

Token Prepending: A Training-Free Approach for Eliciting Better Sentence Embeddings from LLMs

Yuchen Fu* Zifeng Cheng* Zhiwei Jiang[†] Zhonghui Wang
Yafeng Yin Zhengliang Li Qing Gu

State Key Laboratory for Novel Software Technology, Nanjing University, China

{yuchenfu,chengzf}@smail.nju.edu.cn, jzw@nju.edu.cn

zhonghuiwang@smail.nju.edu.cn, yafeng@nju.edu.cn

lzl@smail.nju.edu.cn, guq@nju.edu.cn

Abstract

Extracting sentence embeddings from large language models (LLMs) is a promising direction, as LLMs have demonstrated stronger semantic understanding capabilities. Previous studies typically focus on prompt engineering to elicit sentence embeddings from LLMs by prompting the model to encode sentence information into the embedding of the last token. However, LLMs are mostly decoder-only models with causal attention and the earlier tokens in the sentence cannot attend to the latter tokens, resulting in biased encoding of sentence information and cascading effects on the final decoded token. To this end, we propose a novel Token Prepending (TP) technique that prepends each layer’s decoded sentence embedding to the beginning of the sentence in the next layer’s input, allowing earlier tokens to attend to the complete sentence information under the causal attention mechanism. The proposed TP technique is a plug-and-play and training-free technique, which means it can be seamlessly integrated with various prompt-based sentence embedding methods and autoregressive LLMs. Extensive experiments on various Semantic Textual Similarity (STS) tasks and downstream classification tasks demonstrate that our proposed TP technique can significantly improve the performance of existing prompt-based sentence embedding methods across different LLMs, while incurring negligible additional inference cost. The code are available on https://github.com/fuyuchenIfyw/token_prepending.git.

1 Introduction

Sentence embeddings have a wide range of applications in real-world scenarios, such as information retrieval, recommender systems, sentiment analysis, document clustering, and so on. Recently,

*Equal Contribution.

[†]Corresponding Author.

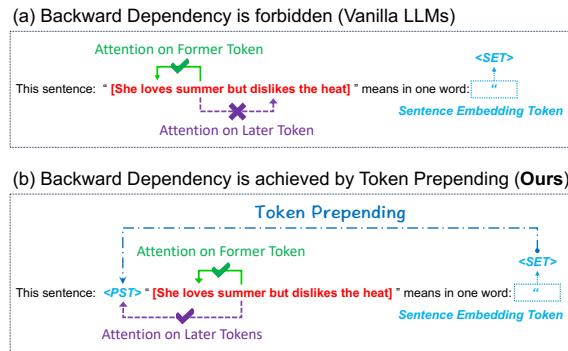


Figure 1: Comparison between (a) vanilla LLMs and (b) our proposed LLMs with token prepending.

with the success of large language models (LLMs) in zero-shot settings for various natural language processing (NLP) tasks, some researchers have begun to focus on directly extracting sentence embeddings from LLMs without the need for additional fine-tuning (Liu et al., 2024a; Lei et al., 2024). This training-free setup is both practical and promising, as it does not require training data, avoids the costs of fine-tuning a large-scale model, and prevents the potential loss of general semantic understanding capabilities caused by fine-tuning on specific data.

Different from previous encoder-only bidirectional language model like BERT (Devlin et al., 2019), current LLMs are mostly decoder-only models with causal attention (Touvron et al., 2023; Brown, 2020), which make the earlier tokens in the sentence cannot attend to the latter tokens, as shown in Figure 1(a). To this end, recent studies (Jiang et al., 2023; Lei et al., 2024; Zhang et al., 2024) attempt to prompt the model to encode sentence information into the embedding of the last token (i.e., the <SET> in Figure 1(a)), which can attend to all preceding tokens, thereby avoiding the problem of backward dependency. Among the prompt-based methods, Jiang et al. (2023) first propose to use a simple and effective prompt (e.g., the prompt in Figure 1(a)) to extract sentence embed-

dings from LLMs. Later, meta-task prompts (Lei et al., 2024) and prompts with CoT and Knowledge Enhancement (Zhang et al., 2024) are employed to extract sentence embeddings.

However, even if the last token is able to attend to all tokens in the sentence under the causal attention mechanism, the earlier tokens in the sentence still cannot attend to the later tokens (i.e., the backward dependency in Figure 1). This results in biased encoding of sentence information and cascading effects on the last token. To address this problem, some previous work (Springer et al., 2024) has attempted to achieve backward dependency through repetition. They point out that processing the input twice allows LLMs to have a deeper understanding of the sentence and improves performance on various tasks. Nonetheless, repetition significantly increases the sequence length and substantially alters the sentence structure, leading to higher inference costs and less ideal performance.

In this paper, we propose a simple yet effective technique called Token Prepending (TP). As shown in Figure 1(b), our core idea is to prepends each layer’s decoded sentence embedding to the beginning of the sentence in the next layer’s input, allowing earlier tokens to attend to the complete sentence information under the causal attention mechanism. Notably, the TP technique is entirely training-free, as it introduces no additional learnable parameters. Specifically, although TP is applicable to all layers, we find that it is not necessary to perform the TP operation across all layers. Instead, performing this operation only in the early layers of the model yields better performance. Therefore, we discontinue the TP operation after several early layers and revert to standard forward propagation. Additionally, considering that the final layer of LLMs is primarily used for token generation and contains less semantic information (Liu et al., 2024b; Jin et al., 2024b), we propose an early-exit strategy that outputs embeddings from intermediate layers, rather than the final layer, to serve as sentence embeddings.

Our main contributions are as follows:

- We propose a novel TP technique for eliciting sentence embeddings from LLMs. This plug-and-play technique neither introduces new parameters nor alters the existing ones, allowing it to be seamlessly integrated with various prompt-based sentence embedding methods and autoregressive LLMs. Moreover, it adds only a single token to the original sentence,

resulting in minimal additional inference overhead.

- We perform an in-depth exploration of the TP technique and identify the most effective ways to utilize it, including the optimal layer scope of operation and the early exit strategy.
- We conduct extensive experiments on various Semantic Textual Similarity (STS) benchmarks and downstream classification tasks. The results demonstrate that our proposed TP technique can significantly improve the performance of existing prompt-based sentence embedding methods across different LLMs.

2 Related Work

Sentence Embeddings Sentence embedding is a fundamental task in natural language processing, aiming to map the semantic information of sentences into fixed-size vector representations. Previous research often employs unsupervised or supervised contrastive learning to fine-tune smaller pre-trained models to enhance sentence embeddings (Gao et al., 2021; Jiang et al., 2022; Ni et al., 2022b; Chanchani and Huang, 2023; Su et al., 2023). For example, Sentence-T5 (Ni et al., 2022b) explores three strategies to extract T5 (Raffel et al., 2020) sentence representations and uses two-stage training to refine T5 sentence embeddings. Unlike these methods, we focus on sentence embeddings extracted by large language models without the need for fine-tuning.

LLMs for Sentence Embeddings Recently, a series of studies focus on enhancing the sentence embedding of LLMs with causal attention mechanism through fine-tuning (Li and Li, 2024; BehnamGhader et al., 2024; Lee et al., 2024; Muenninghoff et al., 2024). Due to the limited representation learning capability of unidirectional attention in LLMs, these methods mostly replace it with bidirectional attention and fine-tune LLMs using contrastive learning. For example, BeLLM (Li and Li, 2024) converts the last attention layer from unidirectional to bidirectional and uses SimCSE (Gao et al., 2021) to fine-tune LLMs. However, fine-tuning LLMs is very expensive and inevitably results in the loss of their other general capabilities. Thus, this paper focuses on extracting sentence embeddings from LLMs without fine-tuning.

Extracting Sentence Embeddings from LLMs Existing methods on extracting sentence

embeddings from LLMs mainly focus on designing prompts to improve sentence embeddings. PromptEOL (Jiang et al., 2023) demonstrates the potential of LLMs in generating sentence embeddings by leveraging prompt engineering. Echo embeddings (Springer et al., 2024) repeats the input twice within the context and extracts embeddings from the second occurrence, allowing early token embeddings to encode information about subsequent tokens. MetaEOL (Lei et al., 2024) designs meta-task prompts via ChatGPT-4 to guide LLMs to consider sentence representations from multiple perspectives. Pretended CoT (Zhang et al., 2024) uses CoT to inspire the model to output better embeddings. Knowledge Enhancement (Zhang et al., 2024) provides explicit guidance to the model by conveying human experience in text summarization through prompts. CP (Cheng et al., 2025) introduces an extra auxiliary prompt to elicit better sentence embedding. In this paper, we propose a plug-and-play technique TP to improve the various prompt-based methods with negligible additional inference cost.

3 Preliminary

Previous work mainly focused on eliciting sentence embeddings from LLMs through prompt engineering. This process does not interfere with the internal operations of the LLMs but simply guides their behavior through different prompts. As shown in Figure 2(a), PromptEOL (Jiang et al., 2023) introduces a widely adopted template for extracting sentence embeddings from LLMs:

This sentence: “[Text]” means in one word: “

where [Text] denotes the placeholder for the input sentence and the last token “ is used to decode the Sentence Embedding Token (SET). The phrase “in one word” is a constraint that can prevent LLMs from generating long sentences, limiting a sentence to being represented by the embedding of a single word.

Formally, given the input $T = [t_1, \dots, t_n]$ wrapped in a template, we first obtain the embeddings $\mathbf{h}^0 = [h_1^0, \dots, h_n^0]$ through the embedding layer, and then pass them into the L Transformer layers of LLMs. The previous work (e.g., PromptEOL) use the last layer’s hidden state for the sentence embedding token \mathbf{h}_n^L as the output sentence embedding. Specifically,

$$\mathbf{h}^L = \text{LLM}^{1:L}(\mathbf{h}^0)$$

4 Proposed Method

4.1 Overview

Different from previous work that only focuses on prompt engineering, our proposed method slightly intervenes in the internal operations of the LLMs. Our core idea is to prepend the decoded sentence embedding token from the previous layer to the sentence in the next layer’s input, making the semantics of the sentence perceptible to all tokens in the target sentence. As shown in Figure 2(b), we perform the token prepending (TP) operation within the layer scope of the first few layers, which is denoted in yellow. For the input layer, we prepend a special <PST> token to the input sentence (i.e., [Text]) in prompt. For the intermediate layers, we perform the TP operation between two layers by replacing the embedding of the <PST> token with the sentence embedding decoded from the last token of prompt. By repeating this operation across several layers, the embedding of the <PST> token may contain sufficient sentence information, or all tokens of the target sentence may perceive enough sentence information. After that, we will discontinue the TP operation. Finally, considering that the last layer of LLMs is primarily used for token generation, we will choose a sentence embedding from an intermediate layer as the output sentence embedding.

4.2 Token Prepending

Our proposed TP technique is a plug-and-play operation primarily used to adjust context dependency by intervening in the inputs of LLM layers. From the perspective of its operating layer, it can be described in detail from the following three aspects.

4.2.1 Initial Token Prepending

We first conduct initial token prepending operation that prepends the sentence embedding token to the input text as shown in Figure 2(b). Since the sentence embedding token is not available at this stage, we prepend a custom token “<PST>”, which is not included in the LLM’s vocabulary, serving as a placeholder for sentence embedding token. We randomly initialize the parameters of this token and incorporate it into the input for the first Transformer layer. Consequently, the modified embedding layer output is denoted as $\mathbf{h}^0 = [h_1^0, \dots, h_{i-1}^0, h_i^0, h_i^*, h_i^0, \dots, h_n^0]$, where h_i^* represents the initialized embedding of the <PST> token.

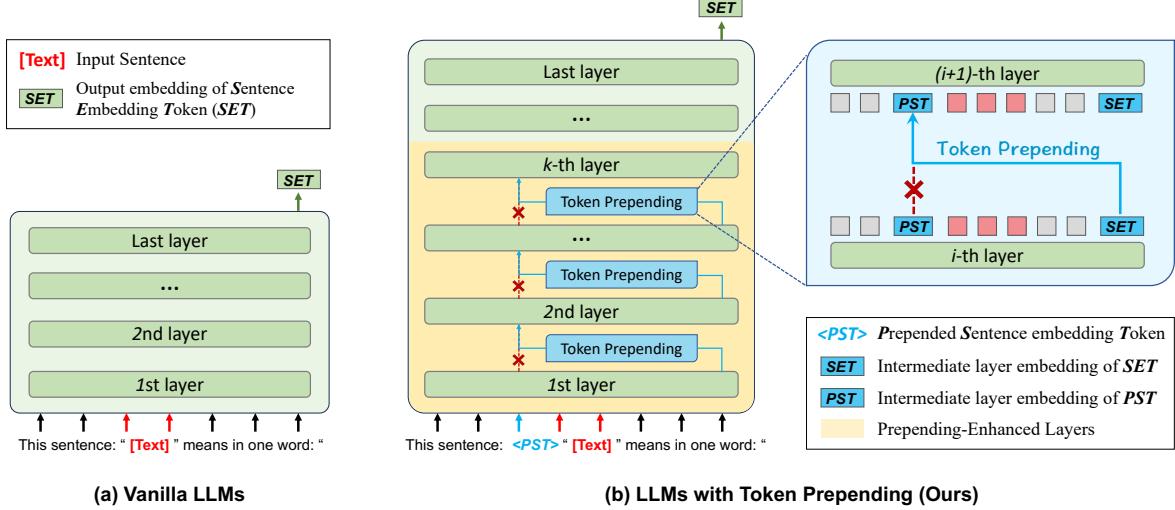


Figure 2: Illustration of extracting sentence embeddings from (a) vanilla LLMs and (b) LLMs with Token Prepending.

4.2.2 Intermediate Token Prepending

After initial token prepending, the input passes through the prepending-enhanced layers, where each layer consists of a standard Transformer layer and a specially designed intermediate token prepending. For intermediate token prepending, we pretend the sentence embedding token **<SET>** to replace **<PST>** as input to the subsequent layer. Prepending the **<SET>** aims to refine the sentence embedding so that subsequent tokens can better capture the sentence’s semantics. This procedure is formalized as follows:

$$\begin{aligned} \mathbf{h}^l &= \text{LLM}^{l-1}(f(\mathbf{h}^{l-1})), \quad l \in [2, k] \\ \mathbf{h}^{l-1} &= [h_1^{l-1}, \dots, h_{i^*}^{l-1}, \dots, h_n^{l-1}], l \in [2, k] \\ f(\mathbf{h}^{l-1}) &= [h_1^{l-1}, \dots, h_n^{l-1}, \dots, h_n^{l-1}], l \in [2, k] \end{aligned}$$

where $f(\mathbf{h})$ represents the function that operates on \mathbf{h} . $k \in [2, L]$ denotes the ending layer for the intermediate token prepending and i^* is the position index of the **<PST>** token.

4.2.3 Layer Scope for Token Prepending

After passing through the prepending-enhanced layers, all tokens in the sentence are contextualized and can perceive the complete semantic meaning of the sentence. Therefore, we do not use intermediate token prepending in the later layers and directly feed the hidden states into the standard Transformer layers of LLMs to obtain the sentence embedding. Specifically,

$$\mathbf{h}^{l+1} = \text{LLM}^l(\mathbf{h}^l), l \in [k, M]$$

where M is the exit layer, which can be either an intermediate layer or the last layer of the LLM.

4.3 Early-Exit from Intermediate Layers

Recent studies (Liu et al., 2024c; Jin et al., 2024a) demonstrate that each layer of LLMs plays a different role, and the embeddings from the last layer are primarily used for prediction and contain weaker semantic information. Thus, we propose the early-exit strategy, which uses embeddings from intermediate layers instead of the last layer to serve as sentence embeddings. We use the validation set to determine which layer of embedding to use, and the overhead of this process is light. Another advantage of the early-exit strategy is that we can obtain sentence embeddings more quickly during the testing phase.

5 Experiments

5.1 Datasets and Experimental Settings

We evaluate sentence embeddings on seven semantic textual similarity (STS) datasets, 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 the STS datasets is annotated with a pairwise semantic similarity score from 0 to 5. We use Spearman correlation as evaluation metric, which measures the rank correlation between the predicted similarity scores and annotated similarity scores using a monotonic function. We use cosine similarity to compute the predicted similarity scores.

Unless otherwise specified, we use the STS-B development set to determine hyperparameters for TP across all prompt and backbone configurations. In all prompts, the placeholder token **<PST>** is placed

Method	Params	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.	Time
BERT avg	110M	30.87	59.89	47.73	60.29	63.73	47.29	58.22	52.57	-
BERT prompt	110M	60.96	73.83	62.18	71.54	68.68	70.60	67.16	67.85	-
ST5-Enc avg	4.8B	34.97	60.19	47.59	66.40	70.62	62.83	63.57	58.02	-
LLaMA2 avg	7B	35.49	53.15	40.12	55.35	53.26	42.10	49.96	47.06	1.00×
LLaMA2 echo	7B	52.40	72.40	61.24	72.67	73.51	65.73	64.39	66.05	1.67×
PromptEOL	7B	58.81	77.01	66.34	73.22	73.56	71.66	69.64	70.03	1.00×
PromptEOL + TP (<i>Ours</i>)	7B	66.90 \uparrow 8.09	83.12 \uparrow 6.11	74.31 \uparrow 7.97	79.87 \uparrow 6.65	80.03 \uparrow 6.47	80.67 \uparrow 9.01	75.40 \uparrow 5.76	77.19 \uparrow 7.16	1.04×
MetaEOL	7B	64.16	81.61	73.09	81.11	78.94	77.96	74.86	75.96	8.17×
MetaEOL + TP (<i>Ours</i>)	7B	66.15 \uparrow 1.99	82.37 \uparrow 0.76	74.89 \uparrow 1.80	83.77 \uparrow 2.66	81.49 \uparrow 2.55	81.46 \uparrow 3.50	75.27 \uparrow 0.41	77.91 \uparrow 1.95	8.29×
Pretended CoT	7B	67.45	83.89	74.14	79.47	80.76	78.95	73.33	76.86	1.18×
Pretended CoT + TP (<i>Ours</i>)	7B	68.52 \uparrow 1.07	83.44 \downarrow 0.45	75.23 \uparrow 1.09	79.36 \downarrow 0.11	81.33 \uparrow 0.57	80.37 \uparrow 1.42	74.51 \uparrow 1.18	77.54 \uparrow 0.68	1.20×
Knowledge	7B	65.60	82.82	74.48	80.75	80.13	80.34	75.89	77.14	1.17×
Knowledge + TP (<i>Ours</i>)	7B	66.03 \uparrow 0.43	83.43 \uparrow 0.61	74.50 \uparrow 0.02	80.94 \uparrow 0.19	81.28 \uparrow 1.15	80.45 \uparrow 0.11	76.13 \uparrow 0.24	77.54 \uparrow 0.40	1.20%

Table 1: Results on STS tasks (Spearman correlation scaled by 100x) using LLaMA2-7B as backbone. The Time column refers to the ratio of inference time for various prompt methods relative to PromptEOL on the STS-B test dataset, ensuring the same output layer.

before “[Text]” in the template. We use the output from the 27-th layer for PromptEOL, MetaEOL, and Pretended CoT, and from the penultimate layer for Knowledge Enhancement. After the 8-th layer, we do not perform token prepending.

5.2 Baselines

We combine our method with some baselines to demonstrate effectiveness. **BERT avg** (Devlin et al., 2019), **ST5-Enc avg** (Ni et al., 2022a), and **LLaMA2 avg** (Touvron et al., 2023) average token embeddings to obtain sentence embeddings using different backbones. **LLaMA2 echo** (Springer et al., 2024) utilizes the strategy of repetition to obtain sentence embeddings. **BERT prompt** (Jiang et al., 2022) proposes a simple and effective prompt to extract sentence embeddings from BERT. **PromptEOL** (Jiang et al., 2023) first proposes a simple and effective prompt to extract sentence embeddings from LLMs. **MetaEOL** (Lei et al., 2024) leverages a diverse set of meta-task prompts to capture multiple representations of sentences from distinct perspectives. **Pretended CoT** (Zhang et al., 2024) uses CoT to inspire the model to extract sentence embeddings. **Knowledge** (Zhang et al., 2024) explicitly infuses the model with human insights into text summarization.

5.3 Main Results

The results of our method on the STS tasks are presented in Table 1. Our method consistently outperforms all baselines and non-prompt-based methods perform worse than prompt-based ones. Among all prompt-based methods across all datasets on LLaMA2-7B, our model shows improvement in 26 out of 28 cases. This shows our method can be

seamlessly integrated with various prompt-based methods without training. Notably, our method achieves the most significant improvement with PromptEOL, enhancing performance by 7.16.

The significant improvement in PromptEOL may be because the other three baselines incorporate prior knowledge to understand sentences, whereas PromptEOL relies more on modeling backward dependency to grasp semantics. Moreover, our method effectively narrows the performance gap between different prompts, improving the model’s robustness to prompts.

Another advantage of TP technology is that it introduces minimal additional inference time compare to prompt-based methods. We evaluate the inference time by running the LLaMA2-7B model on the STS-Benchmark test dataset, fixing the batch size to 1. To mitigate the impact of repeatedly loading the prompt prefix, we employ KV cache. The comparison results are shown in the Time column of Table 1. We observe that Pretended CoT, Knowledge Enhancement, and MetaEOL incur 1.18, 1.17 and 8.17 times the inference time of PromptEOL, respectively. In contrast, the inference time of prompt-based methods with TP technology is within 1.04 times of the original, adding negligible overhead.

5.4 Evaluation of Different Backbones

Table 2 highlights the performance across various model backbones. In addition to the 7B and 13B versions of LLaMA2 , we evaluate our method on several state-of-the-art decoder-only large language models, including Qwen2-7B (Yang et al., 2024), LLaMA3-8B (Dubey et al., 2024), and Gemma2-9B (Team et al., 2024), using Pretended CoT as the prompt template.

Method	Backbone	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Pretended CoT	LLaMA2-7B	67.45	83.89	74.14	79.47	80.76	78.95	73.33	76.86
Pretended CoT + TP (<i>Ours</i>)	LLaMA2-7B	68.52 \uparrow 1.07	83.44 \downarrow 0.45	75.23 \uparrow 1.09	79.36 \downarrow 0.11	81.33 \uparrow 0.57	80.37 \uparrow 1.42	74.51 \uparrow 1.18	77.54 \uparrow 0.68
Pretended CoT	LLaMA2-13B	64.27	78.61	69.93	76.37	79.28	75.88	69.04	73.34
Pretended CoT + TP (<i>Ours</i>)	LLaMA2-13B	65.65 \uparrow 1.38	79.50 \uparrow 0.89	71.01 \uparrow 1.08	77.27 \uparrow 0.90	80.07 \uparrow 0.79	77.36 \uparrow 1.48	71.51 \uparrow 2.47	74.62 \uparrow 1.28
Pretended CoT	LLaMA3-8B	66.65	82.60	72.40	79.36	80.86	77.09	73.66	76.09
Pretended CoT + TP (<i>Ours</i>)	LLaMA3-8B	66.94 \uparrow 0.29	83.20 \uparrow 0.60	73.33 \uparrow 0.93	79.81 \uparrow 0.45	81.72 \uparrow 0.86	78.46 \uparrow 1.37	73.99 \uparrow 0.33	76.78 \uparrow 0.69
Pretended CoT	Qwen2-7B	61.64	78.24	70.14	74.44	76.63	76.22	73.30	72.94
Pretended CoT + TP (<i>Ours</i>)	Qwen2-7B	65.02 \uparrow 3.38	79.50 \uparrow 1.26	71.64 \uparrow 1.50	77.94 \uparrow 3.5	79.15 \uparrow 2.52	78.47 \uparrow 2.25	74.05 \uparrow 0.75	75.11 \uparrow 2.17
Pretended CoT	Gemma2-9B	69.50	82.71	74.18	79.64	80.60	78.89	73.60	77.02
Pretended CoT + TP (<i>Ours</i>)	Gemma2-9B	69.48 \downarrow 0.02	83.39 \uparrow 0.68	74.32 \uparrow 0.14	80.71 \uparrow 1.07	81.24 \uparrow 0.64	79.24 \uparrow 0.35	74.26 \uparrow 0.66	77.52 \uparrow 0.50

Table 2: Results on STS tasks (Spearman correlation scaled by 100x) using different backbones. Since MetaEOL uses multiple prompts, we use the simple and effective Pretended CoT for our experiments.

Prompt Template	STS Avg.
This sentence : <PST> “[Text]” means in one word: “	77.19
<PST> This sentence : “[Text]” means in one word: “	76.35
This sentence : “<PST> [Text]” means in one word: “	76.71
This sentence : “[Text]” <PST> means in one word: “	75.54
After thinking step by step , this sentence : <PST> “[Text]” means in one word: “	77.54
After thinking step by step , <PST> this sentence : “[Text]” means in one word: “	77.81
After thinking step by step , this sentence : “<PST> [Text]” means in one word: “	77.51
After thinking step by step , this sentence : “[Text]” <PST> means in one word: “	77.44

Table 3: Influence of the <PST> token’s position in the sentence. We use PromptEOL and Pretended CoT as the prompt.

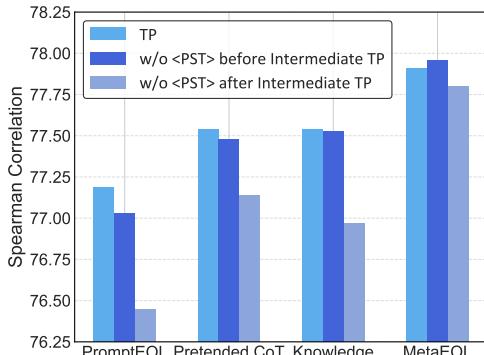


Figure 3: Ablation of <PST> token before and after intermediate token prepending.

The results demonstrate that our method adapts effectively to a range of large language models, delivering performance gains across different backbones. Notably, on the Qwen2-7B model, our model achieves an improvement of 2.17 points. In addition, LLaMA2-13B and LLaMA3-8B do not achieve better performance than LLaMA2-7B.

5.5 Analysis of <PST> Token

In this section, we analyze the prepended <PST> token in detail using LLaMA2-7B.

Effects of the position of <PST> We employ PromptEOL and Pretended CoT as the prompt template to examine how the placement of the <PST>

token at different positions in the sentence affects performance on STS tasks, as shown in Table 3. The performance is worst when <PST> token is inserted right after the input text. When <PST> token is placed before the text, performance fluctuation is small. The optimal position of <PST> token varies depending on the prompt, typically positioning it close to the text. To avoid the additional overhead of searching the position, we simply place <PST> token after the colon for all prompts.

Effectiveness of retaining the <PST> token before and after intermediate TP We ablate the <PST> token before and after intermediate token prepending to show the effectiveness of retaining the <PST> token. Ablating the <PST> token before the intermediate token prepending is equivalent to removing the initial token prepending and directly performing intermediate token prepending.

The results as shown in Figure 3. Without <PST> before intermediate token prepending generally results in a slight decrease in performance across most prompts. This is because the initial <PST> token does not carry semantic information, and its main purpose is to maintain the same input sequence length across all layers. In contrast, ablating the <PST> token after intermediate token prepending has a more pronounced negative impact. This may be because the representations are

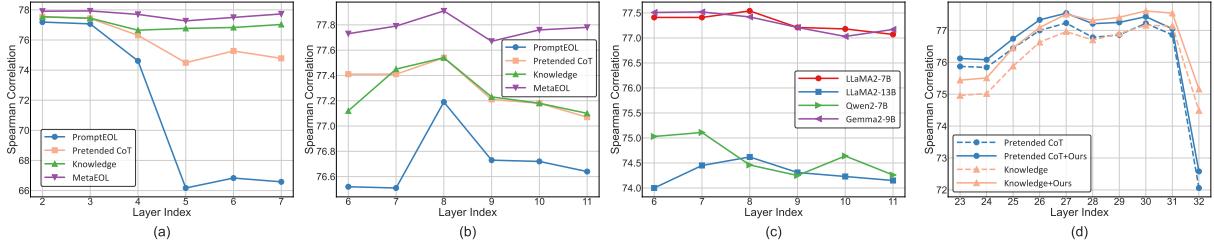


Figure 4: Effects of layer scope for intermediate token prepending and early-exit layer. The reported Spearman correlation is the average across the seven STS tasks. (a) The influence of start layer for the intermediate token prepending. (b) The influence of end layer k for the intermediate token prepending. (c) The token prepending ending layer k on different backbones. (d) The influence of exit layer M for sentence embeddings.

Initialization Method	STS Avg.
All 0	77.54
All 1	77.54
Uniform	77.53
Gaussian	77.54
Existing token	77.55

Table 4: Influence of the <PST> token’s initialization method.

aligned with this input pattern, and modifying the input could lead to a decrease in performance.

Influence of <PST> token initialization We use Pretended CoT to investigate the effect of various initialization methods for the <PST> token parameters in the embedding layer. We evaluate five initialization techniques: all 0, all 1, uniform distribution within the range [0,1], Gaussian distribution, and using existing token parameters. For the existing token initialization, we select the embedding of the space character, allowing the model to interpret the <PST> token as a space, thereby minimizing its impact on the whole sentence’s meaning.

As shown in Table 4, the variation among different initialization is minimal, with the maximum performance difference in STS tasks being just 0.01. This suggests that our method remains robust regardless of the <PST> token’s initialization.

5.6 Analysis of Layer Scope for TP

In this section, we analyze the effects of layer scope for token prepending in detail.

Influence of start and end layer for TP In Figure 4(a) and (b), we explore the impact of the start and end layer for intermediate token prepending on LLaMA2-7B.

As illustrated in Figure 4(a), performance is suboptimal if intermediate token prepending does not begin at the second layer. This is because the <PST> token is randomly initialized and lacks se-

mantic information. Thus, we need to replace it with semantically meaningful tokens at the early layers of the LLMs to mitigate this issue.

Furthermore, our results indicate that halting token prepending after the 8-th layer yields the best performance across all prompts used.

Influence of end layer for intermediate TP on different backbones We further examine the optimal layer to terminate token prepending across different backbones. As shown in Figure 4(c), for the LLaMA2-7B and LLaMA2-13B models, stopping token prepending at the 8-th yields the best performance. While for the Qwen2-7B and Gemma2-9B models, the optimal ending layer is 7-th. This suggests that, for most decoder-only LLMs, modeling shallow-layer backward dependencies is crucial for enhancing sentence comprehension. The best layer for ending token prepending is similar across different backbones, typically falling within the 7-th to 8-th layer range.

5.7 Influence of Exit Layers

We examine the impact of exit layers in LLaMA2-7B using Pretended CoT and Knowledge Enhancement. As illustrated in Figure 4(d), our model consistently improves both Pretended CoT and Knowledge Enhancement across all layers and configurations. Furthermore, Pretended CoT and Knowledge Enhancement exhibit greater variability in performance across different layers compared to ours. This implies that the our method offers a more stable representational quality across layers.

Notably, employing the output of the model’s last layer is consistently suboptimal for STS tasks, consistent with prior research (Li and Li, 2024; Lei et al., 2024). Pretended CoT achieves optimal performance at the sixth-to-last layer, while Knowledge Enhancement peaks at the second-to-last layer. This variation suggests that the optimal layer can

Method	MR	CR	SUBJ	MPQA	SST2	TREC	MRPC	Avg.
PromptEOL	90.63	92.87	96.32	91.19	95.00	95.40	75.19	90.94
PromptEOL + TP (<i>Ours</i>)	90.90↑0.27	93.35↑0.48	96.58↑0.26	91.51↑0.32	95.50↑0.50	96.00↑0.60	76.12↑0.93	91.42↑0.48
Pretended CoT	90.10	92.24	96.32	91.54	95.11	94.20	75.77	90.75
Pretended CoT + TP (<i>Ours</i>)	90.45↑0.35	92.61↑0.37	96.52↑0.20	91.59↑0.05	95.77↑0.66	96.00↑1.80	76.81↑1.04	91.39↑0.64
Knowledge	89.84	93.03	96.21	91.54	94.78	97.20	73.91	90.93
Knowledge + TP (<i>Ours</i>)	90.39↑0.55	93.32↑0.29	96.31↑0.10	91.56↑0.02	94.51↓0.27	97.60↑0.40	76.06↑2.15	91.39↑0.46

Table 5: Results (accuracy scaled by 100x) on transfer learning tasks using LLaMA2-7B.

shift depending on the prompt.

5.8 Transfer Learning Tasks

We further evaluate the performance of our model on transfer learning tasks. We use standard transfer learning tasks provided by SentEval, including MR (Pang and Lee, 2005), CR (Hu and Liu, 2004), SUBJ (Pang and Lee, 2004), MPQA (Wiebe et al., 2005), SST-2 (Socher et al., 2013), TREC (Voorhees and Tice, 2000), and MRPC (Dolan and Brockett, 2005). For each task, we use the sentence embeddings generated by the model as features to train logistic regression classifiers.

The results for the transfer tasks, shown in Table 5, demonstrate that our method consistently outperforms all baselines, with improvements in 20 out of 21 cases across all datasets. This indicates that TP cultivates generalized sentence embeddings that perform outstandingly across various tasks. Pretended CoT and Knowledge Enhancement do not surpass the performance of PromptEOL, indicating that they are not consistently effective in enhancing performance across tasks.

Additionally, we find that ending token prepending at deeper layers typically between layer index 14 and 21, enhances performance on transfer tasks. This phenomenon differs significantly from the optimal layer for STS tasks, suggesting that transfer tasks benefit from additional layers to effectively model backward dependencies.

5.9 Evaluation of Capturing Dependencies in Contexts

We quantitatively analyse whether our proposed method enhances the ability of LLMs to capture dependencies in contexts on the STS-B test set using LLaMA2-7B. For both models, we follow Ethayarajh (2019) by selecting the last token as the pivot token. We then compute the Spearman correlation between the pivot token and the remaining tokens in each sentence to assess their dependency-capturing capabilities. The results are shown in a box plot in Figure 5.

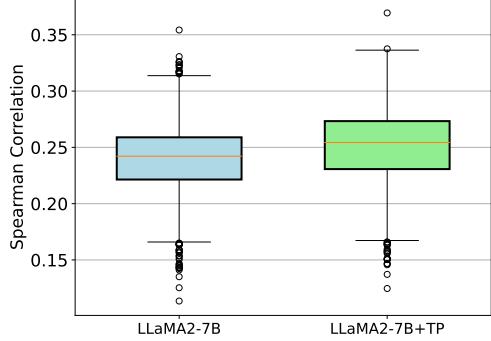


Figure 5: Box plot of the sentence-level Spearman correlation on the STS-B test set using Pretended CoT prompt.

The average sentence-level Spearman score for LLaMA2-7B and LLaMA2-7B+TP are 23.97 and 25.11, respectively. The results indicate that our method achieves an improved capability to capture backward dependencies compared to vanilla LLaMA2-7B model. This suggests that token prepending offers benefits for enhancing the ability of LLMs to capture dependencies in contexts.

6 Conclusion

In this paper, we introduce Token Prepending technique, a plug-and-play approach for deriving high-quality sentence embeddings from autoregressive LLMs without requiring any training and data. By intervening in the inputs to Transformer layers, TP enhances the ability of autoregressive LLMs to capture backward dependencies. Moreover, TP involves simply prepending a single token to the sentence, which adds negligible inference cost and can seamlessly integrate with prompt-based methods. Our extensive experiments demonstrate that TP technique can effectively and generally elicit sentence embeddings across a range of LLMs with varying architectures and parameter sizes, achieving outstanding performance on both STS datasets and transfer learning tasks. We find that starting TP from the first layer yields optimal results, and the best stopping point is typically around the 7th or 8th layer for LLMs with about 7B parameters.

Limitations

Although Token Prepending is a training-free technique, it requires tuning two hyperparameters (i.e., end layer for intermediate token prepending and exit layer) to achieve optimal sentence embeddings. Our results show that the best hyperparameters for TP vary based on the model, dataset, and prompt, which may increase adaptation costs when applying it to new scenarios.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the National Natural Science Foundation of China under Grants Nos. 62441225, 61972192, 62172208, 61906085. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. This work is supported by the Fundamental Research Funds for the Central Universities under Grant No.14380001.

References

- Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 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. The Association for Computer Linguistics.
- Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 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. The Association for Computer Linguistics.
- Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 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. The Association for Computer Linguistics.
- 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. The Association for Computer Linguistics.
- 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. Association for Computational Linguistics.
- 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*.
- Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*.
- Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. *Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation*. *CoRR*, abs/1708.00055.
- Sachin Chanchani and Ruihong Huang. 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. Association for Computational Linguistics.
- 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. *arXiv preprint arXiv:2505.12831*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: pre-training of deep bidirectional transformers for language understanding*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 4171–4186.
- Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65.

- 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.
- Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.
- Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023. Scaling sentence embeddings with large language models. *arXiv preprint arXiv:2307.16645*.
- Ting Jiang, Jian Jiao, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Deny 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.
- Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. 2024a. **Exploring concept depth: How large language models acquire knowledge at different layers?** *CoRR*, abs/2404.07066.
- Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, et al. 2024b. Exploring concept depth: How large language models acquire knowledge and concept at different layers? *arXiv preprint arXiv:2404.07066*.
- Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. **Nv-embed: Improved techniques for training llms as generalist embedding models**. *CoRR*, abs/2405.17428.
- Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. 2024. Meta-task prompting elicits embedding from large language models. *arXiv preprint arXiv:2402.18458*.
- Xianming Li and Jing Li. 2023. Angle-optimized text embeddings. *arXiv preprint arXiv:2309.12871*.
- 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.
- Tian Yu Liu, Matthew Trager, Alessandro Achille, Pramuditha Perera, Luca Zancato, and Stefano Soatto. 2024a. Meaning representations from trajectories in autoregressive models. In *The Twelfth International Conference on Learning Representations*.
- Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. 2024b. Fantastic semantics and where to find them: Investigating which layers of generative llms reflect lexical semantics. *arXiv preprint arXiv:2403.01509*.
- Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. 2024c. **Fantastic semantics and where to find them: Investigating which layers of generative llms reflect lexical semantics**. *CoRR*, abs/2403.01509.
- Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 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.
- Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. **Generative representational instruction tuning**. *CoRR*, abs/2402.09906.
- Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.
- Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022a. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874.
- Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022b. **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.
- Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278.
- Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *J. Mach. Learn. Res.*, 21:140:1–140:67.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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

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. Association for Computational Linguistics.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussonot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaee, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39:165–210.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple techniques for enhancing sentence embeddings in generative language models. *arXiv preprint arXiv:2404.03921*.

A Appendix

A.1 Comparison with Bidirectional Attention

We explore the performance of removing the causal attention mask. To this end, we design two types of bidirectional attention masks: 1) enabling bidirectional attention for the last token, and 2) enabling bidirectional attention for the input sentence. To ensure fairness, the starting position of the non-causal attention aligns with the position of the prepended <PST> token.

Using Pretended CoT as the prompt, the results are presented in Table 6. Both types of bidirectional attention masks result lead to a substantial decrease in performance. This observation is consistent with prior research (BehnamGhader et al., 2024; Li and Li, 2024), which indicates that, due to the inductive bias of autoregressive large language models, employing a bidirectional attention mechanism tends to reduce model performance.

A.2 Multi-Task Evaluation

We evaluate the TP technique across 12 classification datasets, 3 pair classification datasets, 4 reranking datasets, 11 clustering datasets, 1 summarization dataset, and 1 additional STS dataset.

Initialization Method	STS Avg.
Vanilla LLM	76.86
TP (Ours)	77.54
Bidirectional Attention (Last token)	53.06
Bidirectional Attention (Input Sentence)	43.70

Table 6: Influence of the modified attention masks.

Method	PromptEOL	PromptEOL+TP
AmazonCounterfactual	70.83	71.71
AmazonPolarity	88.48	94.57
AmazonReviews	46.03	47.77
Banking77	78.94	82.24
Emotion	48.35	51.05
Imdb	79.10	81.44
MassiveIntent	72.49	75.22
MassiveScenario	75.41	78.69
MTOPDomain	90.49	93.63
MTOPIntent	81.48	83.16
ToxicConversations	64.51	68.68
TweetSentimentExtraction	60.55	61.15
Average (12)	71.39	74.11

Table 7: Results (accuracy scaled by 100x) on classification datasets using LLaMA2-7B.

The datasets we used are all from the MTEB benchmark (Muennighoff et al., 2022).

Method	PromptEOL	PromptEOL+TP
SprintDuplicateQuestions	43.02	51.61
TwitterSemEval2015	65.61	67.70
TwitterURLCorpus	78.97	80.90
Average (3)	62.53	66.74

Table 8: Results (accuracy scaled by 100x) on pair classification datasets using LLaMA2-7B.

Method	PromptEOL	PromptEOL+TP
AskUbuntuDupQuestions	53.88	57.02
MindSmallRerank	29.97	29.89
SciDocsRR	71.38	77.49
StackOverflowDupQuestions	40.63	43.19
Average (4)	48.97	51.90

Table 9: Results (average precision scaled by 100x) on reranking datasets using LLaMA2-7B.

The 12 classification datasets include AmazonCounterfactual, AmazonPolarity, AmazonReviews, Banking77, Emotion, Imdb, MassiveIntent, MassiveScenario, MTOPDomain, MTOPIntent, ToxicConversations, and TweetSentimentExtraction. The 3 pair classification datasets are SprintDuplicateQuestions, TwitterSemEval2015, and TwitterURLCorpus. The 4 reranking datasets are AskUbuntuDupQuestions, MindSmallRerank, SciDocsRR, and StackOverflowDupQuestions. The 11 clustering datasets are ArxivClusteringP2P, ArxivClusteringS2S, BiorxivClusteringP2P, BiorxivClusteringS2S, MedrxivClusteringP2P, MedrxivClusteringS2S, RedditClustering, RedditClusteringP2P, StackExchangeClustering, StackExchangeClusteringP2P, TwentyNewsgroupsClustering. The summarization dataset is SummEval. The additional STS dataset is BIOSSES.

The results on classification datasets, pair classification datasets, reranking datasets, clustering datasets, retrieval datasets, summarization dataset and additional STS dataset are presented in Table 7, Table 8, Table 9, Table 10, Table 11, and Table 12, respectively. Our method shows improvement in 40 out of 44 cases across all datasets. Specifically, the TP technique achieves an average improvement of 2.72 on classification datasets, a 4.21 increase on pair classification datasets, a 2.93 gain on reranking datasets, an improvement of 4.51 on clustering datasets, a 1.23 gain on summarization dataset, and a 7.07 gain on additional STS dataset.

A.3 More Prompt Baseline Evaluation

We identify two prompts A and B similar to PromptEOL that could benefit more significantly

Method	PromptEOL	PromptEOL+TP
ArxivClusteringP2P	34.87	43.57
ArxivClusteringS2S	31.19	39.82
BiorxivClusteringP2P	19.56	25.75
BiorxivClusteringS2S	24.34	31.92
MedrxivClusteringP2P	27.65	26.41
MedrxivClusteringS2S	34.53	35.98
RedditClustering	24.69	31.69
RedditClusteringP2P	48.52	48.54
StackExchangeClustering	42.16	44.90
StackExchangeClusteringP2P	33.56	33.03
TwentyNewsgroupsClustering	27.61	36.98
Average (11)	31.70	36.24

Table 10: Results (V-measure scaled by 100x) on clustering datasets using LLaMA2-7B.

Method	PromptEOL	PromptEOL+TP
SummEval	28.88	30.11

Table 11: Results (Spearman correlation scaled by 100x) on summarization dataset using LLaMA2-7B.

from TP. These prompts are derived from (Li and Li, 2024) and (Li and Li, 2023). In addition, we design two prompts C and D to impart clear semantic information to the <PST> token. The specific prompts are shown below:

Prompt A: "The representative word for sentence <PST> '[TEXT]' is:"
Prompt B: "Summarize sentence <PST> '[TEXT]' in one word."
Prompt C: "Given the keyword <PST>, this sentence: '[TEXT]' means in one word."
Prompt D: "This sentence: <PST> and '[TEXT]' means in one word."

We conduct comparative experiments with and without TP using Prompt A and B. The results are shown in the Table 13. As shown in the table, our proposed method significantly improves the performance of prompt A and B, achieving a 10.64 and 9.26 increase, respectively. This validates our hypothesis that simple prompts without prior knowledge, similar to PromptEOL, rely more heavily on modeling backward dependencies to effectively capture semantics.

We observe that compared to the results in Table 1, Prompt C and D do not further enhance TP's performance in the Table 13. We speculate this is because TP edits occur in the intermediate layers of the LLM, and providing prior knowledge about the <PST> token in the input does not effectively

Method	PromptEOL	PromptEOL+TP
BIOSSES	62.66	69.73

Table 12: Results (Spearman correlation scaled by 100x) on additional STS dataset using LLaMA2-7B.

help the model grasp its intended meaning.

A.4 Number of <PST> tokens

We further analyze the impact of the number of inserted <PST> tokens on performance based on PromptEOL. The results are shown in the Table 14. Incorporating two <PST> tokens achieve a slight improvement in TP performance (by 0.08 points). However, prepending more <PST> tokens leads to a decline in performance, as evidenced by the results of 3 <PST> tokens and 4 <PST> tokens.

A.5 Masking <PST> Token in the First Layer

We mask the <PST> token in the first layer of the LLaMA2 7B model to mitigate the impact of token initialization. The experimental results are shown in the Table 15.

The performance is slightly lower than that of PromptEOL+TP. This may be attributed to the role of the <PST> token in the first layer, where it acts as a placeholder, enabling the LLM to interpret the input length as N+1. Despite its random initialization, the <PST> token ensures consistent input length across all layers.

A.6 Resuming TP at Different Layers

We explore the practice of using TP for a few layers, and then pause for a few layers and then resume again. We conduct experiments based on the best practices of PromptEOL and TP, applying TP at the 1st layer, stopping TP at the 8th layer, and early exiting at the 27th layer. The results are shown in the Table 16. Once resumed, TP remains active until output. We find that resuming TP at deeper layers (e.g., layer 21) yields a slight performance improvement.

A.7 TP without Prompt

We test performance of TP without any prompt. The results are shown in the Table 17. Although the absence of a prompt significantly degrades performance, TP still manages to provide improvements.

Method	Backbone	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Prompt A	LLaMA2-7B	44.79	65.73	50.39	58.70	58.10	51.42	47.92	53.86
Prompt A + TP (<i>Ours</i>)	LLaMA2-7B	52.22	67.44	58.00	69.89	71.32	65.76	66.86	64.50
Prompt B	LLaMA2-7B	51.18	73.74	63.13	68.87	70.96	63.29	67.45	65.52
Prompt B + TP (<i>Ours</i>)	LLaMA2-7B	64.32	80.18	70.49	77.29	78.36	79.32	73.47	74.78
Prompt C	LLaMA2-7B	64.34	77.87	67.62	74.25	72.15	77.33	74.91	72.64
Prompt D	LLaMA2-7B	61.99	80.83	71.69	78.31	77.06	77.82	73.68	74.48

Table 13: Results on STS tasks (Spearman correlation scaled by 100x) using prompt A, B, C, and, D in Appendix A.3.

Method	Backbone	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
1 <PST> (PromptEOL+TP)	LLaMA2-7B	66.90	83.12	74.31	79.87	80.03	80.67	75.40	77.19
2 <PST>	LLaMA2-7B	65.98	83.34	74.31	80.31	80.17	80.78	76.01	77.27
3 <PST>	LLaMA2-7B	64.98	82.92	72.99	79.33	79.34	80.01	75.95	76.50
4 <PST>	LLaMA2-7B	64.52	82.90	72.01	78.90	78.67	79.50	75.75	76.04

Table 14: Results on STS tasks (Spearman correlation scaled by 100x) based on PromptEOL, with various number of <PST> placeholder tokens incorporated into the prompt.

Method	Backbone	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
PromptEOL+TP	LLaMA2-7B	66.90	83.12	74.31	79.87	80.03	80.67	75.40	77.19
Masking <PST>	LLaMA2-7B	66.85	82.97	74.17	79.72	79.94	80.55	75.26	77.07

Table 15: Results on STS tasks (Spearman correlation scaled by 100x) based on PromptEOL, masking the <PST> token in the first layer.

Method	Backbone	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
PromptEOL+TP	LLaMA2-7B	66.90	83.12	74.31	79.87	80.03	80.67	75.40	77.19
Resuming at Layer 9	LLaMA2-7B	66.97	82.75	73.24	78.75	79.36	80.40	75.33	76.69
Resuming at Layer 10	LLaMA2-7B	66.76	82.91	73.26	79.11	79.40	80.30	75.58	76.76
Resuming at Layer 11	LLaMA2-7B	66.68	83.01	73.34	79.13	79.46	80.17	75.60	76.77
Resuming at Layer 16	LLaMA2-7B	66.84	83.07	74.22	79.83	79.96	80.56	75.36	77.12
Resuming at Layer 26	LLaMA2-7B	66.95	83.18	74.33	79.89	80.04	80.66	75.40	77.21

Table 16: Results on STS tasks (Spearman correlation scaled by 100x) based on PromptEOL, resuming TP at different layer.

Method	Backbone	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
w/o prompt	LLaMA2-7B	9.13	22.25	11.04	33.09	34.92	15.94	33.73	22.87
w/o prompt + TP	LLaMA2-7B	11.68	25.85	13.08	33.70	46.58	22.05	42.61	27.94

Table 17: Results on STS tasks (Spearman correlation scaled by 100x) based on PromptEOL without any prompt.