PromptEmbedder: Efficient and Transferable Text Embedding via Dual-LLM Soft Prompting

Yu-Che Tsai[1] Kuan-Yu Chen[1] Yuan-Hao Chen[1]
Yu-Han Chang[1] Ching-Yu Tsai[1] Yu-Hsiang Chuang[1] Shou-De Lin[1,2]
[1]Department of Computer Science and Information Engineering, National Taiwan University
[2]National Taiwan University AI Center of Research Excellence
Taipei, Taiwan
{f09922081,d13922034,b11902172,d14922008,r14922006,b12902129,sdlin}@csie.ntu.edu.tw

Abstract

Large Language Models (LLMs) have demonstrated remarkable efficacy in text embedding, yet current adaptation methods like LoRA face significant bottlenecks in computational efficiency and cross-architecture transferability. Whenever a new backbone emerges, existing approaches require costly retraining from scratch. To address this, we propose PromptEmbedder, a novel dual-LLM framework that decouples embedding knowledge from specific backbone weights. PromptEmbedder utilizes a Prompting LLM to generate instruction-aware soft prompts for a frozen Embedding LLM via a differentiable generation process with continuous relaxation, ensuring full gradient flow during contrastive training. By localizing task-specific knowledge within the Prompting LLM, adapting to new architectures requires only retraining a lightweight linear alignment matrix. Evaluations on the MTEB benchmark show that PromptEmbedder achieves comparable performance with LoRA finetuning while reducing GPU memory by 40% and accelerating training by 3.7×. Our approach establishes a scalable, architecture-agnostic paradigm for efficient LLM-based representation learning.

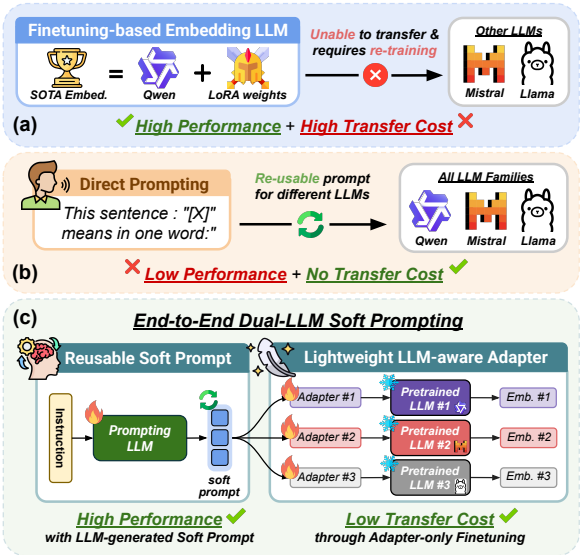

Figure 1: Comparison of existing embedding approaches and PromptEmbedder. (a) Fine-tuning-based methods achieve high performance but require costly retraining for each new backbone. (b) Direct prompting methods are reusable but suffer from substantially lower embedding quality. (c) PromptEmbedder achieves both high performance and transferability via a dual-LLM architecture.

1 Introduction

The advent of Large Language Models (LLMs) has fundamentally transformed natural language processing, extending their utility beyond traditional text generation to sophisticated representation learning tasks. Recent investigations have revealed that LLMs, when appropriately adapted, can produce high-quality text embeddings that rival or surpass dedicated embedding models across diverse downstream applications, including text classification, information retrieval, and semantic similarity (Karpukhin et al., 2020; Liu et al., 2024; Xiong et al., 2021).

The dominant paradigm for adapting LLMs to embedding tasks relies on parameter-efficient fine-tuning (PEFT) techniques. Widely used embedding models such as E5-Mistral (Wang et al., 2024) and GRITLM (Muennighoff et al., 2024) apply Low-Rank Adaptation (LoRA) (Hu et al., 2022) with contrastive objectives to generate high-quality text embeddings. While effective, this approach fundamentally couples all learned embedding knowledge to the weights of the target backbone. As a consequence, whenever a new LLM architecture emerges, the entire fine-tuning pipeline must be restarted from scratch. According to recent release histories, major LLM backbones (e.g., Qwen series (Qwen et al., 2025; Yang et al., 2025, 2024)) have been updated every six to nine months. More

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 9450–9464
October 24-29, 2026 ©2026 Association for Computational Linguistics

importantly, training a state-of-the-art embedding model such as Qwen3-Embedding requires upwards of 150 million training samples (Zhang et al., 2025), which could cost millions of dollars in compute (Cottier et al., 2024). As LLM backbones continue to evolve rapidly, it is essential to develop a more generalized paradigm that allows for the efficient transfer of embedding knowledge across model generations.

To alleviate the intensive training process, another line of research focuses on developing training-free methods via specialized prompting techniques. For instance, PromptEOL (Jiang et al., 2024), MetaEOL (Lei et al., 2024), and Echo (Springer et al., 2025) use hand-crafted instruction templates to extract semantic representations from the hidden states of pretrained LLMs. Since these prompting methods do not alter any parameters of the pretrained LLM, they are fully transferable across architectures by design. However, without appropriate adaptation, they significantly underperform fine-tuned models on standard benchmarks, making them impractical for real-world applications.

As illustrated in Figure 1, existing approaches face a fundamental trade-off between performance and transfer cost. To address these limitations, we propose PromptEmbedder, a framework that achieves strong performance while remaining efficiently transferable across LLM architectures. PromptEmbedder is built on a dual-LLM architecture consisting of a *Prompting LLM* and a *Pretrained LLM*. The Prompting LLM autoregressively generates k soft instruction tokens, which are inserted into the input text to guide the pretrained LLM in producing text representations from the last-token hidden state. Built on this architecture, PromptEmbedder introduces two core technical strengths: (1) **Differentiable Soft Prompt Generation.** Soft prompts are generated through a continuous relaxation over the vocabulary. This allows the Prompting LLM to be jointly optimized with the frozen pretrained LLM in an end-to-end fashion without discrete sampling. (2) **Efficient Cross-Model Transfer.** All embedding-specific knowledge is localized entirely within the Prompting LLM without modifying the pretrained LLM. As a result, transferring PromptEmbedder to an unseen pretrained LLM requires retraining only a lightweight linear alignment matrix rather than repeating the full fine-tuning pipeline.

We conduct extensive evaluations on the MTEB benchmark and demonstrate that PromptEmbedder achieves three critical advantages. In terms of effectiveness, PromptEmbedder consistently outperforms existing prompting-based embedding methods such as PromptEOL and Prompt-Tuning. Notably, our framework achieves 96.0%–99.8% of the performance of LoRA finetuning by inserting only five soft tokens into the input text. Furthermore, PromptEmbedder exhibits superior transferability by adapting to unseen LLM backbones with 3.8× faster convergence. Specifically, PromptEmbedder achieves the same performance level as LoRA training for 66 hours in only 18 hours. Finally, in terms of efficiency, maintaining a completely frozen pretrained LLM alleviates gradient computation overhead. This design reduces GPU memory usage by 36% to 40% and delivers a 3.7× training speedup compared to LoRA. Consequently, PromptEmbedder establishes a new paradigm for scalable and efficient LLM-based text embeddings.

2 Related Work

Fine-tuning LLMs as Embedding Models. Fine-tuning decoder-only LLMs has become the dominant paradigm for generating high-quality text representations. E5-Mistral (Wang et al., 2024) was among the first to demonstrate that PEFT-based fine-tuning of LLMs with contrastive objectives substantially outperforms encoder-only models. GRITLM (Muennighoff et al., 2024) further unifies generative and embedding objectives within a single model. To address the inherent limitation of unidirectional attention in decoder-only LLMs, LLM2Vec (BehnamGhader et al., 2024) introduces bidirectional attention combined with masked next-token prediction. NV-Embed (Lee et al., 2024) proposes a latent attention pooling layer and a two-stage training strategy to handle false negatives in non-retrieval tasks. BGE-en-ICL (Li et al., 2025) instead leverages in-context learning within the original LLM framework to guide embedding generation. Most recently, Qwen3-Embedding (Zhang et al., 2025) pushes performance to new heights by training on over 150 million samples.

Prompting LLMs as Embedding Models. To avoid the cost of fine-tuning, a parallel line of research explores training-free methods that elicit embeddings through prompting. PromptEOL (Jiang et al., 2024) and MetaEOL (Lei et al., 2024) design hand-crafted instruction templates that guide frozen LLMs to summarize input text, extracting semantic

representations directly from hidden states without any parameter updates. While these methods eliminate training overhead, they result in substantially lower performance compared to fine-tuned methods. Soft prompt tuning offers a middle ground by prepending learnable continuous vectors to the input and optimizing only these parameters while keeping the backbone frozen. Prefix Tuning (Li and Liang, 2021) and Prompt Tuning (Lester et al., 2021) demonstrate the viability of this approach on general NLP tasks, and P-Tuning v2 (Liu et al., 2022) extends it with deep, layer-wise prefixes for greater expressiveness. In the context of embedding models, soft prompts can similarly steer a frozen backbone toward the embedding objective, avoiding model modification while still benefiting from task-specific adaptation. However, the fixed set of prompt vectors limits their ability to perform diverse tasks, hindering the performance of soft prompt tuning behind conventional finetuning approaches.

3 Preliminary

Notations. We consider a LLM–based embedding system that maps a text document to a dense vector representation. Specifically, we adopt a pretrained decoder-only language model with parameters $\psi = \{\mathbf{E}, \theta, \phi\}$, where $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ denotes the token embedding matrix, θ represents the transformer decoder parameters, and ϕ corresponds to the language modeling head used for next-token prediction. Here, d denotes the hidden dimension and $|\mathcal{V}|$ is the vocabulary size.

Instruction-based Text Embedding. Following recent state-of-the-art embedding models (Muennighoff et al., 2024; Wang et al., 2024; Zhang et al., 2025), we adopt an instruction-based embedding paradigm. Given a task instruction $t = [t_1, \ldots, t_m]$ and an input text $X = [x_1, \ldots, x_n]$, we first concatenate them into a single sequence:

$$\texttt{Instruction: } t \texttt{ Query: } X \texttt{ <EOS>}.$$

Here, t is a short natural-language description of the task (e.g., "*Given a question, retrieve passages that answer the question*" in retrieval settings). The concatenated sequence is then mapped to token embeddings through $\mathbf{E}$ and then processed by the transformer decoder Dec_θ to produce contextual hidden states. Finally, the resulting text representation is obtained through last-token pooling, which refers to the hidden state corresponding to the <EOS> token:

$$\mathbf{v} = \mathrm{Dec}_\theta(\mathbf{E}([t; X; \texttt{<EOS>}]))_{\texttt{<EOS>}} \in \mathbb{R}^d. \quad (1)$$

Finally, the obtained embeddings are optimized by minimizing a contrastive loss via finetuning the pretrained LLM.

4 Method

Overview. As illustrated in Fig. 2, PromptEmbedder consists of a *Prompting LLM* $\mathcal{M}_{\text{prompt}}$ with parameters $\psi_p = \{\mathbf{E}_p, \theta_p, \phi_p\}$ and a frozen *Pretrained LLM* $\mathcal{M}_{\text{pretrained}}$ with parameters $\psi_e = \{\mathbf{E}_e, \theta_e, \phi_e\}$. Given a task description t, $\mathcal{M}_{\text{prompt}}$ generates k soft prompt vectors $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_k]$ in a fully differentiable manner. These vectors are projected into the embedding space of $\mathcal{M}_{\text{pretrained}}$ and inserted as a prefix to the input text. The final embedding is then extracted from the augmented sequence via last-token pooling. During training, only the LoRA parameters of $\mathcal{M}_{\text{prompt}}$ and the projection matrix are optimized using a contrastive objective while $\mathcal{M}_{\text{pretrained}}$ remains completely frozen.

4.1 Soft Prompt Generation through LLM

A key innovation of PromptEmbedder is to dynamically construct soft prompts from task instructions using an auxiliary LLM $\mathcal{M}_{\text{prompt}}$. Since $\mathcal{M}_{\text{prompt}}$ must be trained through the frozen $\mathcal{M}_{\text{pretrained}}$, the prompt generation process should remain fully differentiable to allow gradients from the contrastive loss to propagate back to $\mathcal{M}_{\text{prompt}}$. However, standard autoregressive generation involves discrete token sampling, which breaks the gradient flow. Inspired by prior work on soft token generation (Deng et al., 2025; Tsai et al., 2026), we instead generate soft prompts through an iterative process over a continuous relaxation of the vocabulary, which preserves differentiability and allows $\mathcal{M}_{\text{prompt}}$ to be trained end-to-end via the embedding objective.

Instruction-conditioned Soft Prompt Generation. Given a task instruction $t = [t_1, \ldots, t_m]$, we first feed it into the prompting model $\mathcal{M}_{\text{prompt}}$, consisting of the embedding matrix $\mathbf{E}_p$ and transformer decoder θ_p. Let $\mathbf{H}^{(0)} = \mathrm{Dec}_{\theta_p}(\mathbf{E}_p(t))$ denote the final-layer hidden states of the instruction, and let $\mathbf{g}^{(0)} = \mathbf{H}_m^{(0)} \in \mathbb{R}^{d_p}$ be the hidden state of the last instruction token.

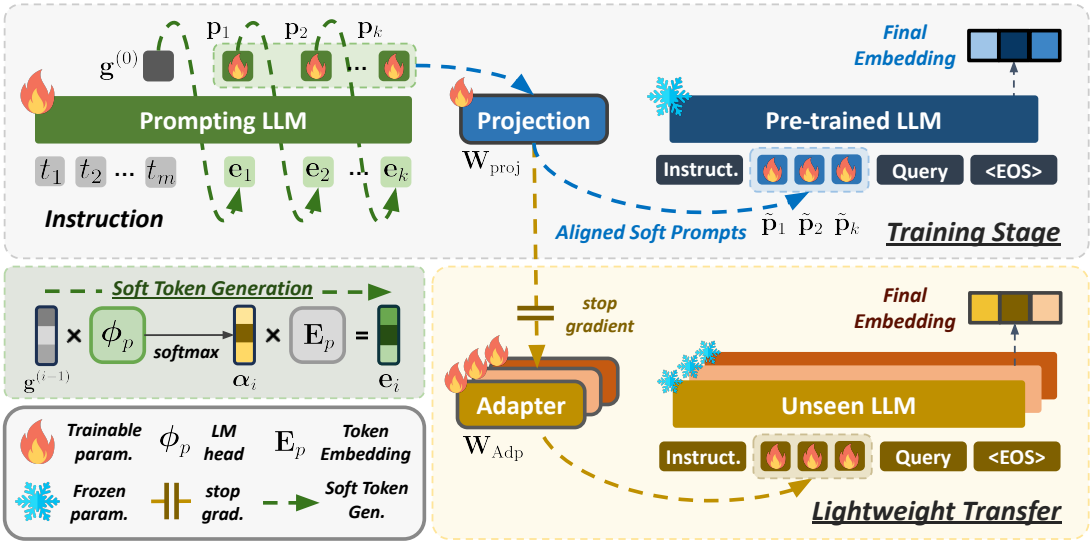

Figure 2: Overview of the PromptEmbedder framework. **Top:** During training, $\mathcal{M}_{\text{prompt}}$ generates k soft prompt vectors from a task instruction, which are projected and prepended to the input of the frozen pretrained LLM $\mathcal{M}_{\text{pretrained}}$. **Bottom-left:** Each soft token $\mathbf{e}_t$ is generated differentiably as a weighted sum over the token embedding matrix $\mathbf{E}_p$. **Bottom-right:** For lightweight transfer to an unseen backbone, only a small adapter is trained while $\mathcal{M}_{\text{prompt}}$ and $\mathcal{M}_{\text{pretrained}}$ are completely frozen.

Soft prompt generation then starts autoregressively from $\mathbf{g}^{(0)}$. At generation step i, we first construct a soft token $\mathbf{e}_i$ from the current last hidden state $\mathbf{g}^{(i-1)}$. Specifically, we compute a vocabulary distribution using the LM head ϕ_p:

$$\boldsymbol{\alpha}_i = \text{softmax}\Big(\phi_p(\mathbf{g}^{(i-1)})\Big) \in \mathbb{R}^{|\mathcal{V}|}, \quad (2)$$

and obtain the corresponding soft token by computing a convex combination over the vocabulary:

$$\mathbf{e}_i = \boldsymbol{\alpha}_i^\top \mathbf{E}_p \in \mathbb{R}^{d_p}. \quad (3)$$

We then append $\mathbf{e}_i$ to the current sequence and pass the extended sequence through $\mathcal{M}_{\text{prompt}}$ again. Formally, let $\mathbf{Z}^{(i)} = [t; \mathbf{e}_1; \ldots; \mathbf{e}_i]$ denote the instruction augmented with the first i generated soft tokens. We compute $\mathbf{H}^{(i)} = \text{Dec}_{\theta_p}\big(\mathbf{E}_p(\mathbf{Z}^{(i)})\big)$ and take the hidden state of the last position as $\mathbf{p}_i = \mathbf{H}^{(i)}_{m+i} \in \mathbb{R}^{d_p}$. This hidden state $\mathbf{p}_i$ serves as the prompt representation retained at step i, while the next generation step uses $\mathbf{g}^{(i)} = \mathbf{p}_i$ as its current state. Repeating this process for K steps yields the final soft prompt sequence: $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_K] \in \mathbb{R}^{K \times d_p}$.

Cross-LLM Prompt Alignment. Since the two pretrained models, $\mathcal{M}_{\text{prompt}}$ and $\mathcal{M}_{\text{pretrained}}$, may differ not only in hidden dimension (with hidden dimensions d_p and d_e) but also in the structure of their embedding spaces, the generated soft prompts cannot be directly used as inputs to $\mathcal{M}_{\text{pretrained}}$. We therefore introduce a learnable linear projection $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{d_e \times d_p}$ to map each prompt vector into the embedding space of $\mathcal{M}_{\text{pretrained}}$:

$$\tilde{\mathbf{p}}_i = \mathbf{W}_{\text{proj}} \mathbf{p}_i \in \mathbb{R}^{d_e}. \quad (4)$$

This produces the aligned prompt sequence $\tilde{\mathbf{P}} = [\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_k] \in \mathbb{R}^{k \times d_e}$, which is inserted as a prefix to the input sequence processed by $\mathcal{M}_{\text{pretrained}}$.

4.2 Embedding Extraction

Given the aligned soft prompt sequence $\tilde{\mathbf{P}}$, we insert it between the instruction t and the input text X to construct the input sequence for the frozen embedding model $\mathcal{M}_{\text{pretrained}}$:

$$\mathbf{X}_{\text{combined}} = \mathbf{E}_e(t) \| \tilde{\mathbf{P}} \| \mathbf{E}_e(X) \| \mathbf{E}_e(\texttt{<EOS>}), \quad (5)$$

where $\mathbf{E}_e$ denotes the token embedding matrix of $\mathcal{M}_{\text{pretrained}}$ and $\|$ denotes concatenation. The resulting sequence is processed by the frozen transformer decoder θ_e of $\mathcal{M}_{\text{pretrained}}$, and the final text embedding is obtained by last-token pooling as described in Eq. (1):

$$\mathbf{v} = \text{Dec}_{\theta_e}(\mathbf{X}_{\text{combined}})_{\texttt{<EOS>}} \in \mathbb{R}^{d_e}. \quad (6)$$

4.3 Training Objective

Following prior embedding models (Muennighoff et al., 2024; Tsai et al., 2026; Wang et al., 2024), we train PromptEmbedder with a contrastive InfoNCE objective (Oord et al., 2018). The training data consists of triplets (q_i, d_i^+, d_i^-), where q_i is

a query, d_i^+ is a relevant document, and d_i^- is a hard negative. For each query q_i, the negative set is $\mathcal{N}_i = \{d_i^-\} \cup \{d_j^+\}_{j \neq i}$, which combines the paired hard negative with in-batch positives from other examples. The loss is defined as:

$$\mathcal{L} = -\frac{1}{N_b} \sum_{i=1}^{N_b} \log \frac{e^{\mathrm{sim}(\mathbf{v}_{q_i}, \mathbf{v}_{d_i^+})/\tau}}{\sum_{d \in \{d_i^+\} \cup \mathcal{N}_i} e^{\mathrm{sim}(\mathbf{v}_{q_i}, \mathbf{v}_d)/\tau}}, \tag{7}$$

where $\mathrm{sim}(\cdot, \cdot)$ is cosine similarity and $\tau = 0.2$ is the temperature hyperparameter. Gradients propagate through the frozen $\mathcal{M}_{\text{pretrained}}$ to update $\mathbf{W}_{\text{proj}}$ and the LoRA parameters of $\mathcal{M}_{\text{prompt}}$, while $\mathcal{M}_{\text{pretrained}}$ remains fixed throughout training.

4.4 Transferability to Unseen LLM Backbones

Another important advantage of PromptEmbedder is its ability to generalize across different LLM backbones without retraining the prompting model $\mathcal{M}_{\text{prompt}}$ and the projection matrix $\mathbf{W}_{\text{proj}}$. Since all task knowledge is already learned and encoded in these two modules, adapting to a new embedding model $\mathcal{M}_{\text{pretrained}}'$ only requires a lightweight adapter. Specifically, we introduce an additional trainable projection $\mathbf{W}_{\text{adp}} \in \mathbb{R}^{d_{e'} \times d_e}$, which is appended after the original projection layer:

$$\tilde{\mathbf{p}}_i' = \mathbf{W}_{\text{adp}} \mathbf{W}_{\text{proj}} \mathbf{p}_i \in \mathbb{R}^{d_{e'}}. \tag{8}$$

The adapted prompt sequence $\tilde{\mathbf{P}}' = [\tilde{\mathbf{p}}_1', \ldots, \tilde{\mathbf{p}}_k']$ is then inserted into the new embedding model $\mathcal{M}_{\text{pretrained}}'$ following the same procedure as in Section 4.2. During the transfer learning stage, only $\mathbf{W}_{\text{adp}}$ is trained using the contrastive objective from Eq. (7), while the new embedding model $\mathcal{M}_{\text{pretrained}}'$ and both $\mathcal{M}_{\text{prompt}}$ and the original projection $\mathbf{W}_{\text{proj}}$ remain fixed.

5 Experiment

5.1 Experimental Setup

Backbone LLMs. We evaluate PromptEmbedder on a range of pretrained LLMs spanning 1B to 8B parameters: Llama-3.2-1B, Llama-3.2-3B, Llama-3.1-8B, Mistral-7B, and Qwen-2.5-7B.

Evaluation. We evaluate on MTEB (English, v2) (Enevoldsen et al., 2025), a comprehensive benchmark covering 41 datasets across seven task categories. For a more extensive comparison, we have also evaluated on instruction following benchmark (Feng et al., 2025). Detailed results and the specific instructions used for each dataset are provided in Appendix C and Appendix D, respectively.

Baselines. We compare PromptEmbedder against three categories of methods. **(1) Training-free prompting:** comprising Vanilla (direct inference without any instructional template) alongside PromptEOL (Jiang et al., 2024), Pretend-CoT (Zhang et al., 2024), and Echo (Springer et al., 2025). These methods elicit embeddings from LLMs via carefully designed prompts without any parameter updates. **(2) Prompt tuning** (Lester et al., 2021): The soft-prompt baseline that learns a fixed set of continuous prefix vectors prepended to the input, with all LLM parameters frozen. [1] **(3) LoRA fine-tuning:** Parameter-efficient adaptation via low-rank weight updates (Hu et al., 2022) applied directly to the pretrained LLM. **(4) Off-the-shelf embedding models:** including E5-large (Wang et al., 2022), GTE-large (Li et al., 2023), LLM2Vec (BehnamGhader et al., 2024) and GRITLM (Muennighoff et al., 2024).

Training Details. We utilize the dataset from (Li et al., 2025), which contains supervised pairs and hard negatives for contrastive learning. To balance training efficiency and computational costs, we sample 10% (0.2M) of the original data for our experiments following prior work (Tsai et al., 2026). We use Qwen3-0.6B as our default prompting model and set the number of soft prompting tokens to $k = 5$. The implementation details and hyperparameters for PromptEmbedder and baseline methods are provided in Appendix B.

5.2 Main Results

We first compare the embedding performance of PromptEmbedder with other baseline methods on the MTEB (English, v2) benchmark. The results are presented in Table 1. We highlight two key observations.

PromptEmbedder consistently achieves the best performance among frozen-backbone methods. Although training-free prompting methods such as PromptEOL and Echo require no training data and could be applied to arbitrary pretrained LLMs, they suffer a substantial performance gap compared to soft-prompting methods (e.g., PromptEOL at 44.88 vs. PromptEmbedder at 61.18 on Llama-3.2-

[1] We also attempted to train layer-wise soft prompts like P-Tuning v2 (Liu et al., 2022) as an additional baseline; however, we found it consistently failed to converge and yield degenerate embeddings. We therefore exclude it from our main comparison.

Method	Backbone: Llama-3.2-1B								Backbone: Llama-3.2-3B							
	Retr.	Rerank.	Clust.	PairClass.	Class.	STS	Summ.	Avg.	Retr.	Rerank.	Clust.	PairClass.	Class.	STS	Summ.	Avg.
Training-free Discrete Prompting																
Vanilla	8.96	25.07	41.00	47.36	50.67	50.38	-2.11	35.38	9.24	25.22	41.67	50.76	51.57	50.60	-3.27	36.02
PromptEOL	17.56	28.24	43.00	51.62	68.57	62.55	8.37	44.88	19.27	31.16	47.94	67.24	73.24	67.30	22.64	49.82
Pretend-CoT	14.24	27.70	42.04	50.68	**71.23**	62.35	11.97	44.30	19.33	31.81	45.75	57.62	72.32	70.62	19.05	49.18
Echo	14.21	29.36	44.26	**63.20**	62.05	61.93	8.53	43.70	16.31	29.58	44.83	67.48	63.92	65.06	8.51	45.74
Soft-Prompting Methods																
Prompt-Tuning	66.97	39.66	76.03	41.37	28.49	68.17	24.89	50.90	42.18	45.61	51.29	76.21	72.75	**78.03**	29.90	60.15
PromptEmbedder	**76.37**	**50.72**	**77.40**	47.42	41.59	**76.45**	**32.16**	**61.18**	**46.87**	**48.58**	**53.42**	**81.30**	**80.39**	77.70	**35.98**	**63.79**
Fine-tuning																
LoRA	81.96	52.76	82.57	47.71	48.13	75.48	31.49	63.73	48.57	47.23	51.69	84.18	80.47	77.19	34.63	63.89

Method	Backbone: Qwen2.5-7B								Backbone: Mistral-7B							
	Retr.	Rerank.	Clust.	PairClass.	Class.	STS	Summ.	Avg.	Retr.	Rerank.	Clust.	PairClass.	Class.	STS	Summ.	Avg.
Training-free Discrete Prompting																
Vanilla	7.40	28.23	40.98	49.75	43.59	45.02	-2.12	32.65	8.30	25.85	40.16	50.24	51.37	48.67	-2.93	35.01
PromptEOL	24.48	32.78	46.13	71.36	69.65	67.00	6.22	50.04	21.64	30.58	43.52	61.95	72.68	67.01	15.75	48.86
Pretend-CoT	26.06	31.15	47.23	68.71	74.76	68.98	23.02	52.29	21.12	30.19	43.14	67.09	75.49	69.07	23.33	50.19
Echo	10.29	30.58	39.38	66.74	55.24	57.22	6.40	39.57	13.82	29.83	42.89	66.56	62.79	61.74	4.97	43.60
Soft-Prompting Methods																
Prompt-Tuning	39.37	46.95	50.71	76.47	70.74	75.19	28.04	58.38	47.38	48.53	53.96	80.59	78.10	78.22	**36.01**	63.64
PromptEmbedder	**47.88**	**49.73**	**55.03**	**83.15**	**79.88**	**75.70**	**31.32**	**63.89**	**50.42**	**49.77**	**55.16**	**84.11**	**79.75**	**78.26**	34.77	**65.23**
Fine-tuning																
LoRA	51.10	47.49	55.26	84.46	80.10	74.71	33.21	64.18	55.24	49.21	54.28	85.65	84.36	73.98	36.31	66.32

Table 1: Main results on the MTEB (English, v2) benchmark across four representative LLM backbones. Methods are grouped by their adaptation strategy: training-free discrete prompting, soft-prompting, and parameter-efficient fine-tuning. **Bold** highlights the best performance among frozen-backbone methods.

1B). This demonstrates the necessity of additional training data and the importance of prompt tuning. Among soft-prompting methods, PromptEmbedder outperforms Prompt-Tuning across all four backbones by a consistent and significant margin (e.g., +10.3 on Llama-3.2-1B, +5.5 on Qwen2.5-7B, +3.6 on Llama-3.2-3B). We attribute this advantage to the aid of the prompting LLM, which generates dynamic and semantically rich tokens that more effectively steer the frozen backbone for generating effective embeddings.

PromptEmbedder achieves competitive performance to LoRA fine-tuning without modifying the backbone. Without any parameter updates to the backbone LLM, PromptEmbedder remains within 2.6 average scores of LoRA across all backbones. On Llama-3.2-3B and Qwen2.5-7B, the gap narrows to just 0.1 and 0.3 average scores, respectively. This demonstrates that inserting only a few soft tokens is sufficient to achieve performance comparable to weight-level adaptation while fully preserving the generality of the backbone.

5.3 Comparison with Off-the-shelf Embedding Models

We further compare PromptEmbedder among representative off-the-shelf embedding models on MTEB, including encoder-based models (E5-Large, GTE-Large), and LLM-based methods (LLM2Vec, E5-Mistral, GRITLM). As illustrated in Table 2, PromptEmbedder using Mistral-7B backbone outperforms all encoder-based baselines and LLM2Vec, achieving a superior average score of 65.2. Remarkably, even without any parameter updates to the backbone, PromptEmbedder remains on par with state-of-the-art models like E5-Mistral (66.3) and GRITLM (66.9). This performance is particularly noteworthy as it demonstrates that our frozen-backbone paradigm can rival heavily fine-tuned models.

Method	Retr.	Rerank.	Clust.	PairClass.	Class.	STS	Summ.	Avg.
E5-Large	49.3	45.7	45.2	86.1	76.4	80.7	32.3	62.8
GTE-Large	53.3	47.8	48.2	85.1	75.5	83.3	32.9	64.8
LLM2Vec	51.3	47.7	44.1	88.0	79.7	83.7	31.1	64.6
E5-Mistral†	55.2	49.2	54.3	85.7	84.4	74.0	36.3	66.3
GRITLM†	55.4	48.7	54.6	86.3	84.9	75.9	36.1	66.9
PromptEmbedder	50.4	49.8	55.2	84.1	79.8	78.3	34.8	65.2

Table 2: Comparison of PromptEmbedder with Mistral-7B backbone against off-the-shelf embedding models on MTEB. † indicates reproduced results with the same training data to ensure a fair baseline comparison.

6 Embedding Model Transferability

6.1 Cross-Architecture Transferability

We evaluate cross-architecture transferability by comparing PromptEmbedder transfer against Prompt-Tuning transfer across different source to target backbone settings. For reference, we also report LoRA fine-tuning performance trained from scratch on each target backbone. Results are presented in Table 3, where we highlight three key observations. **1) PromptEmbedder transfers sub-**

Source LLM	Method	Target LLM				Avg.	Δ vs. LoRA
		Llama-3.2 (3B)	Llama-3.1 (8B)	Mistral (7B)	Qwen2.5 (7B)		
Training-free Prompting							
—	PromptEOL	49.11	49.40	48.86	50.04	49.35	−16.21
—	Echo	45.74	44.42	43.60	39.57	43.26	−22.30
In-model Fine-tuning							
—	LoRA	65.19	66.56	66.32	64.18	65.56	—
Cross-model Transfer							
Llama-3.1 (8B)	Prompt-Tuning	60.79	63.38	64.26	57.93	61.59	−3.97
	PromptEmbedder	**63.16**	**64.86**	**64.87**	**63.97**	**64.22**	**−1.34**
Mistral (7B)	Prompt-Tuning	60.68	60.49	63.64	51.72	59.13	−6.43
	PromptEmbedder	**62.92**	**64.19**	**65.22**	**63.11**	**63.86**	**−1.70**
Qwen2.5 (7B)	Prompt-Tuning	59.71	62.10	**64.90**	58.38	61.27	−4.29
	PromptEmbedder	**63.08**	**64.08**	64.14	**63.89**	**63.80**	**−1.76**

Table 3: Cross-model transferability on MTEB (English, v2). Underlined values represent the non-transfer cases (i.e., train from scratch). **Bold** indicates the superior performance among the cross-model transfer methods. **Avg.** is computed over all four target LLMs for fairness. **Δ vs. LoRA** measures the gap to the LoRA fine-tuning average.

stantially better than Prompt-Tuning. Across all transfer settings, PromptEmbedder consistently outperforms Prompt-Tuning transfer by a large margin. The average gap to LoRA is reduced from 3.97 to 6.43 average scores, down to just 1.34 to 1.76 average scores. This demonstrates that the auxiliary prompting LLM learns more transferable representations than conventional soft prompts. **2) Transfer performance closely matches training from scratch.** The performance drop when transferring to a different model backbone is minimal. When using Mistral-7B as the target, PromptEmbedder achieves 65.22 when trained from scratch, and still maintains 64.87 and 64.14 when transferred from Llama-3.1 and Qwen2.5, respectively. Notably, PromptEmbedder's average performance remains within 1.5% of LoRA, which requires fine-tuning on the target model from scratch. This shows that PromptEmbedder learns generalizable representations that are agnostic to the source LLM used during initial training.

6.2 Transfer Efficiency on Training Cost

Beyond superior transfer performance, PromptEmbedder offers a significant advantage in computational efficiency during cross-model transfer. To evaluate this, we perform a cross-model transfer from a Llama3-8B source model to a Qwen2.5-7B target model. We compare the wall-clock training time of PromptEmbedder against LoRA and Prompt-Tuning by tracking MTEB performance over training hours.[2]

As illustrated in Figure 3, PromptEmbedder demonstrates two efficiency gains. First,

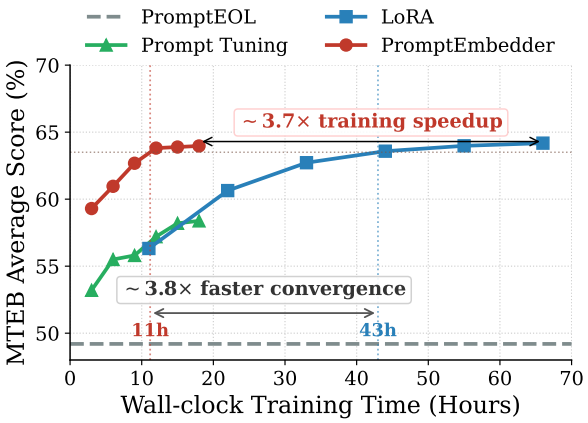

Figure 3: Training efficiency comparison: MTEB performance vs. wall-clock time. PromptEmbedder achieves a **3.7× speedup** over LoRA in reaching comparable performance levels.

PromptEmbedder achieves **3.8× faster convergence** compared to LoRA. Specifically, PromptEmbedder reaches an MTEB score of 63.81 after only 11 hours of training and plateaus shortly thereafter, whereas LoRA requires 43 hours to achieve a comparable performance level of 63.58. We attribute this accelerated convergence to the embedding knowledge stored within $\mathcal{M}_{\text{prompt}}$, which can be effectively transferred to arbitrary LLM backbones without extensive retraining. Second, PromptEmbedder delivers a **3.7× training time speedup** per epoch compared to LoRA. In our experiments, PromptEmbedder requires only 18 hours to iterate over the entire training dataset, while LoRA requires 66 hours to complete the same process. This speedup stems from the lightweight fine-tuning of the adaptation matrix $\mathbf{W}_{\text{adp}}$. These results demonstrate that PromptEmbedder significantly reduces the computational overhead of cross-model transfer while maintaining high performance.

[2]To ensure a fair and reproducible comparison, all experiments are conducted on a single NVIDIA RTX 3090 GPU.

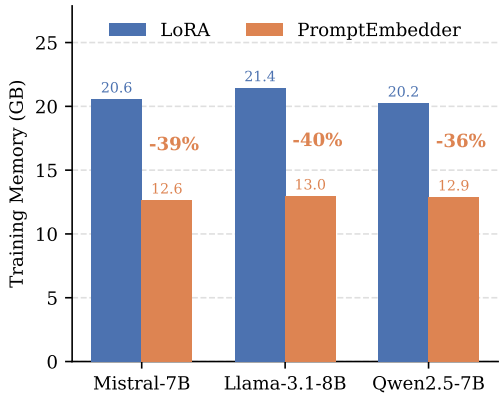

Figure 4: Comparison of peak GPU memory usage between LoRA and PromptEmbedder across three representative LLM backbones.

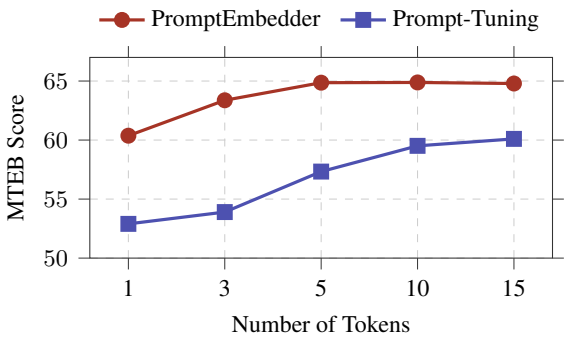

Figure 5: MTEB performance by varying the number of soft prompting tokens with Llama-8B backbone.

	Prompting LLM $\mathcal{M}_{\text{prompt}}$			
Pretrained LLM $\mathcal{M}_{\text{pretrained}}$	**0.6B**	**1.7B**	**4B**	**8B**
Mistral-7B	65.23	65.48	65.31	65.72
Llama-8B	64.86	64.92	64.66	64.71

Table 4: MTEB scores (%) across different combinations of Prompting LLM and pretrained LLM. The Prompting LLM size is varied from 0.6B to 8B using the Qwen3 family (Yang et al., 2025).

7 Discussion

7.1 Memory Efficiency

We evaluate the peak GPU memory consumption of PromptEmbedder during training and compare it against LoRA across three representative LLM backbones. As illustrated in Figure 4, PromptEmbedder reduces training memory by 36% to 40% across all backbones. This significant reduction is primarily attributed to the fact that no gradients are propagated through the pretrained LLM $\mathcal{M}_{\text{pretrained}}$. Combined with the training time efficiency demonstrated in Section 6.2, these results validate PromptEmbedder as a resource-efficient alternative to LoRA fine-tuning.

7.2 Number of Prompting Tokens

We investigate how the number of soft tokens affects the embedding quality of both PromptEmbedder and Prompt-Tuning. We varied the number of tokens $k \in \{1, 3, 5, 10, 15\}$ and summarized the results in Figure 5. In general, performance improves consistently as the number of soft tokens increases for both methods. However, PromptEmbedder consistently outperforms Prompt-Tuning across different numbers of soft tokens. For instance, with only 5 soft tokens, PromptEmbedder achieves an MTEB score of 64.86, compared to 57.40 for Prompt-Tuning. Furthermore, PromptEmbedder achieves near-optimal performance at k=5, with negligible gains beyond this point. Based on these results, we set k=5 as the default configuration in all experiments.

7.3 Choice of Prompting LLM

To investigate the sensitivity of PromptEmbedder to the capacity of the prompting LLM $\mathcal{M}_{\text{prompt}}$, we evaluate the performance of PromptEmbedder using different scales of pretrained LLMs from the Qwen3 family, spanning from 0.6B to 8B parameters. As shown in Table 4, we observe remarkable performance stability across different scales of $\mathcal{M}_{\text{prompt}}$. For instance, scaling the prompting model from 0.6B to 8B yields only a marginal improvement of 0.49% for the Mistral-7B backbone. This finding demonstrates that the generation of effective soft prompts does not necessitate a large-scale LLM. Consequently, lightweight models (e.g., 0.6B) can be utilized to maintain high efficiency without compromising the quality of embeddings.

8 Conclusion

We introduce PromptEmbedder, a dual-LLM framework that separates embedding-specific knowledge from the pretrained LLMs. By using an auxiliary Prompting LLM to generate differentiable soft prompts, PromptEmbedder enables high-quality representation learning with a frozen pretrained LLM and efficient transferability. Extensive experiments show that PromptEmbedder outperforms existing prompting-based methods and achieves performance comparable to LoRA-based fine-tuning while significantly improving training efficiency. We believe PromptEmbedder offers a practical and scalable paradigm for LLM-based text embeddings.

Limitations

While PromptEmbedder substantially closes the gap to LoRA fine-tuning, a small performance difference of up to 2.6 average score remains on certain backbones. We note that this gap is a direct consequence of keeping $\mathcal{M}_{\text{pretrained}}$ entirely frozen, a deliberate design choice that enables cross-architecture transferability and significant reductions in memory and training cost. Additionally, the current cross-architecture transfer setting requires a small amount of labeled data to train the linear alignment layer. Exploring unsupervised or zero-shot transfer to unseen backbones is a promising direction for future work.

Acknowledgments

This material is based upon work supported by National Science and Technology Council, ROC under grant number 114-2221-E-002-134-MY3, NTU AI Center of Research Excellence within Taiwan Centers of Excellence in Artificial Intelligence, and by National Taiwan University and Academia Sinica Innovative Joint Program, under grant AS-NTU-114-06.

References

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*.

Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, Tamay Besiroglu, and David Owen. 2024. The rising costs of training frontier ai models. *arXiv preprint arXiv:2405.21015*.

Jingcheng Deng, Zhongtao Jiang, Liang Pang, Zihao Wei, Liwei Chen, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. 2025. Following the autoregressive nature of llm embeddings via compression and alignment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 12672–12688.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzeminski, Genta Indra Winata, and 1 others. 2025. Mmteb: Massive multilingual text embedding benchmark. In *International Conference on Learning Representations*. International Conference on Learning Representations.

Yingchaojie Feng, Yiqun Sun, Yandong Sun, Minfeng Zhu, Qiang Huang, Anthony Kum Hoe Tung, and Wei Chen. 2025. Don't reinvent the wheel: Efficient instruction-following text embedding based on guided space transformation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24511–24525.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

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. In *The Thirteenth International Conference on Learning Representations*.

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)*, pages 10141–10157, Bangkok, Thailand. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059.

Chaofan Li, Minghao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Defu Lian, Yingxia Shao, and Zheng Liu. 2025. Making text embedders few-shot learners. In *The Thirteenth International Conference on Learning Representations*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

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

Qijiong Liu, Nuo Chen, Tetsuya Sakai, and Xiao-Ming Wu. 2024. Once: Boosting content-based recommendation with both open-and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 452–461.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Niklas Muennighoff, SU Hongjin, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Qwen, :, 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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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*.

Yu-Che Tsai, Kuan-Yu Chen, Yuan-Chi Li, Yuan-Hao Chen, Ching-Yu Tsai, and Shou-De Lin. 2026. Let LLMs speak embedding languages: Generative text embeddings via iterative contrastive refinement. In *The Fourteenth International Conference on Learning Representations*.

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

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple techniques for enhancing sentence embeddings in generative language models. In *International Conference on Intelligent Computing*, pages 52–64.

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

A Use-of-LLMs

In this work, we utilized large language models (LLMs) as part of the core research methodology. Specifically, we fine-tuned existing open-source LLMs (e.g., LLaMA-3 and Mistral) to develop embedding models. These pre-trained models served as the foundation for our experiments, and our main contributions build upon their architectures and representations. Additionally, a LLM-based assistant was used for minor writing support, including grammar checking and improving manuscript readability. All decisions regarding research design, fine-tuning strategies, experimental setup, and final interpretations were made solely by the authors.

B Implementation Details

For both PromptEmbedder and the baseline models we re-implement for comparison, we use nearly identical fine-tuning hyperparameters across different model sizes. We use the AdamW optimizer with a learning rate of 1e-4 and a warmup ratio of 0.03. We adopt Low-Rank Adaptation (LoRA) (Hu et al., 2022) for efficient fine-tuning, setting the rank to 64 and the scaling factor α to 16. Due to limited computational resources, we train with a batch size of 2 and accumulate gradients over 8 steps, resulting in an effective batch size of 16. All models are fine-tuned for a single epoch. For prompt-tuning baseline, we optimize 20 virtual tokens throughout the experiment.

C Evaluation on Instruction Following Benchmark

Table 5 provides the instruction following performance breakdown across STS, Triplet Alignment, and Clustering tasks. The results demonstrate that PromptEmbedder consistently outperforms all frozen-backbone baselines, including training-free prompting and Prompt-Tuning, across all four evaluated architectures. For instance, on the LLaMA-3.2-3B backbone, PromptEmbedder achieves an average score of 44.37, representing a significant improvement over the prompt-tuning method with a score of 36.71. Most notably, PromptEmbedder exhibits high competitiveness even when compared against LoRA. Using LLaMA-3.2-1B as backbone, PromptEmbedder reaches an average of 44.07, actually surpassing LoRA (43.65). For larger-scale models such as Qwen2.5-7B, PromptEmbedder remains remarkably robust with a score of 42.20, narrowing the performance gap with LoRA (43.12) to within 1 point.

D Full Dataset Instructions

This section provides the complete set of natural language instructions used with the datasets in our experiments. We include the instructions for the MTEB benchmark datasets. These instructions define the intended task for each dataset and serve as the input prompts during embedding evaluation. The full text of the instructions is listed in Table 6.

This section provides the complete set of natural language instructions used with the datasets in our experiments. We include the instructions for both the MTEB benchmark datasets and instruction-following embedding tasks. These instructions define the intended task or target aspect for each dataset and serve as the input prompts during embedding evaluation. The full text of the instructions is listed in Table 6 and Table 7.

	Backbone: LLaMA-3.2-1B								***Backbone: LLaMA-3.2-3B***							
	STS ↑			Triplet Align. ↑			Clust. ↑		STS ↑			Triplet Align. ↑			Clust. ↑	
Method	PC	MH	BP	IE	Tox.	AG	NYC	Avg.	PC	MH	BP	IE	Tox.	AG	NYC	Avg.
Training-free Prompting																
PromptEOL	24.61	9.94	23.65	15.26	52.29	68.10	2.77	25.52	24.37	11.21	23.48	11.08	53.79	71.00	1.87	24.90
Pretend-CoT	21.41	10.71	22.27	10.70	54.15	66.21	3.21	24.58	26.82	15.41	21.25	17.37	54.27	72.87	2.73	26.68
Echo	**33.13**	6.14	20.74	20.74	52.68	76.13	37.18	37.82	36.37	6.63	22.38	28.38	52.28	75.74	39.90	39.51
Frozen Backbone Methods																
Prompt-Tuning	29.23	3.93	20.21	37.05	**54.63**	72.80	4.96	29.64	**39.50**	**16.70**	26.47	57.18	54.35	80.62	3.40	36.71
PromptEmbedder	27.74	**13.30**	**24.85**	**53.42**	54.59	**83.62**	**42.12**	**44.07**	28.16	8.86	**26.92**	**59.15**	**59.58**	**84.17**	**39.75**	**44.37**
Fine-tuning																
LoRA	30.61	16.76	24.85	53.56	60.71	80.37	37.72	43.65	42.56	26.29	26.91	42.04	56.27	80.92	39.95	44.53

	Backbone: Qwen2.5-7B								***Backbone: Mistral-7B***							
	STS ↑			Triplet Align. ↑			Clust. ↑		STS ↑			Triplet Align. ↑			Clust. ↑	
Method	PC	MH	BP	IE	Tox.	AG	NYC	Avg.	PC	MH	BP	IE	Tox.	AG	NYC	Avg.
Training-free Prompting																
PromptEOL	28.92	11.09	26.23	16.76	54.94	78.57	2.21	31.25	26.14	14.07	21.70	21.82	53.77	72.63	2.31	26.91
Pretend-CoT	28.47	14.63	25.67	17.94	56.07	76.77	5.14	32.10	30.31	19.26	21.17	20.33	53.94	71.81	3.20	28.84
Echo	26.10	5.32	21.74	14.96	53.37	78.54	**25.94**	32.28	28.41	6.00	19.79	22.24	**56.41**	76.11	**31.57**	34.54
Frozen Backbone Methods																
Prompt-Tuning	33.97	**18.26**	**27.38**	**57.59**	52.81	81.91	5.08	39.57	**39.32**	**21.49**	**23.91**	53.10	55.55	81.60	2.08	35.03
PromptEmbedder	**38.44**	15.46	25.90	46.65	**58.95**	**85.36**	24.66	**42.20**	21.72	7.40	18.60	**65.01**	53.78	**82.24**	23.85	**37.87**
Fine-tuning																
LoRA	39.52	18.84	26.15	45.30	58.12	85.90	28.01	43.12	44.26	15.25	18.83	35.33	60.09	84.81	58.76	44.03

Table 5: Evaluation on instruction-following embedding tasks across four LLM backbones, covering Semantic Textual Similarity (STS), Triplet Alignment, and Clustering. **Bold** denotes the best score among frozen-backbone methods. Dataset abbreviations: PC (PaperCode), MH (MultiHate), BP (Big Patent), IE (IntentEmo), Tox. (Toxic), AG (AG-News), NYC (NYTClust).

Table 6: Instructions for the corresponding datasets in the MTEB benchmark. We mainly follow the instructions from the **GritLM** paper. Note that for retrieval and reranking datasets, queries (Q) and corpus (C) documents may require different instructions, denoted as {dataset}-**Q** and {dataset}-**C**, respectively. For datasets with query instructions only (i.e., {dataset}-**Q**), no instructions are applied to the corpus.

Dataset	**Instruction**
SummEvalSummarization	Given a news summary, retrieve other semantically similar summaries.
ArXivHierarchicalClusteringP2P	Identify the main and secondary category of Arxiv papers based on the titles and abstracts.
ArXivHierarchicalClusteringS2S	Identify the main and secondary category of Arxiv papers based on the titles.
Touche2020Retrieval.v3-**Q**	Given a question, retrieve passages that answer the question.
ClimateFEVERHardNegatives-**Q**	Given a claim about climate change, retrieve documents that support or refute the claim.
FEVERHardNegatives-**Q**	Given a claim, retrieve documents that support or refute the claim.
HotpotQAHardNegatives-**Q**	Given a multi-hop question, retrieve documents that can help answer the question.
AmazonCounterfactualClassification	Classify a given Amazon customer review text as either counterfactual or not-counterfactual.
AmazonPolarityClassification	Classify Amazon reviews into positive or negative sentiment.
AmazonReviewsClassification	Classify the given Amazon review into its appropriate rating category.
Banking77Classification	Given a online banking query, find the corresponding intents.
EmotionClassification	Classify the emotion expressed in the given Twitter message into one of the six emotions: anger, fear, joy, love, sadness, and surprise.
ImdbClassification	Classify the sentiment expressed in the given movie review text from the IMDB dataset.
MassiveIntentClassification	Given a user utterance as query, find the user intents.
MassiveScenarioClassification	Given a user utterance as query, find the user scenarios.
MTOPDomainClassification	Classify the intent domain of the given utterance in task-oriented conversation.
MTOPIntentClassification	Classify the intent of the given utterance in task-oriented conversation.
ToxicConversationsClassification	Classify the given comments as either toxic or not toxic.
TweetSentimentExtractionClassification	Classify the sentiment of a given tweet as either positive, negative, or neutral.
ArxivClusteringP2P	Identify the main and secondary category of Arxiv papers based on the titles and abstracts.
ArxivClusteringS2S	Identify the main and secondary category of Arxiv papers based on the titles.
BiorxivClusteringP2P	Identify the main category of Biorxiv papers based on the titles and abstracts.
BiorxivClusteringS2S	Identify the main category of Biorxiv papers based on the titles.
MedrxivClusteringP2P	Identify the main category of Medrxiv papers based on the titles and abstracts.
MedrxivClusteringS2S	Identify the main category of Medrxiv papers based on the titles.
RedditClustering	Identify the topic or theme of Reddit posts based on the titles.
RedditClusteringP2P	Identify the topic or theme of Reddit posts based on the titles and posts.
StackExchangeClustering	Identify the topic or theme of StackExchange posts based on the titles.
StackExchangeClusteringP2P	Identify the topic or theme of StackExchange posts based on the given paragraphs.
TwentyNewsgroupsClustering	Identify the topic or theme of the given news articles.
SprintDuplicateQuestions	Retrieve duplicate questions from Sprint forum.
TwitterSemEval2015	Retrieve tweets that are semantically similar to the given tweet.
TwitterURLCorpus	Retrieve tweets that are semantically similar to the given tweet.
AskUbuntuDupQuestions-**Q**	Retrieve duplicate questions from AskUbuntu forum.
AskUbuntuDupQuestions-**C**	Retrieve duplicate questions from AskUbuntu forum.
MindSmallReranking-**Q**	Retrieve relevant news articles based on user browsing history.
MindSmallReranking-**C**	Retrieve relevant news articles based on user browsing history.
SciDocsRR-**Q**	Given a title of a scientific paper, retrieve the titles of other relevant papers.
SciDocsRR-**C**	Given a title of a scientific paper, retrieve the titles of other relevant papers.
StackOverflowDupQuestions-**Q**	Retrieve duplicate questions from StackOverflow forum.
StackOverflowDupQuestions-**C**	Retrieve duplicate questions from StackOverflow forum.
ArguAna-**Q**	Given a claim, find documents that refute the claim.
ClimateFEVER-**Q**	Given a claim about climate change, retrieve documents that support or refute the claim.
CQADupstackRetrieval-**Q**	Given a question, retrieve detailed question descriptions from Stackexchange that are duplicates to the given question.
DBPedia-**Q**	Given a query, retrieve relevant entity descriptions from DBPedia.
FEVER-**Q**	Given a claim, retrieve documents that support or refute the claim.
FiQA2018-**Q**	Given a financial question, retrieve user replies that best answer the question.
HotpotQA-**Q**	Given a multi-hop question, retrieve documents that can help answer the question.
MSMARCO-**Q**	Given a web search query, retrieve relevant passages that answer the query.
NFCorpus-**Q**	Given a question, retrieve relevant documents that best answer the question.
NQ-**Q**	Given a question, retrieve Wikipedia passages that answer the question.

Continued on next page

Table 6 – continued from previous page

Dataset	Instruction
QuoraRetrieval-**Q**	Given a question, retrieve questions that are semantically equivalent to the given question.
SCIDOCS-**Q**	Given a scientific paper title, retrieve paper abstracts that are cited by the given paper.
SciFact-**Q**	Given a scientific claim, retrieve documents that support or refute the claim.
Touche2020-**Q**	Given a question, retrieve detailed and persuasive arguments that answer the question.
TRECCOVID-**Q**	Given a query on COVID-19, retrieve documents that answer the query.
STS12	Retrieve semantically similar text.
STS13	Retrieve semantically similar text.
STS14	Retrieve semantically similar text.
STS15	Retrieve semantically similar text.
STS16	Retrieve semantically similar text.
STS17	Retrieve semantically similar text.
STS22	Retrieve semantically similar text.
BIOSSES	Retrieve semantically similar text.
SICK-R	Retrieve semantically similar text.
STSBenchmark	Retrieve semantically similar text.
SummEval	Given a news summary, retrieve other semantically similar summaries.
CQADupstackTexRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Tex StackExchange forum.
CQADupstackTexRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Tex StackExchange forum.
CQADupstackWebmastersRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Webmasters StackExchange forum.
CQADupstackWebmastersRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Webmasters StackExchange forum.
CQADupstackEnglishRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the English StackExchange forum.
CQADupstackEnglishRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the English StackExchange forum.
CQADupstackGamingRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Gaming StackExchange forum.
CQADupstackGamingRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Gaming StackExchange forum.
CQADupstackGisRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Gis StackExchange forum.
CQADupstackGisRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Gis StackExchange forum.
CQADupstackUnixRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Unix StackExchange forum.
CQADupstackUnixRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Unix StackExchange forum.
CQADupstackMathematicaRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Mathematica StackExchange forum.
CQADupstackMathematicaRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Mathematica StackExchange forum.
CQADupstackStatsRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Stats StackExchange forum.
CQADupstackStatsRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Stats StackExchange forum.
CQADupstackPhysicsRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Physics StackExchange forum.
CQADupstackPhysicsRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Physics StackExchange forum.
CQADupstackProgrammersRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Programmers StackExchange forum.
CQADupstackProgrammersRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Programmers StackExchange forum.
CQADupstackAndroidRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Android StackExchange forum.
CQADupstackAndroidRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Android StackExchange forum.
CQADupstackWordpressRetrieval-**Q**	Represent the title of a user question to find a duplicate user question title with body from the Wordpress StackExchange forum.
CQADupstackWordpressRetrieval-**C**	Represent the question title with body posted by a user to find a duplicate user question title from the Wordpress StackExchange forum.

Table 7: Instructions for the corresponding datasets in our instruction-following embedding tasks.

Dataset	**Aspect**	**Instruction**
PaperCode	Method	Represent the paper based on the method used.
	Task	Represent the paper abstract based on the research task.
MultiHate	Hateful	Represent the text based on whether it is hateful or not.
	Language	Represent the text based on the language.
Big Patent	Patent Category	Represent the text based on the category the patent belongs to according to the Cooperative Patent Classification (CPC) code.
IntentEmo	Intent	Represent the intent of this text.
	Emotion	Represent the emotion of this text.
Toxic	Toxic	Represent the text based on whether it is toxic (yes/no).
AG-News	Topic	Represent the news according to their topic category.
NYTClust	Topic	Represent the text based on the main news category.
	Location	Represent the text based on where the news happen.