# Focusing Condition: Inference-Time Self-Contrastive Steering Elicits Better Conditional Text Embeddings in LLMs

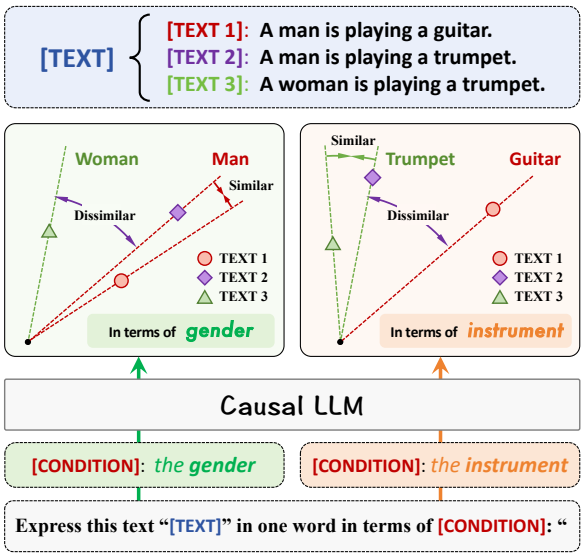
**Zifeng Cheng**[1] **Lingyun Qian**[1] **Zhiwei Jiang**[1]*
**Cong Wang**[1] **Yafeng Yin**[1] **Fei Shen**[2] **Ao Zhou**[1] **Qing Gu**[1]
[1] State Key Laboratory for Novel Software Technology, Nanjing University
[2] National University of Singapore
chengzf@nju.edu.cn 522024330064@smail.nju.edu.cn
{jzw,wang.c,yafeng}@nju.edu.cn shenfei29@nus.edu.sg
za@smail.nju.edu.cn guq@nju.edu.cn

## Abstract

Extracting conditional text embeddings from large language models (LLMs) is a promising paradigm, as it requires neither additional data nor fine-tuning. Existing methods incorporate conditions into prompts to guide LLMs to focus on specific aspects and elicit conditional text embeddings. However, relying solely on prompts often fails to produce high-quality conditional text embeddings, as they remain entangled with general text embeddings, ultimately degrading their quality. To this end, we propose an inference-time, plug-and-play Self-Contrastive Steering (SCS) method that constructs unconditional general text embeddings and uses them to refine conditional text embeddings, making them more focused on the target condition. Specifically, we modify the attention mask and positional encodings to mask the condition, thereby obtaining unconditional text embeddings and intervening in the multi-head self-attention computation process. Notably, our method is highly efficient, requiring only a single additional multi-head self-attention computation at inference time. Extensive experiments on clustering, Semantic Textual Similarity, and triplet alignment datasets demonstrate that our method can seamlessly improve the performance of existing prompt-based methods across different LLMs in a training-free and plug-and-play manner. Our code will be released at https://github.com/zifengcheng/SCS.

## 1 Introduction

Text embeddings aim to represent a text as a generic fixed-size vector and have a wide range of applications, including text classification, clustering, semantic textual similarity (Agirre et al., 2012, 2013), and information retrieval. In practice, text often contains multiple types of semantics, and a single universal embedding cannot satisfy users' specific fine-grained embedding needs from multiple aspects (Deshpande et al., 2023; Wang et al., 2023). For example, as shown in Figure 1, TEXT 1 and TEXT 2 are similar with respect to *gender*, but they differ with respect to the *instrument*. Therefore, conditional text embeddings are crucial for a wide range of downstream tasks.




***Prompt*** with condition description

Figure 1: An example of the conditional text embedding task. Notably, compared to labels, conditions often correspond to coarser-grained information.

Given the success of Large Language Models (LLMs) on a wide range of NLP tasks in zero-shot settings, several studies (Yamada and Zhang, 2025; Jiang et al., 2024; Springer et al., 2025; Lei et al., 2024) focus on *prompt engineering* to directly extract text embeddings from LLMs *without any data and further training*. This training-free, zero-shot setting offers a substantial improvement over methods that rely on extensive fine-tuning to achieve high performance. For example, PonTE (Yamada and Zhang, 2025) builds on PromptEOL (Jiang et al., 2024) by adding condition and introduces a

*Corresponding author.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1135–1147

simple and effective *semantic compression* prompt that compresses the conditional text embedding into the last token: *Express this text:* "`[TEXT]`" *in one word in terms of* "`[CONDITION]`"*: "*, where `[TEXT]` and `[CONDITION]` serve as placeholders for the text and condition. The phrase "in one word" encourages the LLM to condense the semantic meaning of the text into the last token.

Even if these methods are effective, embeddings obtained by relying solely on the conditions in the prompt still encode other condition-irrelevant information. Empirically, we observe that conditional text embeddings still encode general textual information, which affects their distance measurements and ultimately undermines their performance. For example, the average cosine similarity between general text embeddings and conditional text embeddings reaches as high as 0.763 on the TweetEmotion dataset. Moreover, prompts are sensitive to subtle variations (Lu et al., 2022), and the mechanisms underlying their effectiveness remain unclear (Shi et al., 2024).

To solve these issues, we propose a simple and effective plug-and-play Self-Contrastive Steering (SCS) method to enhance existing prompt engineering methods. SCS adjusts the attention mask and positional encoding to derive multi-view embeddings from the text, including both unconditional text embeddings and conditional text embeddings. By contrasting the two embeddings, SCS can steer the embeddings to focus on condition-related information while filtering out unconditional general text information. Specifically, we first forward propagate the prompt-wrapped text to the specific layer and obtain the hidden states. Next, we obtain an unconditional text embedding by modifying the attention mask and positional encoding, allowing us to compare it with the conditional text embedding. Furthermore, we propose two self-contrastive activation steering methods, direct elimination and projection-based elimination, to purify the conditional text embeddings. Finally, we continue to forward propagate the refined representations to obtain the conditional text embeddings.

Our main contributions are as follows:

- We first propose a lightweight approach to create multi-view embeddings for a text by modifying the attention mask and positional encoding.
- We propose a Self-Contrastive Steering method, which leverages two self-generated embeddings to guide the model to focus on the conditional part.
- We conduct extensive experiments on seven datasets spanning clustering, Semantic Textual Similarity (STS), and triplet alignment tasks. Experimental results demonstrate that our proposed method significantly improves the performance of existing prompt-based methods across different LLMs.

## 2 Related Work

**Text Embedding** Text embedding aims to use a general fixed-size vector to represent a text, playing a fundamental role in various applications (Yang et al., 2026). Existing methods typically use contrastive learning to fine-tune models for text embedding (Gao et al., 2021; Jiang et al., 2022; Ni et al., 2022), where positive pairs are pulled closer and negative pairs are pushed apart. A classic work is SimCSE (Gao et al., 2021), which uses dropout to generate positive samples for contrastive learning. Recently, some works (Li and Li, 2024; BehnamGhader et al., 2024) have begun to fine-tune LLMs to obtain text embeddings.

**Conditional Text Embedding** Unlike general text embeddings, conditional text embeddings can selectively focus on specific aspects of a text to more precisely meet user needs at a finer granularity. Existing approaches primarily fine-tune LLMs to obtain conditional text embedding. InstructOR (Su et al., 2023) first integrates instructions into text embeddings for generating task-relevant embeddings. It annotates instructions for 330 different tasks and employs contrastive learning for training. C-STS (Deshpande et al., 2023) first annotates a conditional semantic textual similarity dataset to accurately evaluate the similarity between sample pairs. InBedder (Peng et al., 2024) leverages the generative capabilities of LLMs to re-encode the generated content into embeddings and fine-tunes an LLM on preprocessed QA datasets with short responses. HyperCL (Yoo et al., 2024) integrates hypernetworks with contrastive learning to compute conditioned sentence representations. However, these methods often require large amounts of data and incur high fine-tuning costs, while also sacrificing the general capabilities of the LLM by turning it into an embedding model.

**Extracting Conditional Embeddings from LLMs** Early work primarily focus on extract-

ing general text embeddings from LLMs using prompt engineering methods (Jiang et al., 2024; Springer et al., 2025; Lei et al., 2024). For example, PromptEOL (Jiang et al., 2024) first proposes a simple and effective *semantic compression* prompt that compresses the text embedding into the last token: *This sentence:* "`[TEXT]`" *means in one word: "*. Subsequently, considering the causal attention property of LLMs, Echo (Springer et al., 2025) proposes duplicating the text twice, while TP (Fu et al., 2025) prepends the sentence embedding token to allow all tokens in the sentence to perceive the complete semantics. Recently, considering the need for more fine-grained embeddings, PonTE (Yamada and Zhang, 2025) first adds conditions on top of PromptEOL to extract more fine-grained conditional text embeddings. However, relying solely on the conditions in the prompt cannot ensure that the output embeddings capture only condition-relevant information.

**Activation Steering** Activation steering (Zou et al., 2023) is a training-free technique that creates steering vectors to modify the activations of LLMs, thereby controlling their generation. The first type of method (Rimsky et al., 2024; Li et al., 2023a; Cheng et al.) generates steering vectors from positive-negative sample pairs, often relying on supervised data. The second type (Leong et al., 2023; Postmus and Abreu, 2024; Cheng et al., 2025) constructs auxiliary prompts through forward passes, typically incurring additional computational overhead. Unlike these methods, our approach requires no supervised data, constructs input-related steering vectors, and only requires a single additional multi-head self-attention computation.

## 3 Preliminary

**Extracting Conditional Text Embeddings from LLMs** Existing methods mainly focus on prompt engineering techniques to *compress* the semantics of a text into a single token for extracting embeddings from LLMs. PonTE (Yamada and Zhang, 2025) builds upon PromptEOL (Jiang et al., 2024) by incorporating additional conditions to extract conditional text embeddings:

> **PonTE**: Express this text "`[TEXT]`" in one word in terms of `[Condition]`: "

where `[TEXT]` and `[Condition]` denote placeholders for the text and the condition, respectively. The phrase "in one word" encourages the LLM to condense the semantic meaning of the text into the hidden state of the next token. Notably, the hidden state of the last token " already encodes information about the next token for next-token prediction, allowing us to directly use its representation as the conditional text embedding without generation.

**Self-Attention with RoPE in LLMs** LLMs employ Rotary Position Embedding (RoPE) (Su et al., 2024) in the Multi-Head Self-Attention (MHSA) module to encode relative positional information by applying a position-dependent rotation to each query and key vector before computing attention. Specifically, the $\ell$-th MHSA layer receives the hidden state at position $m$ from the previous layer and transforms it into the corresponding query and key representations as follows:

$$\mathbf{q}_m^\ell = \mathrm{RoPE}_q(\mathbf{h}^{\ell-1}, m) = \mathbf{W}_q^\ell \mathbf{h}^{\ell-1} e^{im\theta} \quad (1)$$

$$\mathbf{k}_m^\ell = \mathrm{RoPE}_k(\mathbf{h}^{\ell-1}, m) = \mathbf{W}_k^\ell \mathbf{h}^{\ell-1} e^{im\theta} \quad (2)$$

where $\mathbf{W}_q^\ell$ and $\mathbf{W}_k^\ell$ denote the query and key projection matrices at layer $\ell$, and $\theta$ is a dimension-dependent constant that determines the rotation frequency.

## 4 Method

Our proposed SCS method is an inference-time plug-and-play steering method that requires no additional data or fine-tuning of the LLM, and can integrate with existing prompt engineering techniques.

SCS consists of three steps to obtain conditional text embedding, as illustrated in Figure 2. In the first step, it inputs the prompt-wrapped input into the LLM up to the $(\ell-1)$-th Transformer layer to obtain the output hidden states. Next, it modifies the attention mask to hide the conditional information in the prompt and further adjusts the positional encodings to make them contiguous, thereby extracting an unconditional text embedding at layer $\ell$. Finally, it performs self-contrast between the two embeddings obtained with and without the condition to remove general information from the conditional text embedding, and then continues to forward the purified representations to obtain the final conditional text embedding.

### 4.1 Prompt Construction

In the first step, we construct a prompt (i.e., Express this text "`[TEXT]`" in one word in terms of

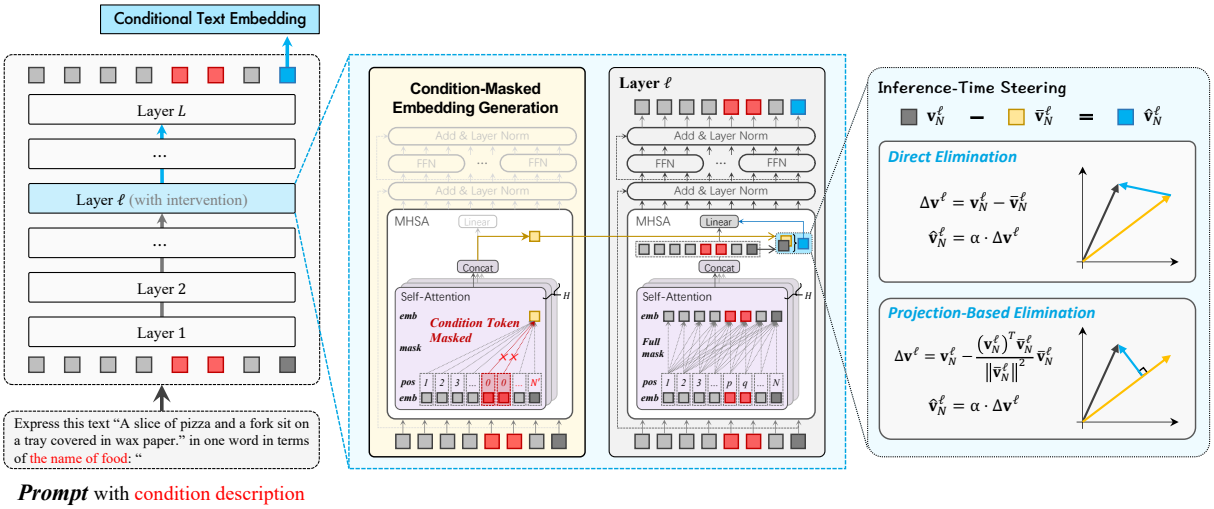



Figure 2: Illustration of the self-contrastive steering method. Our method additionally generates an unconditional text embedding through condition-masked embedding generation and introduces two activation steering strategies.

"[CONDITION]": ") to wrap the text and feed it into the LLM to obtain the hidden states from the $(\ell-1)$-th layer.

Formally, given a prompt-wrapped input $T = [t_1, \ldots, t_{p-1}, t_p, \ldots, t_q, t_{q+1}, \ldots, t_N]$, where $t_p$ to $t_q$ constitute the conditional part and $N$ denotes the input length, we feed it into the LLM and obtain the output hidden states (i.e., $\mathbf{H}^{\ell-1} = [\mathbf{h}_1^{\ell-1}, \cdots, \mathbf{h}_N^{\ell-1}]$) at the $(\ell-1)$-th layer.

### 4.2 Multi-View Embedding Extraction

In the second step, we extract multi-view text embeddings, including the conditional text embedding and the unconditional text embedding. For the *conditional text embedding*, we perform standard MHSA computation without modifying the attention mask or positional encoding to obtain the contextualized value vectors (i.e., $\mathbf{V}^\ell = [\mathbf{v}_1^\ell, \cdots, \mathbf{v}_N^\ell]$); for the *unconditional text embedding*, we modify the attention mask and positional encoding in the MHSA module to mask out the conditional part of the input. Then, we describe the technical details of extracting the unconditional text embedding.

**Attention Mask** We first modify the attention mask to prevent the tokens following the condition from seeing the condition, thereby extracting the unconditional text embedding. Since we only focus on the last token for the unconditional text embedding, the attention mask for the last token can be defined as follows:

$$\mathbf{m}_N^\ell(i) = \begin{cases} 0, & 1 \le i \le p-1, \\ -\infty, & p \le i \le q, \\ 0, & q+1 \le i \le N. \end{cases}$$

where $\mathbf{m}_N^\ell(i)$ denotes the attention mask value of the $N$-th token to the $i$-th token in the $\ell$-th layer.

**Positional Encoding** After applying the attention mask, the positional encodings of the text become discontinuous. Specifically, there are no tokens with positional encodings from $p$ to $q$, which can affect the quality of the unconditional text embeddings.

Thus, we further adjust the positional encoding to align with the modified attention mask. Specifically, we shift the positional encodings of all tokens following the condition forward by q-p+1 positions to ensure positional continuity. The new positional encoding function pos$(i)$ is defined as follows:

$$\text{pos}(i) = \begin{cases} i, & 1 \le i \le p-1, \\ 0, & p \le i \le q, \\ i-(q-p+1), & q+1 \le i \le N. \end{cases}$$

Since the condition part (i.e., $p \le i \le q$) is masked, its positional encoding settings do not affect the final results. We simply set them to 0. Notably, after modifying the attention mask and positional encodings, the resulting new prompt forms a coherent PromptEOL-like prompt, which can be used to extract high-quality general unconditional text embeddings.

**Self-Attention Computation** After defining the attention mask and positional encoding, we incorporate positional information into the query and key vectors using RoPE before computing the attention scores:

$$\alpha_{N,i}^{\ell} = \text{softmax}_i\left(\frac{(\mathbf{q}_{\text{pos}(N)}^{\ell})^{\top}\mathbf{k}_{\text{pos}(i)}^{\ell}}{\sqrt{d}} + \mathbf{m}_N^{\ell}(i)\right)$$

where $\alpha_{N,i}^{\ell}$ denotes the attention score from the $N$-th token to the $i$-th token and $\mathbf{q}_{\text{pos}(N)}^{\ell}$ represents the query vector after injecting the new positional encoding function.

Then, we can aggregate the value vectors of all tokens to obtain the unconditional text embedding:

$$\bar{\mathbf{v}}_N^{\ell} = \sum_{i=1}^{N} \alpha_{N,i}^{\ell} \left(\mathbf{W}_V^{\ell}\mathbf{h}_i^{\ell-1}\right) \tag{3}$$

where $\bar{\mathbf{v}}_N^{\ell}$ represents the contextualized value vectors used to encode the unconditional text embedding and $\mathbf{W}_V^{\ell}$ denotes the value projection matrix in the $\ell$-th MHSA layer.

Notably, we define the above process on a single head, and it can be extended to the outputs of multiple heads in MHSA. The overhead of unconditional embedding extraction is very light, as it only requires one additional forward pass through the MHSA layer.

### 4.3 Self-Contrastive Activation Steering

In the final step, it contrasts the two embeddings obtained from itself to purify the conditional embedding, and then continues forward propagation to obtain the final conditional embedding. We propose two alternative self-contrastive steering strategies to obtain the conditional steering vector: direct elimination and projection-based elimination.

**Direct Elimination (DE)** contrasts two embeddings to obtain the conditional steering vector $\Delta\mathbf{v}^{\ell}$. Intuitively, these two embeddings differ only in the conditional part, and comparing them can highlight the conditional information in the prompt. Specifically, this vector is calculated as:

$$\Delta\mathbf{v}_N^{\ell} = \mathbf{v}_N^{\ell} - \bar{\mathbf{v}}_N^{\ell} \tag{4}$$

**Projection-Based Elimination (PE)** first projects the conditional text embedding onto the unconditional text embedding to identify the condition-independent components, and then eliminates them through a contrastive operation. Specifically,

$$\Delta\mathbf{v}_N^{\ell} = \mathbf{v}_N^{\ell} - \frac{(\mathbf{v}_N^{\ell})^{\top}\bar{\mathbf{v}}_N^{\ell}}{\|\bar{\mathbf{v}}_N^{\ell}\|^2}\bar{\mathbf{v}}_N^{\ell} \tag{5}$$

Notably, the conditional steering vector is text-specific, with each text having its own distinct steering vector. In addition, we can also apply steering on the hidden states or the outputs of the FFN.

Since the norms of the vector change significantly before and after the contrast, we introduce a scaling factor $\alpha$ to control the norm of the vector after the intervention.

$$\hat{\mathbf{v}}_N^{\ell} = \alpha \cdot \Delta\mathbf{v}_N^{\ell} \tag{6}$$

After getting refined conditional embedding $\hat{\mathbf{v}}_N^{\ell}$, we further use $\hat{\mathbf{v}}_N^{\ell}$ to replace original conditional text embedding $\mathbf{v}_N^{\ell}$ and obtain the new input $\hat{\mathbf{V}}^{\ell}$ for the output matrix in the MHSA layer. Specifically,

$$\hat{\mathbf{V}}^{\ell} = [\mathbf{v}_1^{\ell}, \cdots, \hat{\mathbf{v}}_N^{\ell}]. \tag{7}$$

Finally, we feed the refined contextualized value vector $\hat{\mathbf{V}}^{\ell}$ into the output matrix $\mathbf{W}_O^{\ell}$ of the $\ell$-th MHSA layer and continue the standard forward propagation to extract conditional text embeddings.

## 5 Experiments

### 5.1 Datasets and Experimental Settings

We evaluate conditional text embeddings on the clustering, STS, and triplet alignment tasks. The clustering task includes 3 datasets. The Tweet emotion intensity dataset (**TweetEmo**) (Mohammad and Bravo-Marquez, 2017) consists of tweets annotated with corresponding emotion labels. **Amazon-R** and **Amazon-C** are constructed from the Amazon English review corpus. Amazon-R uses star ratings as labels, while Amazon-C uses product categories as labels. The STS task includes 2 datasets. InstructSTSB (**I.STSB**) (Peng et al., 2024) extends the original STS-B dataset by incorporating conditions. **C-STS** (Deshpande et al., 2023) is a human-annotated dataset designed to assess the semantic similarity between two sentences under specific conditions. The semantic similarity scores of sentence pairs in the C-STS dataset range from [0, 5], while those in the InstructSTSB dataset range from [0, 1]. The triplet alignment task includes 2 datasets: Toxic and AGNews. For each dataset, we randomly sample 10,000 triplets, where the anchor and the positive share the same label, while the

| Model | Clustering (V-measure ↑) | | | STS (Spearman & Pearson ↑) | | Triplet Alignment (Acc. ↑) | | Mean ↑ |
|---|---|---|---|---|---|---|---|---|
| | TweetEmo | Amazon-R | Amazon-C | I.STSB | C-STS | Toxic | AGNews | |
| *Supervised Contrastive Training* | | | | | | | | |
| **sup-SimCSE$_{\text{RoBERTa-large}}$** | 29.4 | 22.4 | 19.5 | - | 4.1/3.4 | - | - | - |
| **E5$_{\text{Mistral-7B-Inst}}$** | 41.3 | 37.6 | 37.4 | - | 34.8/34.6 | - | - | - |
| **GTE$_{\text{Qwen2-7B-Inst}}$** | 36.8 | 36.8 | 38.3 | - | 33.5/33.9 | - | - | - |
| *Without Contrastive Training* | | | | | | | | |
| **PromptEOL**† | 31.7 | 30.8 | 9.4 | 0/0 | 3.2/2.5 | 55.0 | 71.5 | 22.7 |
| **PonTE**† | 39.9 | 34.8 | 32.5 | 33.9/29.5 | 36.2/32.4 | 57.7 | 84.2 | 42.3 |
| **PonTE + TP**† | 42.6 | 35.4 | 33.6 | 30.7/25.5 | 34.6/31.4 | 56.9 | 82.5 | 41.4 |
| **PonTE + Att**† | 43.4 | 32.4 | 33.2 | 33.3/29.1 | 36.8/32.1 | 57.5 | **85.8** | 42.4 |
| **PonTE + Echo**† | 42.0 | 33.8 | 33.5 | 33.0/28.6 | 35.0/30.8 | 57.8 | 83.1 | 42.0 |
| **PonTE + SCS-DE (*Ours*)** | 45.5 ↑5.6 | 37.0 ↑2.2 | **34.2** ↑1.7 | **39.9/37.7** ↑6.0/8.2 | **37.9**/34.5 ↑1.7/2.1 | **58.0** ↑0.3 | 85.6 ↑1.4 | **45.6** ↑3.3 |
| **PonTE + SCS-PE (*Ours*)** | **45.9** ↑6.0 | **37.1** ↑2.3 | 33.9 ↑1.4 | 38.6/36.0 ↑4.7/6.5 | **37.9/34.6** ↑1.7/2.2 | 57.9 ↑0.2 | **85.8** ↑1.6 | 45.3 ↑3.0 |

Table 1: Results on three downstream tasks across seven datasets using LLaMA3-8B-Inst. Since PonTE does not open-source code, † denotes our reproduced results. The best results are in **bold**, and the second-best are underlined.

negative has a different label. **Toxic** (Adelani et al., 2023) consists of comments from the Civil Comments platform annotated for toxicity, while **AGNews** (Zhang et al., 2015) is an English news classification corpus containing approximately 127,600 samples across four categories.

For the clustering task, we use K-means for clustering and V-measure (Rosenberg and Hirschberg, 2007) to evaluate, with the number of clusters provided in advance. The final result is the average over five runs. For the STS task, we use Spearman and Pearson correlation coefficients. For the triplet alignment task, we report triplet alignment accuracy, defined as the proportion of triplets for which the relative ordering is correctly captured.

We use grid search to search for the intervention layer $\ell$ in [2, 20] and the scaling factor of the norm scaling $\alpha$ in {5, 10, 15} for each dataset. The detailed conditions for clustering and triplet alignment datasets are provided in the Appendix A, and the STS datasets include these conditions.

## 5.2 Baselines

Our method is compared with two groups of baseline methods. The first category is supervised learning methods. **sup-SimCSE$_{\text{RoBERTa-large}}$** (Gao et al., 2021) uses NLI datasets and contrastive loss to fine-tune the model. **E5$_{\text{Mistral-7B-Inst}}$** (Wang et al., 2022) is a two-stage training method: it first performs unsupervised contrastive learning on large-scale datasets, and then fine-tunes on supervised datasets. **GTE$_{\text{Qwen2-7B-Inst}}$** (Li et al., 2023b) further expands the amount of data used in the two stages. The second category is training-free methods. **PromptEOL** (Jiang et al., 2024) is an unconditional prompt-based method that compresses the full semantics of the text into the last token. **PonTE** (Yamada and Zhang, 2025) further incorporates conditions into the prompt to steer the LLM's focus toward specific aspects. **PonTE + TP** (Fu et al., 2025) prepends the text embedding token to allow each token in the text to capture the full semantics. **PonTE + Echo** (Springer et al., 2025) duplicates the text twice to enable the subsequent tokens to capture the complete semantics of the text. **PonTE + Att** introduces an additional hyperparameter to amplify the attention scores of the tokens following the condition toward the condition. The detailed prompts are shown in the Appendix B.

## 5.3 Results

The results of our method on the three tasks are shown in Table 1. Our method achieves the best performance in 6 out of 7 cases and achieves the optimal average performance, confirming the effectiveness of our approach. Our method achieves improvements across all seven datasets. Notably, SCS-DE achieves improvements of 6% and 8.2% on the two metrics of the I.STSB dataset. This is because the data in this dataset are shorter and the similarity labels are divided with finer granularity. Furthermore, the even simpler SCS-DE method achieves better average performance, indicating that the projection may be unnecessary. Besides, our method achieves performance comparable to supervised contrastive learning approaches, demonstrating the potential of directly extracting conditional text embeddings from LLMs.

For the baselines, the unconditional PromptEOL method performs poorly due to the lack of conditions, while PonTE significantly improves PromptEOL's performance by incorporating conditions. PonTE + Att and PonTE + Echo have minimal impact on the final results, whereas PonTE+TP

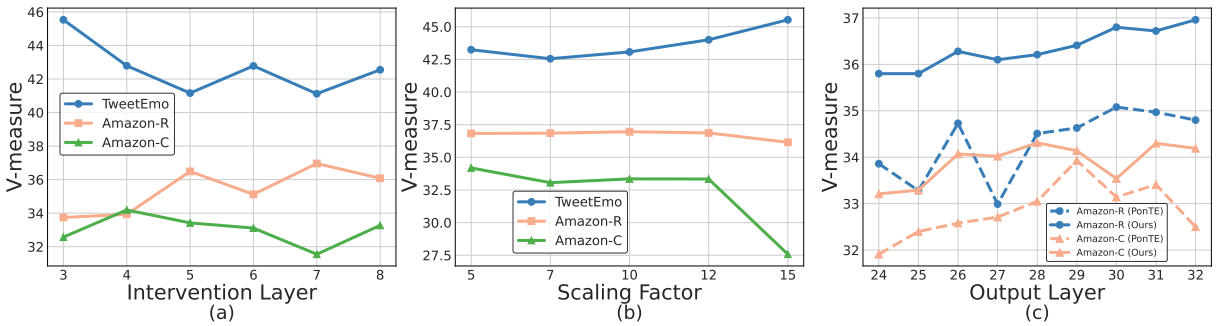



Figure 3: Effects of intervention layer, scaling factor, and output layer on SCS-DE. (a) Effects of the intervention layer. (b) Effects of the scaling factor. (c) The effects of the output layer.

| Method | TweetEmo | Amazon-R |
|---|---|---|
| **PonTE** | 70ms | 79ms |
| **PonTE + SCS-DE (*Ours*)** | 69ms | 80ms |

Table 2: The runtime per sample on two datasets using LLaMA3-8B-Inst.

| Position | I.STSB | TweetEmo | Amazon-R |
|---|---|---|---|
| **Head** | 39.9/37.7 ↑6.0/8.2 | 45.5 ↑5.6 | 37.0 ↑2.2 |
| **FFN** | 40.8/39.1 ↑6.9/9.6 | 44.2 ↑4.3 | 35.7 ↑0.9 |
| **Hidden** | 39.1/37.2 ↑5.2/7.7 | 46.2 ↑6.3 | 35.1 ↑0.3 |

Table 3: Effects of intervention position on three datasets. FFN denotes the output of FFN block, and Hidden denotes the output of the Transformer layer.

causes a significant decline. This may be because instruction-tuned models struggle to comprehend the abstract semantics of the prepended token.

### 5.4 Analysis of Additional Inference Time Overhead

We further compare the time overhead of our method with the baseline methods using a V100 GPU and the same batch size in Table 2. We can see that our method introduces almost no additional time overhead, with the fluctuations mainly due to randomness. This is because our method only computes an additional value vector in a single MHSA layer, while the main computational overhead in LLMs comes from the feed-forward networks.

### 5.5 Model Analysis

Since SCS-DE is simpler and more effective, the subsequent analysis mainly focuses on it.

**Effects of the Intervention Position** We first explore the effects of intervention position on three datasets using LLaMA-3-8B-Inst. Intervening at all three positions can effectively improve performance, indicating the effectiveness of our method, as shown in Table 3. Notably, the optimal intervention positions vary across datasets, which may be determined by the characteristics of each dataset. The FFN and hidden variants incur theoretically higher time overhead, as they need to generate unconditional information within the MHSA module and continue forward propagation for steering. In particular, the hidden variant needs to run through a complete Transformer layer.

We also explore the effects of intervention layer on three datasets in Figure 3(a). For the TweetEmo dataset, the optimal intervention layer is 3. For Amazon-R and Amazon-C datasets, the optimal intervention layers are 7 and 4, respectively. Notably, intervening at the 5-th or 6-th layer generally improves performance across all three datasets.

**Effects of Scaling Factor** We investigate the effects of the scaling factor in LLaMA3-8B-Inst using three datasets, as shown in Figure 3(b). For the TweetEmo dataset, the optimal scaling factor is 15. For Amazon-R and Amazon-C datasets, the optimal scaling factors are 10 and 5, respectively. Compared with PonTE, most scaling factors can achieve performance improvements. When the scaling factor is 7, 10, or 12, all three datasets perform well.

**Effects of Output Layers** We investigate the effects of output layers in LLaMA3-8B-Inst on two datasets, as shown in Figure 3(c). SCS-DE consistently improves PonTE on two datasets. This suggests that our method can enhance the embeddings of all middle and later layers in the LLM, rather than just a single layer. In particular, our method achieves more significant improvements across all later layers on the Amazon-R dataset. In particular, our method changes the optimal layer for conditional text embeddings. On the two datasets, PonTE achieves its best performance at layers 30

| Method | Backbone | I.STSB | TweetEmo | Amazon-R | Mean |
|---|---|---|---|---|---|
| PonTE | LLaMA3-8B | 10.2/9.5 | 26.5 | 23.4 | 17.4 |
| PonTE + SCS-DE (***Ours***) | LLaMA3-8B | 25.9/24.2 ↑15.7/14.7 | 34.8 ↑8.3 | 26.9 ↑3.5 | 28.0 ↑10.6 |
| PonTE | LLaMA3-8B-Inst | 33.9/29.5 | 39.9 | 34.8 | 34.5 |
| PonTE + SCS-DE (***Ours***) | LLaMA3-8B-Inst | 39.9/37.7 ↑6.0/8.2 | 45.5 ↑5.6 | 37.0 ↑2.2 | 40.0 ↑5.5 |
| PonTE | LLaMA3.1-8B | 11.3/11.0 | 26.2 | 23.1 | 17.9 |
| PonTE + SCS-DE (***Ours***) | LLaMA3.1-8B | 25.5/23.1 ↑14.2/12.1 | 38.7 ↑12.5 | 25.9 ↑2.8 | 28.3 ↑10.4 |
| PonTE | LLaMA3.1-8B-Inst | 29.1/24.2 | 40.9 | 34.0 | 32.1 |
| PonTE + SCS-DE (***Ours***) | LLaMA3.1-8B-Inst | 35.9/32.7 ↑6.8/8.5 | 44.8 ↑3.9 | 36.0 ↑2.0 | 37.4 ↑5.3 |
| PonTE | Qwen2.5-7B | 9.3/9.1 | 35.2 | 25.9 | 19.9 |
| PonTE + SCS-DE (***Ours***) | Qwen2.5-7B | 14.5/11.8 ↑5.2/2.7 | 37.7 ↑2.5 | 28.2 ↑2.3 | 23.1 ↑3.2 |
| PonTE | Qwen2.5-7B-Instruct | 27.0/26.3 | 37.2 | 31.6 | 30.5 |
| PonTE + SCS-DE (***Ours***) | Qwen2.5-7B-Instruct | 32.1/31.0 ↑5.1/4.7 | 42.1 ↑4.9 | 32.7 ↑1.1 | 34.5 ↑4.0 |

Table 4: Results on three datasets using different backbones.

| Prompt | I.STSB | TweetEmo | Amazon-R |
|---|---|---|---|
| Express this text "`[TEXT]`" in one word in terms of condition: " | 39.9/37.7 ↑6.0/8.2 | 45.5 ↑5.6 | 37.0 ↑6.2 |
| Express this text "`[TEXT]`" in one word with respect to condition: " | 39.8/38.0 ↑6.8/8.3 | 44.6 ↑1.9 | 35.8 ↑4.5 |
| Express this text "`[TEXT]`" in terms of condition in one word: " | 37.1/33.1 ↑7.8/8.9 | 44.2 ↑2.2 | 37.2 ↑0.4 |
| This text "`[TEXT]`" means in one word in terms of condition: " | 36.5/31.5 ↑6.1/7.3 | 42.5 ↑0.8 | 37.6 ↑0.9 |
| This text "`[TEXT]`" means in one word with respect to condition: " | 31.6/28.3 ↑5.0/6.8 | 42.8 ↑2.6 | 37.2 ↑0.4 |
| This text "`[TEXT]`" means in terms of condition in one word: " | 34.1/29.2 ↑4.2/5.8 | 43.2 ↑1.6 | 37.7 ↑0.1 |

Table 5: Generalization across different prompts using LLaMA3-8B-Inst as the backbone on three datasets.

| Dataset | Condition | V-measure |
|---|---|---|
| **TweetEmo** | **the emotion** | 44.8 ↑3.9 |
| | **the feeling** | 42.4 ↑1.2 |
| | **the sentiment** | 43.5 ↑4.0 |
| **Amazon-R** | **the star** | 32.4 ↑4.7 |
| | **the rating** | 35.7 ↑5.0 |
| | **the star rating** | 37.0 ↑2.2 |

Table 6: Generalization across different conditions using LLaMA3-8B-Inst on two datasets.

and 29, respectively, whereas our method reaches the optimum at layers 32 and 28.

### 5.6 Generalization Analysis

**Generalization across Different LLMs** Table 4 presents the results across various LLMs, including LLaMA3-8B (Dubey et al., 2024), LLaMA3.1-8B (Dubey et al., 2024), Qwen2.5-7B (Yang et al., 2024), and their instruction-tuned versions.

The results demonstrate that our method achieves consistent improvement across various LLMs, highlighting its generalizability. Furthermore, instruction-tuned LLMs generally outperform their non-instruction-tuned counterparts, as our task relies more heavily on following the conditions specified in the prompt. Our method generally achieves greater improvements on LLMs that have not been instruction-tuned, due to their limited instruction-following capability. Additionally, LLaMA3.1-8B does not outperform LLaMA3-8B in terms of conditional text embeddings.

**Generalization across Different Prompts** Table 5 presents the results across various prompts. Our method achieves improvements across six prompts and reduces variance across prompts. This demonstrates the generalizability of our method to different prompts. In addition, we observe that subtle variations in the prompts can have a significant impact on performance, and the optimal prompts differ across datasets.

**Generalization across Different Conditions** Table 6 presents the results across various conditions on two datasets. Our method also achieves consistent improvement across 6 conditions. We can observe that accurate and comprehensive conditions generally lead to better performance. Moreover, the prompts and conditions that were originally optimal are not necessarily the best after steering.

## 6 Conclusion

In this paper, we introduce a plug-and-play, inference-time self-contrastive steering method to extract conditional text embeddings from LLMs without the need for training or additional data. SCS creates multi-view embeddings for a given text by modifying the positional encoding and attention

mask. Subsequently, we propose two alternative self-contrastive activation steering strategies to refine the conditional text embeddings. Extensive experiments on three tasks with their seven associated datasets show that our method can seamlessly improve the performance of existing prompt-based methods across different LLMs in a training-free and plug-and-play manner.

## Limitations

First, our method requires intervening in the internals of LLMs and is therefore not applicable to black-box LLMs. Second, we do not validate our method on larger LLMs, such as 70B models.

## Acknowledgments

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. 62172208, 62441225, 61972192; the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No.JYB2025XDXM118). This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

## References

David Ifeoluwa Adelani, Marek Masiak, Israel Abebe Azime, Jesujoba Alabi, Atnafu Lambebo Tonja, Christine Mwase, Odunayo Ogundepo, Bonaventure F. P. Dossou, Akintunde Oladipo, Doreen Nixdorf, Chris Chinenye Emezue, Sana Al-azzawi, Blessing Sibanda, Davis David, Lolwethu Ndolela, Jonathan Mukiibi, Tunde Ajayi, Tatiana Moteu, Brian Odhiambo, and 46 others. 2023. MasakhaNEWS: News topic classification for African languages. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 144–159, Nusa Dua, Bali.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, and Aitor Gonzalez-Agirre. 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.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 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.

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

Zifeng Cheng, Jinwei Gan, Zhiwei Jiang, Cong Wang, Yafeng Yin, Xiang Luo, Yuchen Fu, and Qing Gu. Steering when necessary: Flexible steering large language models with backtracking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zifeng Cheng, Zhonghui Wang, Yuchen Fu, Zhiwei Jiang, Yafeng Yin, Cong Wang, and Qing Gu. 2025. Contrastive prompting enhances sentence embeddings in llms through inference-time steering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*, pages 3475–3487.

Ameet Deshpande, Carlos E. Jimenez, Howard Chen, Vishvak Murahari, Victoria Graf, Tanmay Rajpurohit, Ashwin Kalyan, Danqi Chen, and Karthik Narasimhan. 2023. C-STS: conditional semantic textual similarity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pages 5669–5690.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Yuchen Fu, Zifeng Cheng, Zhiwei Jiang, Zhonghui Wang, Yafeng Yin, Zhengliang Li, and Qing Gu. 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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2024. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3182–3196.

Ting Jiang, Jian Jiao, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Denvy Deng, and Qi Zhang. 2022. Promptbert: Improving BERT sentence embeddings with prompts. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, pages 8826–8837.

Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. 2024. Meta-task prompting elicits embeddings from large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pages 10141–10157.

Chak Tou Leong, Yi Cheng, Jiashuo Wang, Jian Wang, and Wenjie Li. 2023. Self-detoxifying language models via toxification reversal. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pages 4433–4449.

Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023a. Inference-time intervention: Eliciting truthful answers from a language model. In *Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*.

Xianming Li and Jing Li. 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.

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

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022*, pages 8086–8098.

Saif M Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. *arXiv preprint arXiv:1708.03700*.

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

Letian Peng, Yuwei Zhang, Zilong Wang, Jayanth Srinivasa, Gaowen Liu, Zihan Wang, and Jingbo Shang. 2024. Answer is all you need: Instruction-following text embedding via answering the question. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pages 459–477.

Joris Postmus and Steven Abreu. 2024. Steering large language models using conceptors: Improving addition-based activation engineering. *CoRR*, abs/2410.16314.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. 2024. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pages 15504–15522.

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

Zhenmei Shi, Junyi Wei, Zhuoyan Xu, and Yingyu Liang. 2024. Why larger language models do in-context learning differently? In *Forty-first International Conference on Machine Learning, ICML 2024*.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2025. Repetition improves language model embeddings. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

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

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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

Zihan Wang, Jingbo Shang, and Ruiqi Zhong. 2023. Goal-driven explainable clustering via language descriptions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pages 10626–10649.

Kosuke Yamada and Peinan Zhang. 2025. Out-of-the-box conditional text embeddings from large language models. *CoRR*, abs/2504.16411.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Shufan Yang, Zifeng Cheng, Zhiwei Jiang, Yafeng Yin, Cong Wang, Shiping Ge, Yuchen Fu, and Qing Gu. 2026. Regionmarker: A region-triggered semantic watermarking framework for embedding-as-a-service

copyright protection. In *Fortieth AAAI Conference on Artificial Intelligence, AAAI 2026*, pages 34313–34321.

Young Hyun Yoo, Jii Cha, Changhyeon Kim, and Taeuk Kim. 2024. Hyper-cl: Conditioning sentence representations with hypernetworks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pages 700–711.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 649–657.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2023. Representation engineering: A top-down approach to AI transparency. *CoRR*, abs/2310.01405.

## A Prompts of Our Method

Since the conditions for I.STSB and C-STS are specified by the datasets, we report the detailed prompts for the remaining datasets as follows:

**TweetEmo**: Express this text "`[TEXT]`" in one word in terms of `the emotion`: "
**Amazon-R**: Express this text "`[TEXT]`" in one word in terms of `the star rating`: "
**Amazon-C**: Express this text "`[TEXT]`" in one word in terms of `the product category`: "
**Toxic**: Express this text "`[TEXT]`" in one word in terms of `toxicity`: "
**AGNews**: Express this text "`[TEXT]`" in one word in terms of `the topic category of this news`: "

## B Baselines

We report the detailed prompts for the baselines as follows:

**PromptEOL:** This sentence: "`[TEXT]`" means in one word: "
**PonTE**: Express this text "`[TEXT]`" in one word in terms of `[Condition]`: "
**PonTE+Echo**: Express this text "`[TEXT] [TEXT]`" in one word in terms of `[Condition]`: "

## C Results of Multi-layer Intervention

Table 8 presents the results of our method under multi-layer intervention.

We observe that single-layer intervention often achieves the best or second-best performance, indicating that intervening on only one layer is sufficient to yield strong performance improvements. In addition, single-layer intervention involves only one hyperparameter and introduces much lower additional time overhead.

## D Similarity Measurement Before and After Steering

We first measure the embedding similarity between PromptEOL and PonTE at layer $\ell$ in Table 8, i.e., the cosine similarity of contextualized value vectors before steering. We observe that even after introducing conditions, the cosine similarity of contextualized value vectors obtained under the two prompts remains very high. This indicates that PonTE still encodes a substantial amount of general semantic information.

After steering, we measure the embedding similarity again and observe a clear decrease. This demonstrates that our method can effectively remove general information from the embeddings, making them more focused on the condition.

## E Results on Other Languages

Table 7 shows the performance of our method on other languages in the Amazon-R dataset. Our method still achieves performance gains in German and Spanish, demonstrating its cross-lingual generalization capability.

## F Results on Larger LLMs

Table 10 presents the performance of our method on larger LLMs.

Our method also achieves performance improvements on larger LLMs, which further demonstrates its generalization ability. Notably, for conditional text embeddings, larger LLMs do not always yield better performance, a finding that has also been observed in previous studies (Lei et al., 2024; Fu et al., 2025).

## G Case Study

Table 11 presents the top-8 tokens decoded by different methods.

| Method | German | Spanish |
|---|---|---|
| **PonTE** | 33.0 | 29.0 |
| **PonTE + SCS-DE (*Ours*)** | 35.8 ↑2.8 | 31.6 ↑2.6 |

Table 7: Results on the two languages of the Amazon-R dataset.

| Dataset | Before Steering | After Steering |
|---|---|---|
| I.STSB | 0.95 | -0.17 |
| TweetEmo | 0.94 | -0.52 |
| Amazon-R | 0.96 | -0.27 |

Table 8: Similarity before and after steering on three datasets.

The tokens decoded by PonTE include some information that is irrelevant to the instruction, such as "I" and "Family" in the first text, and "Rows" and "Don" in the second text. The tokens decoded by our method are all relevant to the instruction, and their probability distribution is more concentrated.

| Layer Index | TweetEmo | Amazon-R |
|---|---|---|
| *Intervening on a single layer:* | | |
| 1 | 43.6 | 34.5 |
| 2 | 43.9 | 33.8 |
| 3 | **45.5** | 35.4 |
| 4 | 43.3 | 35.7 |
| 5 | 43.9 | 36.5 |
| 6 | 43.9 | 35.8 |
| 7 | 42.7 | **37.0** |
| 8 | 44.9 | 36.1 |
| *Intervening on two layers:* | | |
| 1-2 | 41.9 | 31.9 |
| 2-3 | **44.0** | 35.1 |
| 3-4 | 43.3 | 36.3 |
| 4-5 | 43.3 | 35.9 |
| 5-6 | 43.8 | **37.0** |
| 6-7 | 42.7 | 36.2 |
| 7-8 | 42.9 | 36.9 |
| *Intervening on three layers:* | | |
| 1-3 | 44.0 | 33.3 |
| 2-4 | **44.3** | 34.6 |
| 3-5 | 43.0 | 35.4 |
| 4-6 | 43.5 | 36.4 |
| 5-7 | 43.7 | 36.5 |
| 6-8 | 43.3 | **36.8** |
| *Intervening on four layers:* | | |
| 1-4 | 42.4 | 33.0 |
| 2-5 | 44.4 | **37.0** |
| 3-6 | 43.1 | 35.4 |
| 4-7 | **44.9** | 36.3 |
| 5-8 | 43.8 | 36.8 |
| *Intervening on five layers:* | | |
| 1-5 | 44.3 | 31.7 |
| 2-6 | **45.8** | 34.3 |
| 3-7 | 43.3 | 36.2 |
| 4-8 | 43.6 | **36.8** |

Table 9: Results of applying multi-layer interventions on the two datasets.

| Method | Backbone | I.STSB | TweetEmo |
|---|---|---|---|
| PonTE | Qwen2.5-14B | 13.5/10.9 | 17.4 |
| PonTE + SCS-DE (***Ours***) | Qwen2.5-14B | 20.2/16.2 ↑6.6/5.3 | 27.0 ↑9.6 |
| PonTE | Qwen2.5-14B-Instruct | 24.7/21.3 | 22.5 |
| PonTE + SCS-DE (***Ours***) | Qwen2.5-14B-Instruct | 28.9/27.5 ↑4.2/5.2 | 43.1 ↑20.6 |

Table 10: Results on three datasets using different backbones.

| Text | Condition | Method | Top-predicted Tokens and Probability |
|---|---|---|---|
| A man and two children lounge in the living area by a sliding glass door. | the number of people | PonTE | Three (0.482), Four (0.054), three (0.046), I (0.038), TH (0.032), Family (0.03), T (0.029), Two (0.029) |
| | | PonTE+SCS-DE | Three (0.550), three (0.056), Four (0.049), Two (0.037) TH (0.028), 3 (0.028), T (0.026), Tri (0.024) |
| A row of green donuts, a row of glazed donuts, and a row of chocolate donuts. | the rows | PonTE | Rows (0.131), Tr (0.107), Tri (0.105), tr (0.067), Don (0.056), T (0.052), rows (0.05), TR (0.043) |
| | | PonTE+SCS-DE | Three (0.423), Tri (0.073), three (0.070), Rows (0.068) TR (0.038), Tr (0.033), TH (0.032), tr (0.025) |

Table 11: Top-8 tokens predicted by different methods using LLaMA3-8B-Inst.