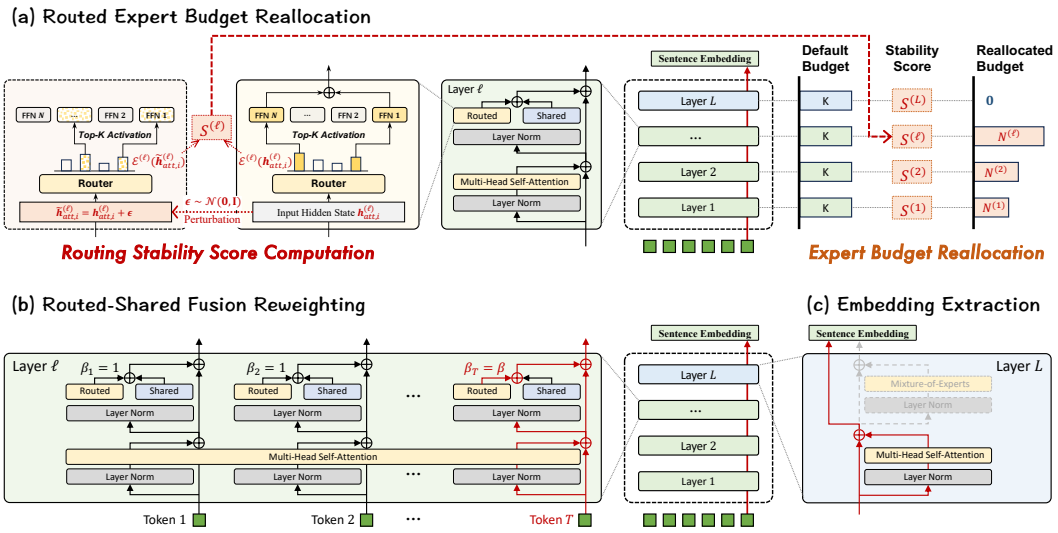


Figure 2: Illustration of our framework.

4 Methodology

Our empirical analysis reveals that several internal components of MoE models exhibit stability issues, suggesting that directly preserving the original MoE architecture may not be optimal for embedding extraction. To address this, we introduce a stability-aware MoE embedding extraction framework, consisting of routed expert budget reallocation, routed-shared fusion reweighting and embedding extraction. The overall framework is illustrated in Figure 2.

Specifically, our method first computes layer-wise routing stability scores on a calibration set and allocates an expert budget to each layer of the MoE model accordingly. This process only needs to be performed once in advance to determine the expert allocation for each layer. During inference, we appropriately adjust the relative contributions of shared and routed experts to the last-token representation in each MoE layer. The sentence embedding is then extracted from the output of the final attention sublayer after the residual connection.

4.1 Routed Expert Budget Reallocation

We propose a layer-wise expert reallocation strategy that assigns an appropriate expert budget to each layer based on the routing stability of its router. Intuitively, layers with low routing stability tend to route similar inputs to substantially different experts, which can compromise the stability of extracted embeddings. Accordingly, such layers are allocated fewer experts. Importantly, this allocation procedure is performed only once during a precomputation stage and can be reused thereafter without incurring additional overhead. The layer-wise expert reallocation module comprises two steps: routing stability score computation and expert budget reallocation.

Routing Stability Score Computation We first introduce a routing stability score to quantify the stability of routers at different layers under input perturbations. Specifically, we inject standard Gaussian noise into the router inputs and measure the overlap between the sets of experts activated before and after noise injection. Given a calibration set $\mathcal{D}$, where each sample $n \in \mathcal{D}$ consists of T_n tokens, we define the routing stability score of layer ℓ as the average over all tokens:

$$S^{(\ell)} = \frac{1}{\sum_{n\in\mathcal{D}} T_n} \sum_{n\in\mathcal{D}} \sum_{i=1}^{T_n} \left| \mathcal{E}^{(\ell)}(\mathbf{h}_{\text{att},i}^{(\ell)}) \cap \mathcal{E}^{(\ell)}(\tilde{\mathbf{h}}_{\text{att},i}^{(\ell)}) \right|. \tag{3}$$

Here, $\tilde{\mathbf{h}}_{\text{att},i}^{(\ell)} = \mathbf{h}_{\text{att},i}^{(\ell)} + \boldsymbol{\epsilon}$ denotes the perturbed input representation, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

In practice, we use the STS-B validation set as the calibration set, and a single precomputation is sufficient to obtain the routing stability scores for all layers. A larger value of $S^{(\ell)}$ indicates better routing stability, whereas a smaller value implies poorer routing stability.

Expert Budget Reallocation Then, we compute a new expert allocation budget $\mathbf{N} = [N^{(1)}, \ldots, N^{(L)}]$ based on the expert routing stability scores $\mathbf{S} = [S^{(1)}, \ldots, S^{(L)}]$. Specifically, for layer ℓ, we compute:

$$\widehat{N}^{(\ell)} = \left(\frac{(S^{(\ell)})^{\alpha}}{\sum_{j=1}^{L} (S^{(j)})^{\alpha}} \right) \cdot N_{\text{total}}; \qquad N^{(\ell)} = \text{round}\left(\widehat{N}^{(\ell)} \right). \tag{4}$$

Here, $N_{\text{total}} = K \times L$ is the total expert budget, with K denoting the default number of experts per layer and L the number of layers. The scaling factor α controls the reallocation disparity across layers, and $\text{round}(\cdot)$ rounds to the nearest integer.

Since rounding may introduce slight discrepancies between the final allocation and the original global expert budget, i.e., $\sum_{\ell=1}^{L} N^{(\ell)} \neq N_{\text{total}}$, we introduce a simple calibration step. Let N denote the total number of allocated experts, defined as $N = \sum_{i=1}^{L} N^{(i)}$. For each layer ℓ, we define the discrepancy between the continuous and rounded budgets as $\delta^{(\ell)} = \widehat{N}^{(\ell)} - N^{(\ell)}$. To ensure budget consistency, we iteratively adjusts the layer-wise expert allocation until $N = N_{\text{total}}$. Specifically, if $N < N_{\text{total}}$, we increment the expert budget of the layer with the largest discrepancy by one:

$$N^{(\ell)} \leftarrow N^{(\ell)} + 1, \quad \ell = \arg\max_{\ell} \delta^{(\ell)}. \tag{5}$$

Conversely, if $N > N_{\text{total}}$, we decrease the expert budget of the layer with the smallest discrepancy by one:

$$N^{(\ell)} \leftarrow N^{(\ell)} - 1, \quad \ell = \arg\min_{\ell} \delta^{(\ell)}. \tag{6}$$

This preserves the global budget $\sum_{\ell=1}^{L} N^{(\ell)} = N_{\text{total}}$, while favoring layers with higher routing stability.

4.2 Routed-Shared Fusion Reweighting

Motivated by the preceding observations, we further introduce a simple yet effective expert adjustment strategy that down-weights routed expert outputs within each MoE layer. Since the embedding is taken from the hidden state of the last token, we apply expert-specific scaling to the MoE output only for the last token, so as to amplify the contribution of the more stable shared experts while suppressing that of the less stable routed experts. The scaled MoE output is defined as follows:

$$\widetilde{\text{MoE}}^{(\ell)}(\mathbf{h}_{\text{att},i}^{(\ell)}) = \mathbf{h}_{\text{shared},i}^{(\ell)} + \beta_i \, \mathbf{h}_{\text{routed},i}^{(\ell)}, \tag{7}$$

where

$$\beta_i = \begin{cases} \beta, & i = T, \\ 1, & \text{otherwise}, \end{cases} \tag{8}$$

where $\beta \in (0, 1]$ is a layer-shared scaling factor. The hidden state is then updated as

$$\mathbf{h}_i^{(\ell)} = \mathbf{h}_{\text{att},i}^{(\ell)} + \widetilde{\text{MoE}}^{(\ell)}\left(\mathbf{h}_{\text{att},i}^{(\ell)} \right). \tag{9}$$

In this way, we only down-weight the routed expert outputs for the final token, while leaving the computation for all other tokens unchanged.

4.3 Embedding Extraction

Considering the potential instability introduced by routers and routed experts, we reconsider the conventional practice of extracting embeddings from the MoE sublayer. To this end, we conduct an exploratory study on the STS-B validation set by extracting embeddings from both the attention and MoE sublayers at each layer and comparing their quality. As illustrated in Figure 3, embeddings derived from the attention sublayer outperform those from the MoE sublayer in most layers, especially in the final layer. Based on this observation, we directly use the hidden state of

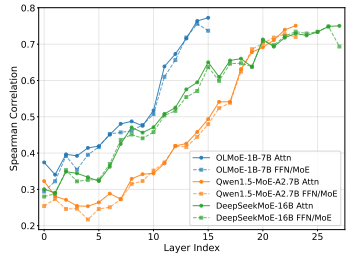


Figure 3: Embedding quality obtained from different layers, comparing representations extracted from different sublayers.

Table 1: Semantic textual similarity (STS) results on seven benchmarks using DeepSeekMoE-16B. **Bold** denotes the best result within each block, and ↑ indicates the absolute improvement over the corresponding HS baseline.

Baseline	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	52.13	69.33	54.75	57.99	68.72	63.83	65.27	61.72
HS + Echo	53.17	75.82	66.22	74.99	76.19	69.79	69.59	69.39 ↑ 7.67
HS + TP	59.32	77.35	64.85	71.40	74.80	65.98	70.29	69.14 ↑ 7.42
HS + SAME (**Ours**)	64.64	**81.02**	**68.46**	**76.71**	**77.77**	69.99	**73.11**	**73.10** ↑ 11.38
MoEE (concat)	66.83	77.53	63.55	64.61	71.04	66.39	68.57	68.36 ↑ 6.64
MoEE (concat) + Echo	56.28	69.01	56.53	61.49	67.55	66.10	64.18	63.02 ↑ 1.30
MoEE (concat) + TP	67.06	78.08	65.51	67.10	72.34	67.20	70.32	69.66 ↑ 7.94
MoEE (concat) + SAME (**Ours**)	**67.92**	78.44	65.57	68.26	73.07	67.32	70.55	70.16 ↑ 8.44
MoEE (sum)	66.68	80.95	67.94	66.84	73.56	69.64	72.58	71.17 ↑ 9.45
MoEE (sum) + Echo	56.08	68.89	56.35	61.46	67.57	66.05	64.06	62.92 ↑ 1.20
MoEE (sum) + TP	66.63	80.65	62.73	68.71	71.99	**70.74**	67.55	69.86 ↑ 8.14
MoEE (sum) + SAME (**Ours**)	64.63	79.83	65.69	72.37	74.81	70.32	70.86	71.22 ↑ 9.50

the last token from the final attention sub-layer as the sentence embedding:

$$\mathbf{e} = \mathbf{h}_{\text{att},T}^{(L)}. \tag{10}$$

This choice preserves the high-level semantic information captured by the top layer while avoiding the extra perturbation introduced by the final MoE transformation.

Expert Reallocation. Since embeddings are extracted from the final attention sublayer, the MoE module in the last layer does not contribute to the resulting sentence representations. Therefore, expert routing and expert computation in this layer become unnecessary during embedding extraction. To utilize the available expert capacity more effectively, we reallocate the experts originally assigned to the final MoE sublayer to earlier layers while keeping the total expert capacity fixed.

5 Experiments

5.1 Datasets and Experimental Settings

We evaluate our method on seven standard semantic textual similarity (STS) benchmarks, including STS 2012–2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), STS-B (Cer et al., 2017), and SICK-R (Marelli et al., 2014). Each sentence pair in these datasets is annotated with a human-assigned semantic similarity score ranging from 0 to 5. Following standard practice, we compute the cosine similarity between sentence embeddings and evaluate embedding quality using Spearman's rank correlation coefficient, which measures the rank consistency between predicted similarity scores and the ground-truth annotations.

We perform a grid search on the STS-B validation set to select the two scaling factors, α and β, ensuring no overlap between the validation and test sets. Specifically, α is searched within the range $(0, 5]$, while β is searched within the range $(0, 1]$. We set $\alpha = 3$ and $\beta = 0.6$ for DeepSeekMoE-16B, $\alpha = 2.5$ and $\beta = 0.8$ for Qwen1.5-MoE-A2.7B, and $\alpha = 1$ and $\beta = 0.6$ for OLMoE-1B-7B. The resulting layer-wise expert allocation for each model is illustrated in Figure 5 in the appendix.

5.2 Baselines

Given the plug-and-play nature of our approach, we integrate it with several baseline methods to evaluate its effectiveness. **MoEE** (Li and Zhou, 2025) is currently the only embedding extraction method specifically tailored to MoE models, where routing weights are incorporated as an additional representation signal to enhance sentence embeddings. Specifically, MoEE adopts concatenation-based combination (concat) and weighted-sum integration (sum) to combine routing weights with hidden states.

Furthermore, we compare our method against two general plug-and-play techniques that are applicable to large language models, both of which are designed to alleviate the limitations of unidirectional attention in autoregressive architectures. **Echo** (Springer et al., 2025) repeats the input text twice and extracts sentence embeddings by applying mean pooling over the representations of the second occurrence, whereas **TP** (Fu et al., 2025) prepends a sentence embedding token to the input sequence.

Table 2: Semantic textual similarity (STS) results on seven benchmarks using Qwen1.5-MoE-A2.7B. **Bold** denotes the best result within each block, and ↑ indicates the absolute improvement over the corresponding HS baseline.

Baseline	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	55.01	77.49	63.64	73.57	73.62	66.94	67.64	68.27
HS + Echo	55.93	77.60	66.27	**76.92**	**75.56**	69.29	67.41	69.85 ↑ 1.58
HS + TP	57.66	77.86	64.94	73.78	74.35	68.12	69.65	69.48 ↑ 1.21
HS + SAME (**Ours**)	**73.85**	79.74	**71.35**	72.60	71.30	**77.37**	73.98	**74.31** ↑ 6.04
MoEE (concat)	64.28	77.27	63.78	66.97	71.31	67.11	69.28	68.57 ↑ 0.30
MoEE (concat) + Echo	54.95	56.84	49.93	51.43	64.76	66.71	60.65	57.89 ↓ 10.38
MoEE (concat) + TP	66.25	77.54	66.00	68.54	72.36	68.30	71.65	70.09 ↑ 1.82
MoEE (concat) + SAME (**Ours**)	64.81	77.43	64.45	68.10	70.96	68.08	70.03	69.12 ↑ 0.85
MoEE (sum)	65.20	**81.87**	70.42	71.72	74.78	70.77	**74.49**	72.75 ↑ 4.48
MoEE (sum) + Echo	54.48	56.58	49.63	51.13	64.55	66.60	60.27	57.60 ↓ 10.67
MoEE (sum) + TP	66.55	80.27	68.94	70.24	74.02	71.50	71.65	71.88 ↑ 3.61
MoEE (sum) + SAME (**Ours**)	61.34	79.18	65.46	72.82	72.07	69.83	71.02	70.25 ↑ 1.98

Table 3: Semantic textual similarity (STS) results on seven benchmarks using OLMoE-1B-7B. **Bold** denotes the best result within each block, and ↑ indicates the absolute improvement over the corresponding HS baseline.

Baseline	STS12	STS13	STS14	STS15	STS16	SICK-R	STS-B	Avg.
HS	65.60	81.93	69.44	77.60	77.31	66.68	71.59	72.88
HS + Echo	56.75	74.92	65.51	77.72	75.83	69.95	71.14	70.26 ↓ 2.62
HS + TP	64.57	81.88	68.99	77.42	78.21	68.78	73.18	73.29 ↑ 0.41
HS + SAME (**Ours**)	66.87	**84.57**	**73.44**	**81.38**	**79.62**	71.78	**78.53**	**76.60** ↑ 3.72
MoEE (concat)	67.42	80.11	68.52	69.08	73.31	67.75	70.63	70.97 ↓ 1.91
MoEE (concat) + Echo	57.66	70.93	59.58	67.01	70.54	66.52	68.28	65.79 ↓ 7.09
MoEE (concat) + TP	67.86	80.12	68.68	70.24	74.04	68.74	71.36	71.58 ↓ 1.30
MoEE (concat) + SAME (**Ours**)	67.96	81.58	71.09	72.70	75.29	69.78	74.11	73.22 ↑ 0.34
MoEE (sum)	68.08	83.70	72.83	71.90	75.71	70.22	74.00	73.78 ↑ 0.90
MoEE (sum) + Echo	57.46	75.38	59.31	66.80	70.40	70.24	70.38	67.14 ↓ 5.74
MoEE (sum) + TP	**69.23**	77.32	71.13	69.53	73.94	69.60	70.90	71.66 ↓ 1.22
MoEE (sum) + SAME (**Ours**)	66.33	83.34	71.74	78.05	75.66	**72.07**	75.90	74.73 ↑ 0.85

Except for **Echo**, all other methods adopt the **PromptEOL** (Jiang et al., 2024) prompting strategy for sentence embedding extraction.

5.3 Main Results

The experimental results of the three MoE models on STS benchmarks are summarized in Table 1, 2, and 3. Among all methods applied on top of HS, our approach achieves the largest performance gains and consistently outperforms all baselines, including MoEE, Echo, and TP. These results demonstrate the effectiveness of our method. Notably, on the DeepSeekMoE-16B model, our approach yields an absolute improvement of 11.38 points, highlighting its substantial impact.

Interestingly, HS combined with our method yields better performance than MoEE combined with our method. A possible explanation is that routing information is implicitly encoded into the hidden representations by our approach, as routing stability is explicitly taken into account during layer-wise expert allocation. As a result, our method does not require the explicit incorporation of routing weights as an additional representation signal, which further simplifies the embedding extraction process.

Table 4: Ablation study on the STS-B using OLMoE-1B-7B.

Dataset	STS-B
HS	75.44
HS + layer-wise	76.92 ↑ 1.48
HS + attention	79.31 ↑ 3.87
HS + scaling	78.71 ↑ 3.27
HS + layer-wise + attention	80.66 ↑ 5.22
HS + layer-wise + scaling	79.14 ↑ 3.70
HS + attention + scaling	80.38 ↑ 4.94
HS + SAME (**Ours**)	**81.38** ↑ **5.94**

Compared to the other two plug-and-play methods, Echo and TP, our approach generally achieves more robust performance improvements. Our method fails to achieve performance improvements in only one case, when combined with MoEE (sum) on Qwen-1.5-MoE-A2.7B, whereas TP and Echo exhibit three and seven such cases, respectively. Furthermore, our method achieves consistent improvements across multiple MoE LLMs, demonstrating its generalizability.

5.4 Ablation Study

We study the impact of three design choices using OLMoE-1B-7B on the STS-B validation set: (i) layer-wise reallocation, (ii) extracting embeddings only from the final attention layer, and (iii) applying scaling exclusively to the outputs of MoE layers.

As shown in Table 4, each component individually improves embedding quality over the HS baseline, with final-layer attention extraction yielding the largest single gain. Notably, layer-wise allocation, while relatively modest on its own, consistently amplifies the benefits of other components when combined. Moreover, combining all three design choices leads to the best performance, indicating that these strategies are complementary rather than redundant.

Table 5: Comparison of reduced expert activation and all-token scaling strategies on STS-B using OLMoE-1B-7B.

Dataset	STS-B
HS	75.44
HS + last token (top 4)	76.07 ↑ 0.63
HS + all tokens scaling	76.85 ↑ 1.41
HS + scaling	**78.71** ↑ 3.27

5.5 Analysis of Scaling Only the Last Token in MoE Layers

Since embeddings are extracted from the hidden state of the last token, we propose to scale only the MoE output of the last token.

Using OLMoE-1B-7B, we investigate whether this strategy is optimal on the STS-B validation set. We first reduce the number of activated experts for the last token, as this also implicitly reduces the magnitude of the MoE output. We vary the number of activated experts from 1 to 8 and observe that activating four experts yields the best performance. As shown in Table 5, while this approach slightly outperforms HS, it is still inferior to HS+scaling, indicating that sufficient activated experts are necessary.

In addition, we also experiment with scaling the MoE outputs for all tokens. Similarly, we sweep the scaling factor in the range $(0, 1]$ and find that a scaling factor of 0.6 achieves the best performance. As shown in Table 5, this approach again yields performance between HS and HS+scaling. We attribute this behavior to the degradation of contextual representations caused by scaling the MoE outputs of context tokens, which reduces the amount of useful information available to the last token during embedding extraction.

5.6 Effect of Scaling Factor

We systematically investigate the effects of two hyperparameters, α and β, on embedding quality using the STS-B validation set.

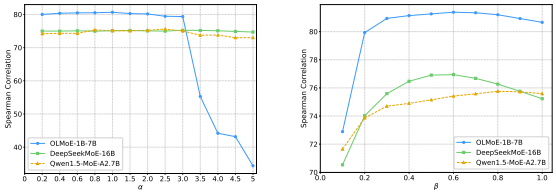


Figure 4: Effect of the scaling factor α and β.

For α, we observe that embedding quality first improves and then degrades as its value increases. A larger α amplifies the differences in the number of experts allocated across layers, enabling more targeted expert allocation and thereby improving embedding quality. However, when the inter-layer disparity becomes excessively large, layers with unstable routing behavior may be assigned too few experts, limiting their representational capacity and negatively affecting overall embedding quality.

A similar trend is observed for β: embedding quality first improves and then declines as β decreases. When β is large, MoE outputs dominate the final representations and may overshadow semantic information captured by attention layers. Moderately reducing β strengthens the relative contribution of attention layers and improves embedding quality. However, an overly small β excessively weakens the MoE/FFN components, limiting nonlinear transformation capacity and degrading performance.

Table 6: Results under different allocation strategies using OLMoE-1B-7B.

Dataset	STS-B
HS	75.44
HS + layer-wise (inverse)	74.43 ↓ 1.01
HS + layer-wise	76.92 ↑ 1.48

5.7 Validation of Allocation Assumptions

We further validate our initial motivation that layers with unstable routing behavior should be allocated fewer experts. As shown in Table 6, assigning more experts to routing-unstable layers leads to a

degradation in embedding quality. This observation suggests that increasing expert capacity in such layers amplifies routing variability rather than alleviating it, resulting in noisier representations.

6 Conclusion

In this work, we revisit the expert mechanisms in MoE architectures and propose SAME, a Stability-Aware MoE Embedding extraction framework, to improve embedding quality. SAME reallocates routed expert budgets across layers based on routing stability, reweights routed-shared expert fusion to reduce the impact of unstable routed experts, and extracts embeddings from the final attention sublayer for more stable representation construction. Notably, our method is training-free, seamlessly integrates with existing approaches, and incurs no additional inference overhead. Extensive experiments across multiple MoE backbones and a wide range of STS benchmarks demonstrate that our method consistently outperforms existing methods.

Acknowledgments and Disclosure of Funding

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the JiangSu Natural Science Foundation under Grant No. BK20251989; the National Natural Science Foundation of China under Grants Nos. 62606226, 62441225; the Jiangsu Funding Program for Excellent Postdoctoral Talent under Grants Nos. 2026ZB185, 2026ZB245; the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118); the "111 Center" (No. B26023). This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

Agirre, E., Banea, C., Cardie, C., Cer, D. M., Diab, M. T., Gonzalez-Agirre, A., Guo, W., Lopez-Gazpio, I., Maritxalar, M., Mihalcea, R., Rigau, G., Uria, L., and Wiebe, J. (2015). Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015*, pages 252–263.

Agirre, E., Banea, C., Cardie, C., Cer, D. M., Diab, M. T., Gonzalez-Agirre, A., Guo, W., Mihalcea, R., Rigau, G., and Wiebe, J. (2014). Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014*, pages 81–91.

Agirre, E., Banea, C., Cer, D. M., Diab, M. T., Gonzalez-Agirre, A., Mihalcea, R., Rigau, G., and Wiebe, J. (2016). Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016*, pages 497–511.

Agirre, E., Cer, D. M., Diab, M. T., and Gonzalez-Agirre, A. (2012). Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012*, pages 385–393.

Agirre, E., Cer, D. M., Diab, M. T., Gonzalez-Agirre, A., and Guo, W. (2013). *sem 2013 shared task: Semantic textual similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013*, pages 32–43.

BehnamGhader, P., Adlakha, V., Mosbach, M., Bahdanau, D., Chapados, N., and Reddy, S. (2024). Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

Cer, D. M., Diab, M. T., Agirre, E., Lopez-Gazpio, I., and Specia, L. (2017). Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055.

Chanchani, S. and Huang, R. (2023). Composition-contrastive learning for sentence embeddings. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15836–15848.

Che, T.-Y., Mao, X.-L., Lan, T., and Huang, H. (2024). A hierarchical context augmentation method to improve retrieval-augmented llms on scientific papers. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 243–254.

Cheng, Z., Qian, L., Jiang, Z., Wang, C., Yin, Y., Shen, F., Zhou, A., and Gu, Q. (2026a). Focusing condition: Inference-time self-contrastive steering elicits better conditional text embeddings in LLMs. In Liakata, M., Moreira, V. P., Zhang, J., and Jurgens, D., editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1135–1147, San Diego, California, United States. Association for Computational Linguistics.

Cheng, Z., Wang, Z., Fu, Y., Jiang, Z., Yin, Y., Wang, C., and Gu, Q. (2025). Contrastive prompting enhances sentence embeddings in llms through inference-time steering. *arXiv preprint arXiv:2505.12831*.

Cheng, Z., Zheng, J., Jiang, Z., Wang, S., Shen, F., Ge, S., and Gu, Q. (2026b). Selecting what matters: Semantic compression-guided selective pooling for long-context embeddings. *arXiv preprint arXiv:2609.37782*.

Chuang, Y.-S., Dangovski, R., Luo, H., Zhang, Y., Chang, S., Soljačić, M., Li, S.-W., Yih, S., Kim, Y., and Glass, J. (2022). Diffcse: Difference-based contrastive learning for sentence embeddings. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4207–4218.

Dai, D., Deng, C., Zhao, C., Xu, R., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., et al. (2024). Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Fu, Y., Cheng, Z., Jiang, Z., Wang, Z., Yin, Y., Li, Z., and Gu, Q. (2025). Token prepending: A training-free approach for eliciting better sentence embeddings from llms. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*, pages 3168–3181.

Gao, T., Yao, X., and Chen, D. (2021). Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Jiang, P., Ouyang, S., Jiao, Y., Zhong, M., Tian, R., and Han, J. (2025a). Retrieval and structuring augmented generation with large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 6032–6042.

Jiang, T., Huang, S., Luan, Z., Wang, D., and Zhuang, F. (2024). Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3182–3196.

Jiang, T., Jiao, J., Huang, S., Zhang, Z., Wang, D., Zhuang, F., Wei, F., Huang, H., Deng, D., and Zhang, Q. (2022). Promptbert: Improving bert sentence embeddings with prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8826–8837.

Jiang, W., Zhang, S., Han, B., Wang, J., Wang, B., and Kraska, T. (2025b). Piperag: Fast retrieval-augmented generation via adaptive pipeline parallelism. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 589–600.

Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M., Catanzaro, B., and Ping, W. (2024). Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Li, X. and Li, J. (2024). Bellm: Backward dependency enhanced large language model for sentence embeddings. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 792–804.

Li, Z. and Zhou, T. (2025). Your mixture-of-experts LLM is secretly an embedding model for free. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., and Zamparelli, R. (2014). A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, pages 216–223.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., et al. (2024a). Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Muennighoff, N., Su, H., Wang, L., Yang, N., Wei, F., Yu, T., Singh, A., and Kiela, D. (2024b). Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. (2022). Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Nigam, P., Song, Y., Mohan, V., Lakshman, V., Ding, W., Shingavi, A., Teo, C. H., Gu, H., and Yin, B. (2019). Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2876–2885.

Oosterhuis, H., Jagerman, R., Qin, Z., Wang, X., and Bendersky, M. (2024). Reliable confidence intervals for information retrieval evaluation using generative ai. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2307–2317.

Springer, J. M., Kotha, S., Fried, D., Neubig, G., and Raghunathan, A. (2025). Repetition improves language model embeddings. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

Su, H., Shi, W., Kasai, J., Wang, Y., Hu, Y., Ostendorf, M., Yih, W.-t., Smith, N. A., Zettlemoyer, L., and Yu, T. (2023). One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121.

Team, Q. (2024). Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters".

Wang, C., Tian, K., Guan, Y., Shen, F., Jiang, Z., Gu, Q., and Zhang, J. (2024). Ensembling diffusion models via adaptive feature aggregation. *arXiv preprint arXiv:2405.17082*.

Yang, S., Cheng, Z., Jiang, Z., Qi, Q., Yin, Y., Wang, C., Zhou, A., and Gu, Q. (2026). AEA: Adaptive expert allocation improves sentence embeddings from mixture-of-experts LLM. In Liakata, M., Moreira, V. P., Zhang, J., and Jurgens, D., editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28837–28851, San Diego, California, United States. Association for Computational Linguistics.

Zamani, H. and Bendersky, M. (2023). Multivariate representation learning for information retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 163–173.

Zhang, B., Chang, K., and Li, C. (2024). Simple techniques for enhancing sentence embeddings in generative language models. *CoRR*, abs/2404.03921.

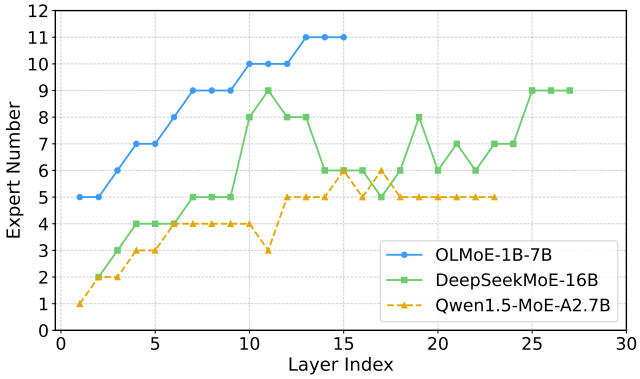


Figure 5: Counts of Layer-wise expert reallocation counts for three MoE models.

Table 7: Results under different prompt settings using OLMoE-1B-7B.

Prompt	Baseline	STS-B
Prompt	HS	66.37
Prompt	HS + SAME (**Ours**)	68.67 ↑ 2.30
Pretended CoT	HS	77.59
Pretended CoT	HS + SAME (**Ours**)	81.05 ↑ 3.46
Knowledge	HS	78.75
Knowledge	HS + SAME (**Ours**)	80.08 ↑ 1.33

A Limitations

Our work has two main limitations. First, our method requires access to router and expert outputs, which may be unavailable in closed-source MoE-based LLMs. Second, our evaluation focuses on English benchmarks, leaving its effectiveness on multilingual datasets unexplored.

B Generalization across Different Prompts

We further investigate the generalization of our method under different prompts on the STS-B validation set. These prompts are presented as follows:

PromptEOL: This sentence: "`[TEXT]`" means in one word: "
Prompt: In one word, describe the style of the following sentence -"`[TEXT]`": "
Pretended CoT: After thinking step by step, this sentence: "`[TEXT]`" means in one word: "
Knowledge: The essence of a sentence is often captured by its main subjects and actions, while descriptive terms provide additional but less central details. With this in mind , this sentence: "`[TEXT]`" means in one word: "

As shown in Table 7, our approach achieves improved performance under all three prompts. This result indicates that the gains brought by our method are not tied to a specific prompt design, but instead reflect a more general improvement in representation quality. Such robustness suggests that our method generalizes well across prompt variations and does not rely on carefully engineered prompts to be effective.

C MTEB Results

We additionally evaluate our method on the MTEB benchmark (Muennighoff et al., 2022), which covers a diverse set of natural language processing tasks, including classification, clustering, pair classification, re-ranking, retrieval, summarization, and semantic textual similarity. In accordance with the MTEB evaluation protocol, we report task-specific metrics such as Accuracy, V-Measure, Average Precision, Mean Average Precision, nDCG, and Spearman's rank correlation. As shown in Table 8, our method attains the strongest overall performance across all evaluated tasks.

Table 8: MTEB benchmark results using OLMoE-1B-7B. We report task-averaged scores for six MTEB task categories, including classification (CLF), clustering (Clust.), pair classification (PairCLF), reranking (Rerank), semantic textual similarity (STS), and summarization (Summ.). **Bold** indicates the best result within each block.

Baseline	CLF	Clust.	PairCLF	Rerank	STS	Summ.	Avg.
HS	58.11	32.97	72.14	58.34	72.88	27.97	53.74
HS + Echo	54.25	30.80	68.45	55.86	70.26	26.58	51.03
HS + TP	59.39	26.41	73.29	59.33	73.29	27.30	53.17
HS + SAME (**Ours**)	**59.62**	35.28	**73.88**	**60.92**	**76.60**	27.25	**55.59**
MoEE (concat)	51.37	29.31	71.27	56.61	70.97	28.91	51.41
MoEE (concat) + Echo	47.64	26.29	64.64	54.38	65.79	29.01	47.96
MoEE (concat) + TP	52.75	28.05	71.51	57.22	71.58	27.84	51.49
MoEE (concat) + SAME (**Ours**)	52.66	30.93	72.93	58.13	73.22	28.91	52.80
MoEE (sum)	52.36	34.37	72.52	58.12	73.78	28.85	53.33
MoEE (sum) + Echo	49.70	34.40	69.45	57.93	67.14	28.00	51.10
MoEE (sum) + TP	52.83	35.16	72.98	59.12	71.66	28.29	53.34
MoEE (sum) + SAME (**Ours**)	53.19	**35.39**	73.67	59.68	74.73	**29.27**	54.32

Table 9: Clustering results.

Baseline	Twenty Newsgroups Clustering	Medrxiv ClusteringS2S	Avg.
HS	39.19	26.75	32.97
HS + Echo	34.84	26.77	30.80
HS + TP	28.42	24.39	26.41
HS + SAME (**Ours**)	**42.80**	27.76	35.28
MoEE (concat)	34.15	24.46	29.31
MoEE (concat) + Echo	27.97	24.61	26.29
MoEE (concat) + TP	34.55	21.56	28.05
MoEE (concat) + SAME (**Ours**)	36.64	25.22	30.93
MoEE (sum)	36.51	32.24	34.37
MoEE (sum) + Echo	37.09	31.72	34.40
MoEE (sum) + TP	37.92	32.40	35.16
MoEE (sum) + SAME (**Ours**)	38.39	**32.38**	**35.39**

In addition, we provide detailed results for individual task types, covering clustering (Table 9), classification (Table 10), pair classification (Table 11), and re-ranking (Table 12).

D Computational Overhead Analysis

We analyze the computational overhead associated with computing the layer-wise routing stability scores. For each MoE model, these scores are computed only once and can be reused across all downstream tasks. In practice, we compute the stability scores on the STS-B validation set, which contains 1500 samples. The entire procedure requires only 2 to 3 minutes on a single NVIDIA V100 GPU, resulting in negligible overhead.

We further examine the inference-time overhead of our method. As reported in Table 13, our approach does not incur additional inference cost and slightly reduces inference latency. This improvement comes from early exiting before the final MoE sublayer while keeping the total expert budget fixed.

E Broader application scope

We further evaluate OLMoE-1B-7B on four retrieval tasks and the cross-lingual STS17 benchmark. As shown in Table 14 and 15, SAME consistently improves the HS baseline across all retrieval tasks, increasing the average score from 29.90 to 37.55. Moreover, SAME substantially improves the performance on STS17 from 53.54 to 66.08, demonstrating its effectiveness in both retrieval and cross-lingual semantic similarity tasks.

Table 10: Classification results.

Baseline	Tweet Sentiment Extraction	Emotion Classification	Toxic Conversations Classification	Avg.
HS	59.60	47.36	67.38	58.11
HS + Echo	53.26	41.58	67.91	54.25
HS + TP	60.70	48.95	68.54	59.39
HS + SAME (**Ours**)	**61.03**	**50.68**	67.16	**59.62**
MoEE (concat)	57.67	40.63	55.80	51.37
MoEE (concat) + Echo	53.37	35.53	54.01	47.64
MoEE (concat) + TP	59.33	44.05	54.87	52.75
MoEE (concat) + SAME (**Ours**)	58.81	42.79	56.39	52.66
MoEE (sum)	55.14	35.02	66.91	52.36
MoEE (sum) + Echo	47.25	30.55	**71.30**	49.70
MoEE (sum) + TP	55.42	36.12	66.96	52.83
MoEE (sum) + SAME (**Ours**)	53.94	35.71	69.92	53.19

Table 11: Pair classification results.

Baseline	TwitterURLCorpus	TwitterSemEval2015	Avg.
HS	82.39	61.88	72.14
HS + Echo	77.63	59.28	68.45
HS + TP	82.16	**64.43**	73.29
HS + SAME (**Ours**)	**83.83**	63.92	**73.88**
MoEE (concat)	81.00	61.53	71.27
MoEE (concat) + Echo	71.60	57.69	64.64
MoEE (concat) + TP	80.71	62.31	71.51
MoEE (concat) + SAME (**Ours**)	83.03	62.83	72.93
MoEE (sum)	82.47	62.57	72.52
MoEE (sum) + Echo	79.08	59.82	69.45
MoEE (sum) + TP	82.21	63.74	72.98
MoEE (sum) + SAME (**Ours**)	83.35	63.99	73.67

Table 12: Re-ranking results.

Baseline	AskUbuntu DupQuestions	SciDocsRR	StackOverflow DupQuestions	Avg.
HS	55.38	78.25	41.38	58.34
HS + Echo	54.43	69.99	43.17	55.86
HS + TP	55.89	79.34	42.75	59.33
HS + SAME (**Ours**)	**57.52**	**81.61**	**43.63**	**60.92**
MoEE (concat)	52.68	77.49	39.67	56.61
MoEE (concat) + Echo	48.60	76.22	38.33	54.38
MoEE (concat) + TP	52.97	78.20	40.48	57.22
MoEE (concat) + SAME (**Ours**)	53.93	79.17	41.29	58.13
MoEE (sum)	53.89	78.75	41.71	58.12
MoEE (sum) + Echo	52.52	77.63	43.62	57.93
MoEE (sum) + TP	54.76	79.44	43.15	59.12
MoEE (sum) + SAME (**Ours**)	55.11	80.82	43.11	59.68

Table 13: Inference time overhead under the PromptEOL setting using OLMoE-1B-7B.

Baseline	**Time**
HS	1.00 ×
HS + SAME (**Ours**)	0.94 ×
MoEE (concat)	1.00 ×
MoEE (concat) + SAME (**Ours**)	0.96 ×
MoEE (sum)	1.00 ×
MoEE (sum) + SAME (**Ours**)	0.93 ×

Table 14: Retrieval performance on four benchmarks.

Method	ArguAna	NFCorpus	SCIDOCS	SciFact	Avg.
HS	32.30	19.99	12.38	54.92	29.90
HS + SAME (Ours)	39.54	30.21	16.30	64.16	**37.55**

Table 15: STS17 results.

Method	STS17
HS	53.54
HS + SAME (Ours)	**66.08**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have described our claims and contributions clearly in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We only demonstrate the effectiveness of the methodology only on the English corpus.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: We prove our claims with extensive empirical results, instead of theoretical proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided these details in the Implementation Details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release the data and code after the paper is accepted.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided these details in the Implementation Details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The experiments support the main claims of the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided sufficient information in the Experimental Settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper does not have such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All of the creators or original owners of assets used in our paper are cited properly.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will release the data and code including documentation after the paper is accepted.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper does not involve research with human subjects.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.