Resource-Efficient Adaptation of Large Language Models for Text Embeddings via Prompt Engineering and Contrastive Fine-tuning

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have become a cornerstone in Natural Language Processing (NLP), achieving impressive performance in text generation. Their token-level representations capture rich, human-aligned semantics. However, pooling these vectors into a text embedding discards crucial information. Nevertheless, many non-generative downstream tasks, like e.g. clustering, classification, or retrieval, still depend on accurate and controllable sentence- or document-level embeddings. We explore several adaptation strategies for pre-trained, decoder-only LLMs: (i) various aggregation techniques for token embeddings, (ii) task-specific prompt engineering, and (iii) text-level augmentation via contrastive fine-tuning. Combining these components yields competitive performance on the English clustering track of the Massive Text Embedding Benchmark (MTEB). An analysis of the attention map further shows that fine-tuning shifts focus from prompt tokens to semantically relevant words, indicating more effective compression of meaning into the final hidden state. Our experiments demonstrate that LLMs can be effectively adapted as text embedding models through a combination of prompt engineering and resource-efficient contrastive fine-tuning on synthetically generated positive pairs.

1 Introduction

In recent years, Large Language Models (LLMs) have emerged as the state-of-the-art in Natural Language Processing (NLP). Owing to their massive amount of training data and parameters, they excel at a wide range of tasks such as reasoning and text generation. Although primarily trained for next-token prediction using final hidden states and pooling layers, LLMs have increasingly influenced the field of text embedding generation, despite requiring modifications such as removing task-specific output layers.

Text embeddings are vector representations that capture the core semantics of a text. They serve as input for a variety of downstream tasks, including information retrieval, text clustering, recommendation, topic modeling, and Retrieval-Augmented Generation (RAG), as outlined in (Zhang et al., 2024).

Early approaches to text embeddings involved simply averaging over word embeddings obtained from traditional methods such as GloVe (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013). A major advancement came with Sentence-BERT (SBERT) (Reimers and Gurevych, 2019), which generates sentence embeddings from BERT using Siamese and triplet network architectures.

A common approach to improve model performance in an unsupervised setting is contrastive learning, which fine-tunes models to generate more semantically meaningful and discriminative embeddings. In SimCSE, the authors leveraged dropout to create the positive pairs for contrastive learning, demonstrating strong performance on semantic similarity tasks using BERT (Gao et al., 2022). PromptBERT (Jiang et al., 2022) combines prompting with contrastive learning to enhance BERT (Devlin et al., 2019), achieving performance beyond that of SimCSE.

Building on BERT, Stankevičius and Lukoševičius summarize and evaluate numerous existing methods for Semantic Textual Similarity (STS), text clustering, and classification. Their study explores information extraction from various layers, experiments with different pooling strategies, and applies post-processing techniques. Another notable method is Diagonal Attention Pooling (DITTO)(Chen et al., 2023), which addresses the bias of BERT-based sentence embeddings toward uninformative words.

Recent advancements in text embedding have been primarily driven by two main fields: prompt engineering and fine-tuning. This paper builds

upon both approaches. One example of interest in the evaluation of text embeddings is the Massive Text Embedding Benchmark (MTEB) Muennighoff et al., which collects publicly available datasets and downstream tasks, including clustering, classification, retrieval, STS, and many more. Completing this, a recent survey by (Cao, 2024) investigates the advancements that have contributed to the best-performing models on MTEB. A key strength of MTEB lies in its large diversity of datasets and task types, which demand general and robust embeddings capable of performing well across varied settings. We are especially interested in improving the clustering performance of English data due to its application to product descriptions.

Our contributions can be summarized as follows:

- **A resource-efficient pipeline:** We combine clustering-oriented prompts with LoRA-based contrastive fine-tuning to transform small LLMs into text-embedding models on a single 24 GB GPU.
- **Systematic evaluation:** We benchmark state-of-the-art STS prompts, introduce clustering-specific prompts, and compare synthetic data-augmentation strategies with a human-curated Natural Language Inference (NLI) dataset.
- **Competitive performance:** The resulting models achieve competitive accuracy on English clustering tasks on MTEB, and the method generalizes well to classification tasks.
- **Interpretability:** Attention-map analysis reveals how fine-tuning shifts the model's focus toward semantically relevant tokens.

All source code, synthetic training data, and model checkpoints used in this study will be made publicly available.

2 Related Work

As already mentioned, LLMs have recently become increasingly popular for text embedding due to their large-scale pretraining. BehnamGhader et al. proposed LLM2Vec, an unsupervised pipeline that creates an embedding model from any decoder-only LLM by enabling bidirectional attention, followed by masked token prediction and unsupervised contrastive learning. By incorporating supervised learning, they achieved state-of-the-art performance on MTEB for a period. First, the authors enhanced the attention mechanism of the LLM by enabling bidirectional attention. They then retrained the model using masked token prediction to adapt it to this new attention scheme. Finally, the authors applied contrastive learning to embeddings obtained through mean-pooling of all attention outputs, fine-tuning the model directly in the embedding space to produce meaningful text representations. Besides fine-tuning, prompting has also become increasingly popular. Jiang et al. use prompts to reshape the embedding space, focusing particularly on encouraging the model to condense a sentence into a single word — example given, *"This sentence: "[X]" means in one word:"* (Jiang et al., 2023). Building on this work, Zhang et al. also leverage LLMs and prompt engineering, testing new prompts and showing that, in a zero-shot inference setting, enforcing an explicit one-word limitation (EOL) improves performance, and the prompt choice significantly affects results. When using efficient fine-tuning via contrastive learning on a human-annotated NLI dataset, they show that the prompt has little impact on STS tasks. They further introduce new zero-shot prompts, such as *After thinking step by step, this sentence: "[X]" means in one word:"* (Zhang et al., 2024), which they refer to *Pretended Chain of Thought*. They also present a second prompt called *Knowledge Enhancement*. Qwen3 Embedding has been recently released, comprising a family embedding and reranking models (Zhang et al., 2025). The models explicitly use an EOS token, leveraging its final hidden state as the text embedding. In their technical report, the authors describe the training pipeline as follows: (1) weakly supervised pretraining using large-scale synthetic paired data, (2) supervised fine-tuning with high-quality synthetic and labeled data, and (3) model merging using sampled checkpoints from the previous step (Zhang et al., 2025). The synthetic data was generated using the Qwen3-32B model, resulting in 150 million pairs for multi-task weakly supervised learning. Filtering the synthetic data by cosine similarity, they reduced it to 12 million high-quality supervised training data pairs. Their methods were evaluated on MTEB (Muennighoff et al., 2023), achieving state-of-the-art performance.

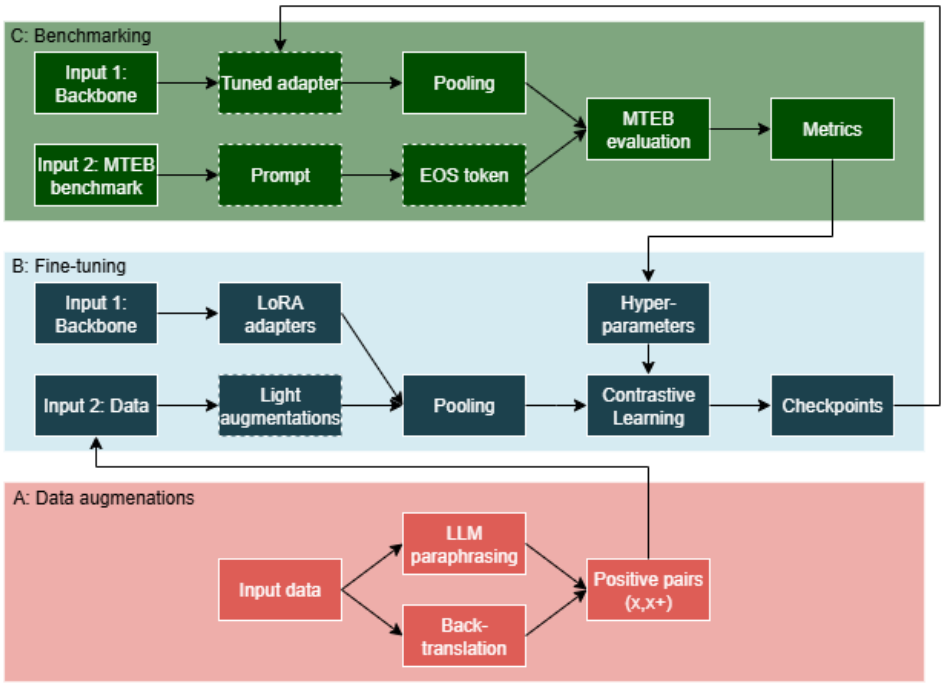

Figure 1: Overview of the proposed pipeline. It comprises three stages: (A) data augmentation via back-translation and LLM paraphrasing, (B) LoRA fine-tuning with contrastive learning, and (C) benchmarking on MTEB with optional prompts, tuned adapters, and EOS tokens. Dashed boxes mark optional components.

3 Methods

Our methodology follows a three-stage workflow visualized in Figure 1. First, we conduct a zero-shot evaluation of token aggregation strategies over pretrained LLM representations, using the English clustering subset of MTEB. Second, we shape the embedding space through prompting by wrapping clustering-oriented prompts around each input, encouraging representations that better capture text-level semantics. Third, we apply LoRA-based contrastive fine-tuning by adding low-rank adapters to the frozen backbone and optimizing them with a contrastive loss (Hu et al., 2021). During this stage, we compare several positive-pair generation strategies to better align the embeddings produced under the selected prompt.

3.1 Embedding Extraction & Pooling

Given an input sequence, we extract the hidden states from the final transformer layer and derive a sentence vector using one of three token aggregation strategies:

- **Mean Pooling** — Average all token hidden states.
- **Last-Token Pooling** — Use the hidden states of the final token in the sequence.
- **EOS-Token Pooling** — Append an explicit end-of-sequence (EOS) token and use its hidden state.

In zero-shot evaluation, EOS-token pooling consistently underperformed compared to the implicit last-token variant. As a result, all subsequent analyses focus on mean and last-token pooling. Mean pooling aggregates information across all positions, while last-token pooling leverages the auto-regressive objective, which encourages the final hidden state to encode the entire context. Although prior work reports promising performance using explicit EOS pooling (Zhang et al., 2024), our experiments indicate that this benefit does not generalize to the models and clustering tasks experimented in this study.

3.2 Prompt-Based Shaping of the Embedding Space

Prompt engineering is a proven technique for steering LLM representations in STS tasks (Jiang et al., 2023; Zhang et al., 2024). To adapt this idea to clustering, we re-evaluated the STS prompts and introduced cluster-specific variants. Table 1 lists the six prompts considered.

We first replicate the three STS prompts — EOL, Pretended Chain of Thought (PCoT), and

Explicit One-Word Limitation (EOL) This sentence: "[X]" means in one word:
Pretended Chain of Thought (PCoT) After thinking step by step, this sentence: "[X]" means in one word:
Summarize (SUM) This sentence: "[X]" can be summarized as
Cluster Command Wrapped (CCW) This sentence: "[X]" belongs to the following cluster:
Cluster Command Prepended (CCP) Cluster the text: "[X]"
Question Which cluster would you assign the sentence: "[X]" to?

Table 1: Prompts used in the experimental evaluation. EOL, PCoT, and SUM are adopted from prior work Zhang et al., while CCW, CCP, and Question are novel contributions specifically designed for clustering tasks. The abbreviated names are used throughout the paper for brevity.

Summarize (SUM) — as introduced by (Zhang et al., 2024). We then design two clustering-oriented prompts, Cluster Command Wrapped (CCW), Cluster Command Prepended (CCP), and a question-style prompt (Question). CCP follows the instruction format used in Qwen3-Embedding (Zhang et al., 2025), which takes the form `{Instruction} {Query} <|endoftext|>`.

This format places the instruction before the input, allowing the final token to remain content-dependent. In contrast, CCW wraps the instruction around the sentence, fixing the final token (":") and allowing the attention mechanism to propagate context. We omit an explicit `<eos>` token, as earlier pooling experiments showed no benefit. By comparing these alternatives, we assess how instruction format and token position influence the quality of the resulting clustering embeddings.

3.3 Contrastive Fine-Tuning with Augmented Data

The third stage applies parameter-efficient LoRA fine-tuning to (i) adapt the model to the selected prompt and (ii) optimize it to produce the sentence embedding directly, rather than merely generating an intermediate representation for next-token prediction. Following (Zhang et al., 2024), we attach LoRA adapters and train them using a contrastive loss in a self-supervised setting. As an initial source of positive pairs, we adopt the wiki1m_for_simcse dataset, a one-million-sentence Wikipedia corpus dataset used by SimCSE (Gao et al., 2022). The motivation for using this dataset follows (BehnamGhader et al., 2024), which argues that since LLMs have already been trained on Wikipedia data, they will not acquire new domain knowledge, but only learn the fine-tuned technique. Negative pairs are generated by treating all non-positive samples in each training batch as negatives (van den Oord et al., 2019). We then extended our study by applying a suite of natural-language data augmentation techniques (Bayer et al., 2022; Chai et al., 2025; Wang et al., 2025), which are described in the remainder of the section.

- **Translation:** We translate each sentence from English into German and then back into English.
- **Generative AI:** We prompt an LLM to generate positive text pairs.
- **Random Deletion:** We randomly delete each word with a probability of 10%.
- **Random Swap:** We randomly swap two words, performing this operation only once per text.
- **Character-Level Noise:** We simulate typographical errors by randomly swapping two adjacent characters within a word, with a probability of 5%.

We progressively diversified the source of positive pairs as follows. First, we performed back-translation (English → German → English) on the Wikipedia corpus and fine-tuned a model on the resulting positive pairs. We then introduced lexical perturbations to increase surface-level variation. Next, we fine-tuned a model on a human-annotated NLI corpus previously used by Zhang et al. and created by (Gao et al., 2021; Bowman et al., 2015; Williams et al., 2018), which provides high-quality positive pairs. Finally, synthetic positive pairs were generated through LLM-based paraphrasing, using the prompts listed in Table 2, and used them in a separate fine-tuning stage.

4 Results

In this section, we address three questions: (i) Which pooling strategy produces the strongest zero-shot clustering performance? (ii) To what extent do clustering-oriented prompts improve these baselines? (iii) Which combination of fine-tuning, dataset, and augmentation strategy provides the best results? Unless stated otherwise, scores are averaged over the 21 English clustering datasets

Rewrite the following sentence with more detail, keeping the original meaning.Respond with only the rewritten sentence: "[X]"
Paraphrase this sentence using simpler language.Respond with only the rewritten sentence: "[X]"
You're a high school teacher. Explain this sentence to your students in your own words.Respond with only the explanation: "[X]"
Write a metaphor that expresses the same idea as this sentence.Respond with only the metaphor: "[X]"
After reading this sentence, what is a natural question someone might ask?Respond with only the question: "[X]"

Table 2: Prompt templates used to generate synthetic positive pairs from sentences in the Wikipedia corpus. Each prompt elicits a paraphrase or semantically equivalent variant, enriching contrastive training with linguistic diversity.

in MTEB v1.38.30. We also provide classification results to assess the generality of our findings. In line with the established MTEB protocol, we report single-run scores without confidence intervals or variance estimates to ensure direct comparability with prior work. Because MTEB fixes dataset splits, random seeds, and evaluation scripts, repeated evaluations of the same model yield identical results. Experiments were conducted on a single NVIDIA RTX A5000 (24 GB), except for models exceeding one billion parameters, which were tested using multiple GPUs. Our study focuses on two smaller LLMs, namely Llama-3.2-1B (Grattafiori et al., 2024) and Qwen3-0.6B (Yang et al., 2025).

4.1 Embedding Extraction & Pooling

The zero-shot baseline was obtained by comparing three token aggregation strategies across Llama-3.2-1B and Qwen3-0.6B. As shown in Table 3, mean pooling delivers the highest average clustering accuracy, outperforming last-token pooling by approximately 12 % and EOS-token pooling by around 20 % across both models. The Llama model also shows a consistent advantage over Qwen, reflecting its larger parameter count.

Model	Mean	Last	EOS
Llama-3.2-1B	**45.8 %**	33.2 %	23.4 %
Qwen3-0.6B	**39.8 %**	28.9 %	22.8 %

Table 3: Zero-shot clustering accuracy on 21 English datasets from MTEB. Mean pooling consistently outperforms last-token and EOS-token aggregation.

4.2 Prompt-Based Shaping of the Embedding Space

Prompting alters the embedding space unevenly across different pooling strategies, as shown in Table 4. Mean pooling yields marginal or even detrimental effects. Intuitively, that is likely because many tokens remain constant across prompts, pulling the mean embedding toward shared content and outweighing improvements in the original text's token representation. By contrast, last-token pooling benefits substantially. This behavior likely stems from the model's tendency to prepare an answer when prompted, concentrating relevant information in the final token. The fact that semantically equivalent prompts yield significantly different results highlights the limited robustness of current LLMs.

Model	Mean	Last
Llama-3.2-1B	45.8 %	33.2 %
Llama-3.2-1B + EOL	**46.1 %**	43.2 %
Llama-3.2-1B + PCoT	44.6 %	42.8 %
Llama-3.2-1B + SUM	43.9 %	36.7 %
Llama-3.2-1B + CCW	45.3 %	**46.8 %**
Llama-3.2-1B + CCP	43.5 %	32.8 %
Llama-3.2-1B + Question	43.4 %	30.8 %
Qwen3-0.6B	**39.8 %**	28.9 %
Qwen3-0.6B + EOL	38.3 %	26.9 %
Qwen3-0.6B + PCoT	38.2 %	24.3 %
Qwen3-0.6B + SUM	38.5 %	28.0 %
Qwen3-0.6B + CCW	38.6 %	**30.7 %**
Qwen3-0.6B + CCP	38.9 %	29.8 %
Qwen3-0.6B + Question	37.5 %	24.8 %

Table 4: Clustering accuracy with and without prompt-based input formatting. Results are reported using mean and last-token pooling across prompts. The CCW prompt leads to the largest gain when combined with last-token pooling.

On Llama-3.2-1B, the CCW prompt increases last-token accuracy from 33.2 % to 46.8 %, a gain of 13.6 percentage points with barely any additional inference cost. Other instruction-style prompts (EOL, PCoT) also improve performance, while the interrogative variant (Question) degrades it — suggesting that commands produce more consistent representations than questions. Qwen3-0.6B exhibits similar but smaller gains, as none of the tested prompts closes the gap to Llama — likely due to Qwen's lower capacity. We were unable to identify an equally effective prompt for Qwen. The results further suggest that wrapping the prompt

Method	Mean-pooling Llama-3.2-1B	Last-token Llama-3.2-1B	Mean-pooling Qwen3-0.6B	Last-token Qwen3-0.6B
Base	45.8 %	33.2 %	39.8 %	28.9 %
EOL	46.1 %	43.2 %	38.3 %	26.9 %
CCW	45.3 %	46.8 %	38.6 %	30.7 %
CCP	43.5 %	32.8 %	38.9 %	29.8 %
Base + nli	49.2 %	47.8 %	46.8 %	47.4 %
CCW + nli	52.5 %	52.2 %	**49.2 %**	49.6 %
CCP + nli	49.5 %	49.7 %	46.2 %	48.9 %
CCW + wiki-translate	51.2 %	52.2 %	47.1 %	49.4 %
CCP + wiki-translate	47.0 %	45.3 %	44.4 %	43.8 %
CCW + wiki-translate + aug	51.7 %	52.6 %	46.8 %	47.7 %
CCP + wiki-translate + aug	47.2 %	46.5 %	43.4 %	45.6 %
CCW + wiki-prompt	**52.7 %**	**54.5 %**	48.8 %	**50.9 %**
CCP + wiki-prompt	48.4 %	51.8 %	45.3 %	49.0 %

Table 5: Comparison of zero-shot, prompt-based, and fine-tuned clustering performance across various training configurations. Each row corresponds to a specific combination of prompt template and training dataset. The best result for Llama-3.2-1B is obtained using CCW prompt with LLM-generated positive pairs from the wiki-prompt dataset.

around the input may be beneficial.

4.3 Contrastive Fine-Tuning with Augmented Data

For fine-tuning, we constrained the search space to two clustering prompts (CCW and CCP), two pooling strategies (mean and last-token), and the data configuration introduced in the Methods section. Results are summarized in Table 5.

Consistent with prior findings, prompts benefit the last-token variant substantially more than mean pooling.

Training LoRA adapters on the human-curated NLI corpus without a prompt improves performance for both models, yet it does not surpass prompt-based fine-tuning. On Llama, the gap is particularly pronounced — fine-tuning NLI without a prompt achieves 47.8 %, barely outperforming CCW prompting alone at 46.8 %.

Using Wikipedia sentences generated through back-translation, referred to as "wiki-translate", yields lower performance than NLI, and the gap becomes even larger when combined with CCP. Adding lexical and character-level noise as data augmentation leads to modest improvements for Llama but often degrades performance for Qwen.

The best results are obtained using LLM-generated positive pairs, referred to as “wiki-prompt”. When combined with CCW, last-token accuracy reaches 54.5 % for Llama and 50.9 % for Qwen, representing the overall best scores. Mean pooling also performs strongly, peaking at 52.7 % for Llama on the same synthetic dataset. For Qwen, mean pooling achieves its best result on NLI data.

CCW, which wraps the sentence, consistently outperforms CCP across all datasets. Synthetic positive pairs generated by an LLM rival human-annotated NLI samples, while remaining domain-agnostic and highly scalable.

Detailed results for each task, including v_measure scores and standard deviations, are reported in Appendix B.2 for the CCW + wiki-prompt setting using Llama with last-token pooling. Also, all hyperparameter details are provided in Appendix A.

4.4 Comparison to other models on MTEB(eng, v2) Clustering

Table 6 compares our best model against a wide set of available embedding models. Proprietary systems such as *gemini-embedding-001* (Lee et al., 2025b) (59.39 %) and large-scale open models like *gte-Qwen2-7B-instruct* (Alibaba-NLP, 2024) (58.97 %) and *Qwen3-Embedding-8B* (58.57 %) currently define the state of the art. Slightly smaller variants such as *Qwen3-Embedding-4B* (57.51 %) and *GritLM-7B* (Muennighoff et al., 2025) (50.82 %) further illustrate the performance advantages of scaling dedicated embedding models.

Within this landscape, our 1.3B-parameter checkpoint achieves 50.97 %, outperforming general-purpose models of comparable or smaller size such as *GIST-large-Embedding-v0* (Solatorio, 2024) (48.84 %), *multilingual-e5-large-instruct* (Wang et al., 2024) (49.89 %), and *NV-Embed-v2* (Lee et al., 2025a) (47.66 %). Compared to the Qwen3-Embedding family, our approach trails the 4B and 8B variants, but remains close to *Qwen3-Embedding-0.6B*, despite being trained

with far less compute and without task-specific prompting.

Importantly, this performance is achieved with a lightweight LoRA adaptation on Llama-3.2-1B, requiring under one GPU-hour on a single RTX A5000. This demonstrates that competitive clustering quality can be obtained without scaling to multi-billion-parameter embedding-specialized models, reinforcing the efficiency and adaptability of our method.

Models	Param.	Clustering
NV-Embed-v2	7.8B	47.66 %
GIST-large-Embedding-v0	0.3B	48.84 %
multilingual-e5-large-in.	0.6B	49.89 %
GritLM-7B	7.2B	50.82 %
gte-Qwen2-1.5B-instruct	1.5B	53.54 %
gte-Qwen2-7B-instruct	7.6B	58.97 %
Qwen3-Embedding-0.6B	0.6B	54.05 %
Qwen3-Embedding-4B	4B	57.51 %
Qwen3-Embedding-8B	8B	58.57 %
gemini-embedding-001	–	59.39 %
Our best	1.3B	50.97 %

Table 6: Comparison of clustering performance between our best Llama model, the Qwen3-Embedding family, and other competitors. Results are reported for MTEB(eng, v2) Clustering, introduced by Enevoldsen et al..

4.5 Classification as a Downstream Task

To evaluate the generality of our pipeline, we applied it to the English classification subset of MTEB without modifying hyperparameters or prompt templates. Guided by insights from clustering experiments, we used the prompt: *"This sentence: "[X]" can be classified as:"*. Datasets with titles containing legal were excluded due to licensing constraints, and YahooAnswersTopicsClassification was omitted because empty entries triggered input formatting errors with the Qwen tokenizer. Table 7 reports the macro-average accuracy across the remaining tasks.

The observed performance pattern closely mirrors that of the clustering tasks. In the zero-shot setting, mean pooling consistently outperforms last-token pooling, and Llama-3.2-1B maintains a slight advantage over Qwen3-0.6B. Introducing the CCW-style prompt yields modest improvements, again more pronounced for last-token embeddings. Fine-tuning with LoRA using the same hyperparameters as in the clustering experiments increases last-token accuracy to 75.4 % for Llama and 72.5 % for Qwen, establishing this as the best overall configuration. Notably, these improvements are obtained without classification-specific prompt tuning or hyperparameter optimization, highlighting the transferability of the pipeline across task types.

Model	Mean	Last
Llama-3.2-1B	72.5 %	68.1 %
Llama-3.2-1B + p	73.0 %	70.3 %
Llama-3.2-1B + p + ft	74.2 %	75.4 %
Qwen3-0.6B	70.2 %	63.0 %
Qwen3-0.6B + p	70.3 %	64.1 %
Qwen3-0.6B + p + ft	72.4 %	72.5 %

Table 7: Classification accuracy on the English subset of MTEB using the same configuration as in the clustering experiment. No hyperparameter adjustment was done. A classification-style prompt structurally aligned with the best-performing clustering prompt (CCW) was used.

We further evaluate a key advantage of our method — its ability to incorporate domain-specific knowledge through targeted fine-tuning. To this end, we fine-tune Llama-3.2-1B on two MTEB classification datasets with labeled training splits: EmotionClassification and ToxicConversationsClassification(Muennighoff et al., 2023). Based on prior results showing its superiority in fine-tuned settings, we use last-token pooling exclusively. For each task, we construct a domain-specific contrastive training set by prompting an LLM to generate paraphrases for each sentence in the task's training set, thereby creating positive pairs. Fine-tuning on these smaller classification datasets is even faster than in the clustering setting. As before, less than a full epoch is required, with performance typically peaking after approximately 100 optimization steps.

Method	Emotion	Toxic
Last	29.26 %	63.53 %
Last + prompt	42.81 %	63.19 %
Fine-tuned (general)	45.33 %	67.38 %
Fine-tuned (task)	**49.78 %**	**68.93 %**

Table 8: Classification accuracy on EmotionClassification and ToxicConversationsClassification. "Last" denotes the last-token embedding without a prompt; "+ prompt" adds a classification-style prompt. "Fine-tuned (general)" refers to LoRA on the wiki-prompt corpus; "Fine-tuned (task)" uses LLM-generated positive pairs from each task's training set.

Table 8 illustrates a consistent pattern. The classification prompt leads to a substantial improve-

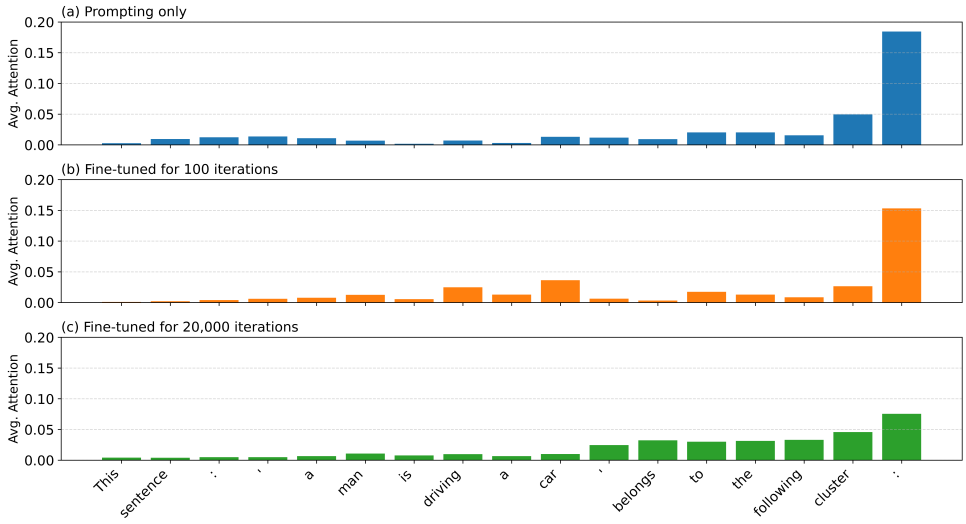

Figure 2: Average attention received by each input token from the final token in the last transformer layer of Llama-3.2-1B, prompted with *"This sentence: 'a man is driving a car' belongs to the following cluster:"*. Bars indicate values for at three stages of training: prompt-only, after 100 fine-tuning steps, and after 20,000 steps. Fine-tuning initially shifts attention to semantically meaningful tokens ("man", "driving", "car"), while prolonged training leads to a more uniform distribution.

ment on the Emotion task but slightly deteriorates for Toxicity. Fine-tuning on the general wiki-prompt corpus enhances performance across both tasks. However, fine-tuning on the task-specific data achieves the highest scores, improving Emotion by 4.5 % and Toxicity by 1.6 % over the general model. While the extent of improvement varies, the addition of task-relevant supervision consistently results in further gains.

4.6 Analysis of Attention

To investigate how LoRA fine-tuning alters the model's internal focus, we follow a method similar to that of Zhang et al.. Attention maps were generated for the probe sentence *"a man is driving a car"* and validated using additional sentences sampled from MTEB. Additional examples and detailed discussion are provided in Appendix B.3. The analysis examines the outgoing attention distribution of the final token in Llama-3.2-1B with the CCW prompt, comparing three settings illustrated in Figure 2: panel (a) shows the zero-shot model, panel (b) presents the best checkpoint after approximately 100 optimization steps, and panel (c) displays an over-trained checkpoint after 20,000 steps.

In all cases, the final token directs most of its attention to itself, a well-known behavior in causal transformers that preserves the accumulated context through residual connections. The key difference lies in the distribution of the remaining attention. After moderate fine-tuning, attention shifts away from the prompt tokens toward the content words man, *driving*, and *car*, emphasizing the sentence's subject, predicate, and object — the elements most informative for downstream semantic tasks. With extended training, the focused attention dissipates; attention becomes more uniformly distributed, and model performance declines accordingly. These observations suggest that an appropriately timed fine-tuning phase encourages the model to encode task-relevant semantics into the final hidden state, while over-training diffuses this information and diminishes embedding quality.

5 Conclusion

This work focuses on turning pretrained LLMs into strong sentence-embedding models. We show that parameter-efficient fine-tuning, such as LoRA steered by a well-chosen prompt, can achieve high-quality embeddings. Additionally, sentence-level data augmentation like synthetic positive pairs enables promising contrastive self-supervised fine-tuning without manually labeled data. We raised Llama-3.2-1B and Qwen3-0.6B to highly competitive performance on the English clustering subset of MTEB, rivaling a dedicated embedding baseline in under an hour on a single RTX A5000. Attention maps indicate a more interpretable adaptation to downstream tasks.

Limitations

The current framework is limited to task-specific prompt formats like clustering or classification, English-only text data, and full-precision models. It does not yet incorporate task-agnostic prompt design, multilingual or cross-modal applications, or quantization methods for reducing resource consumption even further. Furthermore, the attention analysis presented here should not be interpreted as providing causal explanations; instead, it offers descriptive evidence. While the analysis cannot establish definitive cause-and-effect relationships, the consistent patterns observed across a broad range of test sentences increase confidence that the model tends to focus on the input to be encoded, with particular emphasis on the most relevant tokens.

References

Alibaba-NLP. 2024. gte-qwen2-7b-instruct. https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct. Accessed: 2025-09-17.

Markus Bayer, Marc-André Kaufhold, and Christian Reuter. 2022. A survey on data augmentation for text classification. *ACM Computing Surveys*, 55(7):1–39.

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

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. *Preprint*, arXiv:1508.05326.

Hongliu Cao. 2024. Recent advances in text embedding: A comprehensive review of top-performing methods on the mteb benchmark. *Preprint*, arXiv:2406.01607.

Yaping Chai, Haoran Xie, and Joe S. Qin. 2025. Text data augmentation for large language models: A comprehensive survey of methods, challenges, and opportunities. *Preprint*, arXiv:2501.18845.

Qian Chen, Wen Wang, Qinglin Zhang, Siqi Zheng, Chong Deng, Hai Yu, Jiaqing Liu, Yukun Ma, and Chong Zhang. 2023. Ditto: A simple and efficient approach to improve sentence embeddings. *Preprint*, arXiv:2305.10786.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, and et al. 2025. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Datasets-for-simcse: nli_for_simcse.csv. Accessed: 2025-07-22.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings. *Preprint*, arXiv:2104.08821.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023. Scaling sentence embeddings with large language models. *Preprint*, arXiv:2307.16645.

Ting Jiang, Jian Jiao, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Denvy Deng, and Qi Zhang. 2022. Promptbert: Improving bert sentence embeddings with prompts. *Preprint*, arXiv:2201.04337.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025a. Nv-embed: Improved techniques for training llms as generalist embedding models. *Preprint*, arXiv:2405.17428.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, Shanfeng Zhang, Daniel Salz, Michael Boratko, Jay Han, Blair Chen, Shuo Huang, Vikram Rao, Paul Suganthan, and 28 others. 2025b. Gemini embedding: Generalizable embeddings from gemini. *Preprint*, arXiv:2503.07891.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Preprint*, arXiv:1301.3781.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative representational instruction tuning. *Preprint*, arXiv:2402.09906.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. *Preprint*, arXiv:2210.07316.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Preprint*, arXiv:1908.10084.

Aivin V. Solatorio. 2024. Gistembed: Guided in-sample selection of training negatives for text embedding fine-tuning. *Preprint*, arXiv:2402.16829.

Lukas Stankevičius and Mantas Lukoševičius. 2024. Extracting sentence embeddings from pretrained transformer models. *Applied Sciences*, 14(19):8887.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *Preprint*, arXiv:2402.05672.

Zaitian Wang, Pengfei Wang, Kunpeng Liu, Pengyang Wang, Yanjie Fu, Chang-Tien Lu, Charu C. Aggarwal, Jian Pei, and Yuanchun Zhou. 2025. A comprehensive survey on data augmentation. *Preprint*, arXiv:2405.09591.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. pages 1112–1122.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, and et al. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple techniques for enhancing sentence embeddings in generative language models. In *Advanced Intelligent Computing Technology and Applications*, pages 52–64, Singapore. Springer Nature Singapore.

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. *Preprint*, arXiv:2506.05176.

A Hyperparameter-Tuning

Table 9 lists the hyperparameters explored during fine-tuning. The final settings used in our experiments are highlighted in bold. We initially trained LoRA adapters on two attention projections, namely the query and value projections. However, we observed higher scores after extending optimization to include six projections in total. These additional projections also influence the feed-forward pathway and include the output, gate, up, and down projections. This broader parameter coverage is particularly well-suited for embedding tasks, as it allows the model to adapt both the attention mechanism and token-mixing components.

Hyperparameter	Tested values
Dropout	**0.05**, 0.1, 0.2, 0.5
Learning rate	1×10^{-5}, $\mathbf{5 \times 10^{-5}}$, 5×10^{-4}
Batch size	20, 60, **120**, 500
Temperature	1, 0.5, **0.2**, 0.1, 0.05
Projection layers	2, **6**

Table 9: Hyperparameter search space used for LoRA fine-tuning. The tested values for each parameter are listed, with the final configuration highlighted. For projection layers, both two-layer and six-layer settings were evaluated.

All fine-tuning experiments were conducted on a single NVIDIA RTX A5000 GPU with 24 GB of memory. Peak validation performance typically emerged within 0.1 to 0.5 epochs, corresponding to approximately 100 optimization steps or 12,000 examples at the selected batch size of 120. In some cases, performance plateaued later, between 500 and 2,500 steps. To address this, we saved intermediate checkpoints and selected the best-performing model post hoc. In practice, fine-tuning runs completed under 30 minutes, demonstrating the method's efficiency and suitability for low-resource environments. All additional hyperparameters can be found in the code in their final ideal state.

B Results

This appendix section provides supplementary experimental results and analyses that complement the findings in the main text.

B.1 Zero-Shot Comparison by Model Size

In a strictly zero-shot setting, we also evaluated larger checkpoints: Llama-3.2-3B, Qwen3-4B, and Qwen3-8B. Only the CCW prompt was used in this comparison, as shown in Table 10.

Model	Mean	Last
Llama-3.2-1B	45.8 %	33.2 %
Llama-3.2-1B + Prompt	45.3 %	46.8 %
Llama-3.2-3B	45.4 %	33.4 %
Llama-3.2-3B + Prompt	45.5 %	46.8 %
Qwen3-0.6B	39.8 %	28.9 %
Qwen3-0.6B + Prompt	38.6 %	30.7 %
Qwen3-4B	40.8 %	28.3 %
Qwen3-4B + Prompt	40.3 %	46.3 %
Qwen3-8B	35.3 %	26.7 %
Qwen3-8B + Prompt	35.4 %	44.1 %

Table 10: Zero-shot clustering score across different model sizes, with and without the CCW prompt. The results highlight how model scaling and prompt-based input formatting influence performance, both mean and last-token pooling. The CCW prompt is consistently applied.

Scaling Llama from 1 billion to 3 billion parameters yields virtually no change in either pooling metric, suggesting that the Llama-3.2-1B model already saturates clustering performance, for both zero-shot and with prompting. By contrast, increasing Qwen's size from 0.6 billion to 4 billion parameters raises last-token accuracy from 30.7 % to 46.3 %, matching Llama's performance. However, a further increase to 8 billion parameters yields no additional gains in any configuration. Thus, for Qwen, the benefits of scaling plateau before the 8 billion parameter mark, whereas Llama shows minor sensitivity to model size in this task.

B.2 Detailed Performance and Standard Deviation of the Best Model

In Table 11 we report per-dataset v-measure and standard deviation for all 21 English clustering tasks in MTEB v1.38.30. The ± values are not derived from repeated experimental runs, but from the variability induced by MTEB's own evaluation procedure. Specifically, for clustering tasks MTEB fixes the dataset splits and embeddings, but the downstream clustering evaluation involves multiple runs (e.g., different initializations or folds), and the reported standard deviation quantifies this within-task variation. It should therefore be interpreted as a measure of how sensitive the evaluation metric is to these internal clustering runs, not as a statistical confidence interval. The benchmark suite spans diverse domains, including scientific abstracts (ArXiv, BioRxiv, MedRxiv), news (MasakhaNEWS), online forums (Reddit, StackEx-

Dataset/Task	Llama_last
ArXivHierarchicalClusteringP2P	62.71 ± 2.34 %
ArXivHierarchicalClusteringS2S	63.00 ± 4.13 %
BigPatentClustering.v2	32.11 ± 3.44 %
BiorxivClusteringP2P.v2	42.35 ± 0.57 %
BiorxivClusteringS2S.v2	38.80 ± 0.56 %
BuiltBenchClusteringP2P	58.22 ± 12.50 %
BuiltBenchClusteringS2S	50.69 ± 10.74 %
ClusTREC-Covid	84.89 ± 0.45 %
MasakhaNEWSClusteringP2P	64.15 ± 23.64 %
MasakhaNEWSClusteringS2S	59.09 ± 23.70 %
MedrxivClusteringP2P.v2	39.14 ± 0.78 %
MedrxivClusteringS2S.v2	37.21 ± 0.68 %
RedditClustering.v2	58.91 ± 0.85 %
RedditClusteringP2P.v2	59.72 ± 0.37 %
SIB200ClusteringS2S	54.67 ± 5.04 %
StackExchangeClustering.v2	65.67 ± 0.75 %
StackExchangeClusteringP2P.v2	43.73 ± 0.43 %
TwentyNewsgroupsClustering.v2	53.03 ± 1.50 %
WikiCitiesClustering	80.68 ± 0.00 %
WikipediaChemistryTopicsClustering	67.47 ± 0.00 %
WikipediaSpecialtiesInChemistryCl.	28.08 ± 0.00 %
AVG	**54.49** %

Table 11: Clustering accuracy of `Llama_last` across multiple MTEB tasks. Results show the `v_measure` and `v_measure_std` reported by running MTEB using mteb.get_tasks(task_types="Clustering", languages=["eng"]).

change), and encyclopedic text (Wikipedia, WikiCities). It further distinguishes between task granularities, such as Sentence-to-Sentence (S2S) tasks, which use short text segments like titles or individual sentences, and Passage-to-Passage (P2P) tasks, which rely on longer text spans such as abstracts or full passages (Muennighoff et al., 2023). This coverage ensures that the results reflect robustness across heterogeneous text types. The macro-average of 54.49 % indicates stable overall performance, with the strongest results on well-structured corpora such as ClusTREC-Covid and WikiCities. When comparing tasks available in both configurations, P2P generally outperforms S2S, as observed for BioRxiv, BuiltBench, MasakhaNEWS, MedRxiv, and Reddit. However, this trend is not universal: ArXiv shows nearly identical performance for both settings, while StackExchange exhibits a strong advantage for S2S over P2P. These results indicate that the optimal task granularity is dataset-dependent, with P2P often providing richer contextual signals but S2S remaining competitive or superior when concise text segments are more informative. Different tasks reach their best performance after varying numbers of training iterations. Because evaluation was conducted jointly across all 21 tasks, checkpoints were selected based on overall performance rather than task-specific peaks. As a result, the chosen checkpoints may not be optimal for every individual dataset. All results were obtained with the official MTEB evaluation pipeline (Muennighoff et al., 2023), ensuring direct comparability with prior work.

B.3 Analysis of Attention

In this section, we have randomly chosen several additional example sentences for a visual attention analysis. Each plot shows the average attention received by every input token from the final token in the last transformer layer of Llama-3.2-1B. For the fine-tuned condition, we use the best-performing model. Bars indicate attention at three stages of training: prompt-only, after 100 fine-tuning steps, and after 20,000 steps. Across all examples, the pattern is consistent: short fine-tuning shifts attention from prompt tokens toward the actual sentence being encoded, improving over the prompt-only baseline, whereas prolonged fine-tuning results in a more uniform distribution. The sentences were chosen to be short for readability. In addition to examples aligned with Zhang et al., we include samples from the EmotionClassification dataset (Muennighoff et al., 2023) as well as other simple sentences.

Figures 3 and 4 show how fine-tuning changes attention patterns across four sentences. For the EmotionClassification examples, attention shifts from prompt tokens toward emotionally salient words such as *embarrassed*, *worthless*, and *struggling*. In the shorter generic sentences, the model similarly learns to emphasize core content words like *tired*, *running*, *read*, and *story*. Extended training eventually redistributes attention more evenly while still retaining stronger focus on semantically central tokens.

Overall, the zero-shot setting already reveals that semantically relevant tokens tend to attract higher attention, indicating that the model encodes some notion of relevance without task-specific adaptation. Fine-tuning maintains this advantage while shifting focus away from prompt tokens toward the sentence to be embedded. The combination of prompt usage, which guides the model to store information in the final token, and fine-tuning, which encourages greater focus on semantic content, indicates that the model learns to produce more discriminative embeddings.

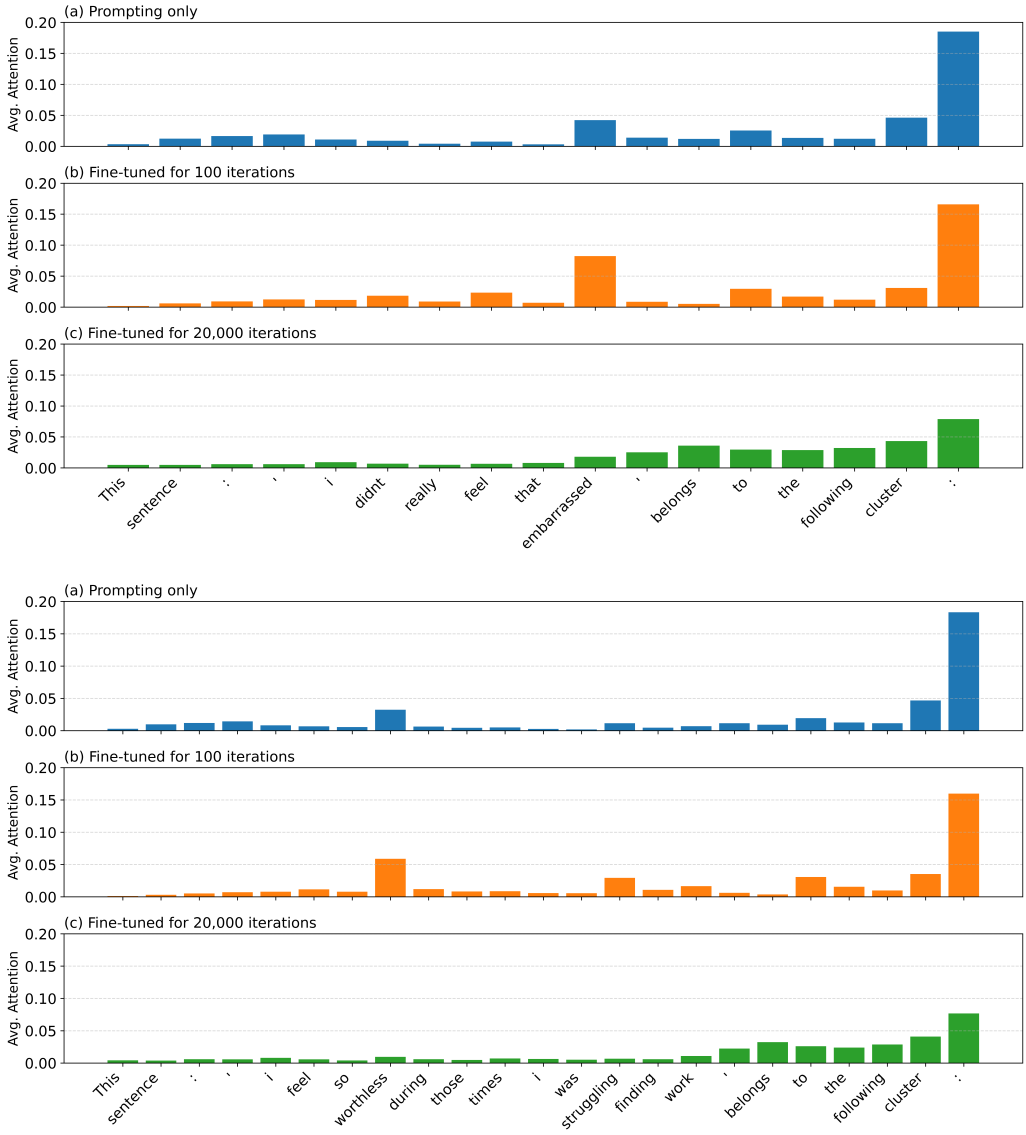

Figure 3: Average attention received by each input token from the final token in the last layer of Llama-3.2-1B with the CCW prompt. Shown for two EmotionClassification sentences (Muennighoff et al., 2023). Each plot reports three stages: prompt-only, after 100 fine-tuning steps, and after 20,000 steps.

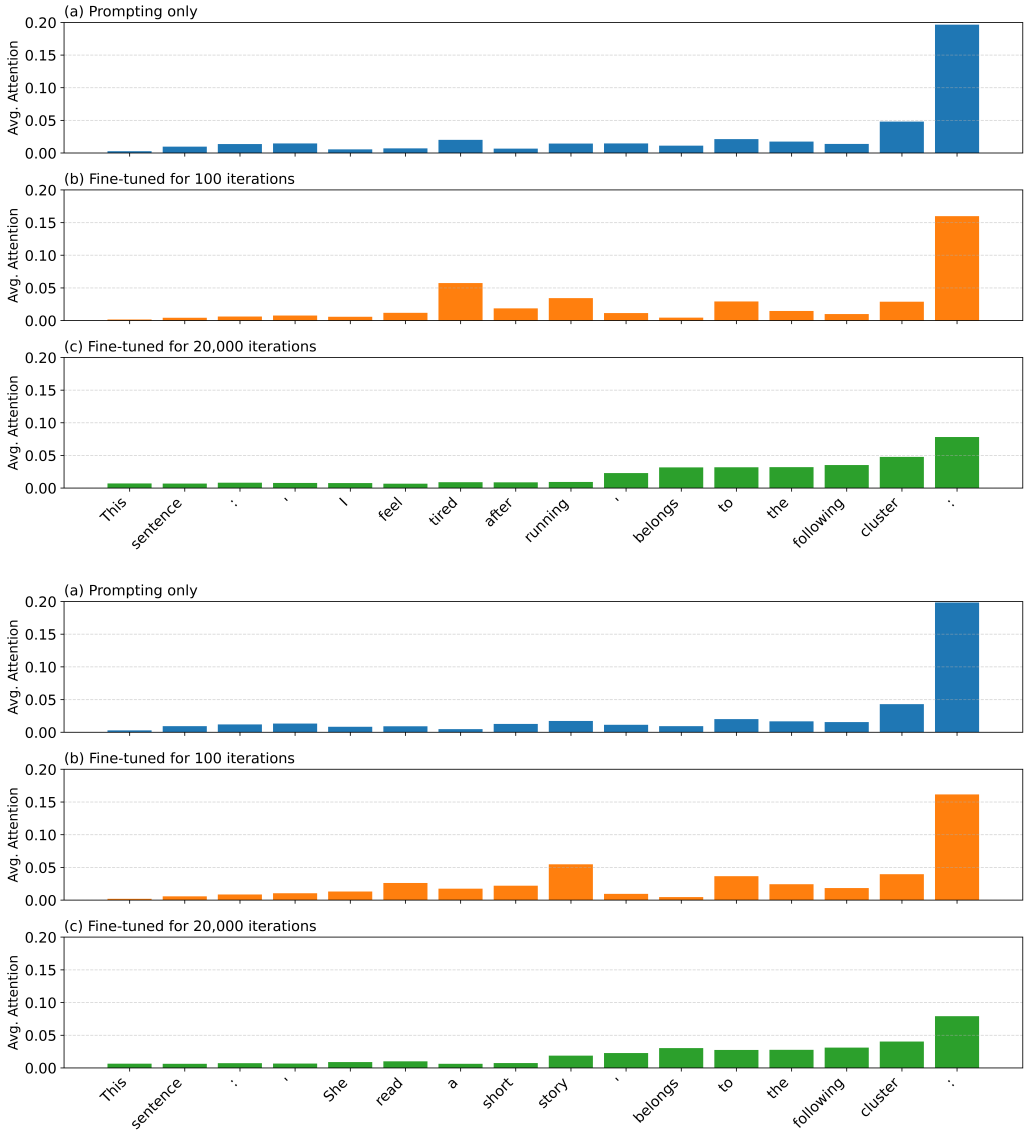

Figure 4: Average attention received by each input token from the final token in the last layer of Llama-3.2-1B with the CCW prompt. Shown for two random short sentences: "I feel tired after running" and "She read a short story." Each plot reports three stages: prompt-only, after 100 fine-tuning steps, and after 20,000 steps.