TriEmbed: Bridge the Gap between Text and Token Indices with Embedding Reparameterization

Anonymous ACL submission

Abstract

The current paradigm of language modeling is a two-stage pipeline that first transforms raw text to token indices, where the distribution is then estimated. It inherently discards linguistic relations between tokens during tokenization, creating a fundamental gap. To address this, we propose **TriEmbed**, a reparameterization method for embeddings that incorporates the morphological relationships inherent in subword tokenizer algorithms. Specifically, by organizing the vocabulary into a Trie structure, we can encode these relations and reparametrize the embeddings, facilitating the recovery of other linguistic relationships during training. Empirical results across various settings demonstrate that TriEmbed outperforms conventional embeddings from the perspective of scaling, while offering more linguistically informative token embeddings.

1 Introduction

In recent years, language modeling has undergone rapid advancements, unlocking unprecedented potential through large-scale scaling. The current paradigm of language modeling is a pipeline consisting of two decoupled components: a tokenizer and a language model. The tokenizer converts raw text, defined in the *text space*, into a sequence of token indices in the *token index space*, where the language model can operate in a differentiable manner. However, this transformation also introduces a fundamental gap: tokenization discards the linguistic relations between tokens (e.g., morphological variations such as "run" and "running" or synonymous expressions like "big" and "large"), reducing them to mere numerical representations (i.e. token indices) with only identity relations preserved.

Common practice in LLMs disregards the gap, relying on end-to-end training with the next token prediction objective to implicitly recover these lost relationships. However, this process is uncontrollable, possibly leading to problems including representation degeneration (Gao et al., 2019), frequency bias (Yu et al., 2022) and so on. Some previous work attempts to include semantic relations by pretraining embedding modules (Minixhofer et al., 2022; Dobler and de Melo, 2023). Yet, structural discrepancies between different representation spaces commonly induce domain shifts in continual training, resulting in worse performance than random initializations (Kim et al., 2024).

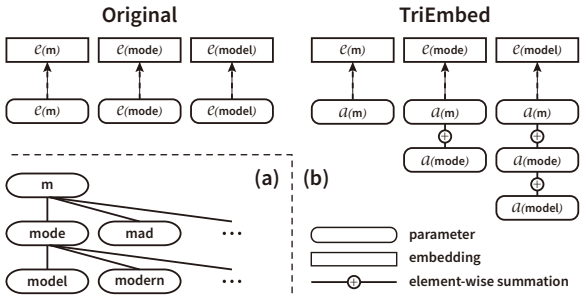

Figure 1: (a) A subtree of token **"m"** in the Trie-structured vocabulary. (b) Comparison between original embeddings and our proposed TriEmbed. Each token is assigned one embedding vector $e(\cdot)$ for the former and one atomic vector $a(\cdot)$ for the latter as parameters.

To bridge the gap, we propose a novel reparameterization method for embedding modules, **TriEmbed**, which incorporates the morphological relations inherent in tokenizer algorithms. Subword tokenizer builds on word formation morphology. And its training process reflects morphological relations within vocabulary. We encode these relations in ancestor-descendant connections within an implicit Trie. Following the hierarchical structure of the Trie, we derive each embedding from all its ancestors. Our method retains the benefits of random initialization while seamlessly integrating inductive bias about the morphological relations among token indices.

We conduct extensive experiments to validate the effectiveness of TriEmbed across different scales of training corpora and model sizes. Notably, when

pretraining Pythia on FineWeb-Edu, TriEmbed achieves performance on par with conventional embedding modules while requiring about 70% of the model parameters or 90% of the pretraining corpus. Furthermore, a detailed analysis reveals that our approach significantly improves the morphology of representation space, offering more meaningful token embeddings.

2 Background

Language modeling aims to estimate the probability $p(s)$ of any given text string $s \in \Sigma^*$. Current paradigm follows a two-component pipeline consisting of a *tokenizer* and a *language model*. A tokenizer segments a text string s into *tokens*, then maps them to a sequence of *token indices* $X = [x_0, ...x_N] \in \mathbb{Z}^*$. It provides an intermediate *token index space*, in which the language model can estimate the probability distribution $p_\theta(x_i|x_{<i})$ in a differentiable way.

Tokenizer algorithm focuses on constructing a suitable vocabulary. LLMs predominantly adopt subword tokenizer such as BPE (Sennrich et al., 2016), which hypothesizes that complex words should be split into multiple subwords. For training, they begin with a base vocabulary (e.g. all Unicode characters), and iteratively merges the most frequent token pairs in training corpus as a new token to expand the vocabulary (see Appendix A for the pseudo-code). Therefore, the resulting vocabulary implicitly encodes a Trie[1], as shown in Figure 1(a).

Although the pipeline simplifies the training process for next token prediction, it also introduces a fundamental gap when transforming from the original text space to the token index space.

To illustrate this issue, consider the word spelling challenge. A sample $s =$ "$model\rightarrow m\,o\,d\,e\,l$" is tokenized as $X = [\mathbf{id}(model), \mathbf{id}(\rightarrow), \mathbf{id}(\_m), \mathbf{id}(\_o), \mathbf{id}(\_d), \mathbf{id}(\_e), \mathbf{id}(\_l)]$. The morphological relations between different tokens (e.g. "$model$" and "$\_m$") are lost, leading to frequent failure of LLMs in word spelling tasks (Karpathy, 2024).

This issue extends beyond morphology and is inherent to the two-component pipeline. A wide range of linguistic relationships, including semantic, etymological, and derivational connections, are similarly discarded after tokenization, since language models can only observe token indices with identity relations left.

Common practice in LLMs tends to disregard the gap, relying on end-to-end training with the next token prediction objective to learn these linguistic information implicitly. However, this process is computational expensive. Moreover, the stochastic nature of gradient-based algorithm makes the training process uncontrollable, leading to severe problems such as representation degeneration (Gao et al., 2019; Biś et al., 2021) and frequency bias (Schick and Schütze, 2020; Gong et al., 2018; Yu et al., 2022). Another line of work (Minixhofer et al., 2022; Dobler and de Melo, 2023) tries to initialize embeddings with pretrained weights to partially recover the missing relations among tokens in vocabulary. However, these pretrained embeddings often exhibit structural discrepancy with the final representation space of the language model. For instance, they tend to have significantly larger variance magnitudes (Kim et al., 2024), which makes the subsequent training difficult. More related work is presented in Appendix B.

3 Method

As discussed in the previous section, the transformation from text space to token index space discards linguistic relationships between tokens, introducing a fundamental gap. To address this, we aim to leverage the inductive bias inherent in the tokenizer to recover the lost information during tokenization.

The subword tokenizer draws inspiration from Word Formation Morphology (Pounder, 2000; wfm, 1986; Görlach, 2003), a field that studies the structure of words in terms of morphemes. And its iterative merging process mirrors the word formation process where words are constructed by sequentially adding affixes to stems. As such, an inherent morphological relation exists among the resulting vocabulary.

By reorganizing the vocabulary into a Trie structure, this morphological information is captured within the ancestor-descendant relationships of the tree. We hypothesize that, the ancestor-descendant relationship can serve as a foundation for the intricate linguistic relations among tokens, facilitating the restoration of other relations during training.

Building on the hypothesis, we propose **TriEmbed**, a novel reparameterization approach for the embedding module, which incorporates the

[1]We additionally consider an non-existing start-of-word token as the prefix of all tokens in the vocabulary, hence constructing a Trie.

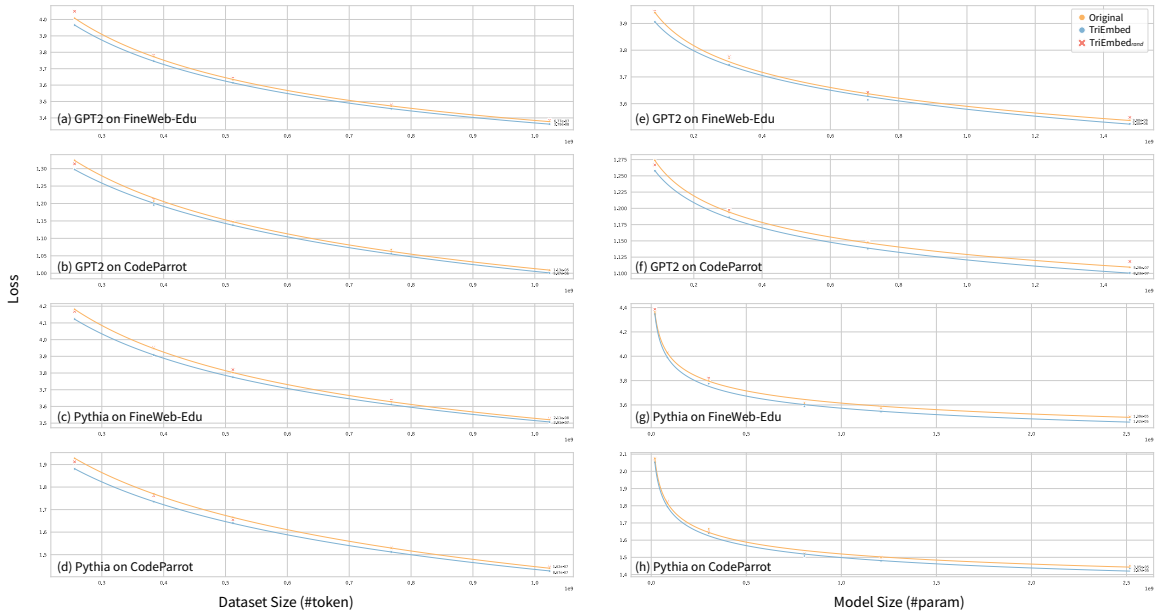

Figure 2: Scalability of different embeddings. (a-d) Pythia-410m and GPT2-large trained on different size of dataset. (e-h) Different scales of Pythia and GPT2 trained on 0.5B tokens. The blue and orange curves represent the scaling laws of TriEmbed and original embedding correspondingly, with fitting error shown on the right of each curve.

inductive bias about the Trie structure among tokens. Given a token, we derive its embedding from those of its ancestors. Specifically, we first assign each token with a unique atomic vector $a(\cdot)$, then define the embedding of a token as the cumulative sum of the atomic vectors of all its ancestors and its own, i.e. $e(v) = \sum_{u \in \text{ancestors(v) or } u=v} a(u)$. This summation process follows a traversal from the root of the Trie to the current node, reflecting the word formation process in which affixes are progressively added to a stem. The method is illustrated in Figure 1(b).

A key advantage of our approach is that it does not introduce additional parameters, maintaining parameter efficiency. Additional computational costs are also minimal due to simplicity of summation. Despite its simplicity, TriEmbed still brings significant improvements, which we will empirically demonstrate in the following experiments.

4 Experiments

4.1 Settings

Our experiments are conducted on two widely used pretraining datasets including FineWeb-Edu (Penedo et al., 2024) and CodeParrot (Tunstall et al., 2022). We consider three different series of language model architectures including Pythia (Biderman et al., 2023), GPT2 (Radford et al., 2019) and Qwen2.5 (Qwen et al., 2025). Notably, all models are trained from scratch, and any references to model names refer to architectures rather than pretrained weights. Unless stated, we adopt Pythia-410m and GPT2-large as the default model configurations and use a 0.5B-token subset of FineWeb-Edu as the default dataset. More details about experimental settings are presented in Appendix C.

4.2 Scalability of TriEmbed

We evaluate the scalability of TriEmbed across varying model and dataset sizes. We fit both model scaling laws and data scaling laws on the experimental results using Huber loss minimization, following Hoffmann et al. (2022). The scaling law fits well in our experimental results with fitting error less than 0.0001. The results for Pythia and GPT2 are presented in Figure 2, while those for Qwen2.5 are included in Appendix D.

It is evident that the fitted scaling curves of TriEmbed are consistently below those of conventional embedding modules. Notably, a Pythia model with TriEmbed achieves performance comparable to a conventional 2.8B-parameter Pythia model while requiring only 70% parameters. Furthermore, under an identical Pythia-410M configuration, TriEmbed requires only 90% of the text corpus to match the performance of a conventional one trained on 1B tokens.

Furthermore, we also evaluate the transfer learning capability of TriEmbed by finetuning it on SAMSum, a dialogue summarization dataset. The

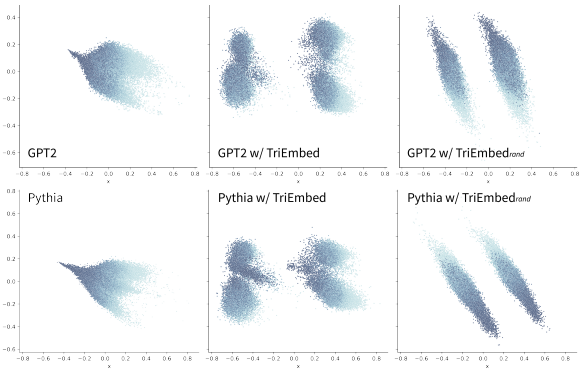

Figure 3: Visualization of different token embeddings through a 2-dimensional PCA projection. The color gradient corresponds to token frequency in training corpus, with darker shades indicating lower frequencies.

results shown in Table 1 further highlight the superiority of TriEmbed.

4.3 Ablation of Inductive Bias

The proposed reparameterization not only incorporates morphological relationships within the vocabulary but also alters some gradient dynamics. To investigate the reasons for the improved scaling performance of TriEmbed, we conduct experiments with other factors ablated. Specifically, we preserve the topological structure of the Trie and randomly shuffle all token indices, thereby disrupting the ground-truth ancestor-descendant relationships. The variant denoted as **TriEmbed**$_{rand}$ now shares a similar gradient dynamics with TriEmbed, but introduces incorrect inductive bias. As shown in Figure 2, results of TriEmbed$_{rand}$ closely aligns with the original embedding module under all settings. This observation strongly suggests that the performance gains stem primarily from the morphological relations between tokens in the tokenizer, further demonstrating our hypothesis.

4.4 Analysis of Embedding

Visualization. We visualize the resulting token embeddings of different embedding modules in Figure 3. Consistent with the representation degeneration problem in Gao et al. (2019), both original embeddings and TriEmbed$_{rand}$ degenerate into narrow cones in the space. Moreover with color indicating token frequency, we find that rare tokens and popular tokens are heavily clustered and lie in different subregions of the space, aligned with the observation of frequency bias in Gong et al. (2018). In contrast, TriEmbed results in a significantly more uniform distribution over the embedding space, with high-frequency and low-frequency tokens evenly mixed together. This suggests that our approach mitigates both representation degeneration and frequency bias, encouraging a better embedding space structure.

Model	BERTScore	Rouge1	Rouge2	RougeL
Pythia-410m	**84.40**	17.27	6.97	**15.70**
Pythia-410m w/ TriEmbed	84.32	**18.08**	**7.31**	15.57
GPT2-large	84.33	31.62	13.65	26.64
GPT2-large w/ TriEmbed	**87.87**	**38.21**	**16.74**	**31.69**

Table 1: Fine-tuning performance of TriEmbed on SAMSum. All models are pretrained on FineWeb-Edu (0.5B tokens subset) first.

Model	Rare Words		Word Analogy	
Embedding	Original	TriEmbed	Original	TriEmbed
GPT2	52.41	**54.93**	30.55	**47.45**
GPT2-medium	55.50	**56.11**	24.75	**43.77**
GPT2-large	**56.88**	53.66	20.65	**50.14**
GPT2-xl	**55.61**	50.90	17.02	**63.51**
Pythia-70m	46.00	**51.62**	36.84	**65.73**
Pythia-160m	50.86	**57.49**	28.95	**49.88**
Pythia-410m	52.07	**53.34**	23.69	**53.78**
Pythia-1b	54.99	**55.64**	13.18	**37.83**
Pythia-1.4b	52.00	**56.14**	13.21	**36.09**
Pythia-2.8b	48.36	**50.93**	10.59	**52.69**

Table 2: Embedding performance on benchmarks for word similarity and analogy. All models are pretrained on FineWeb-Edu (0.5B tokens subset).

Word embedding benchmarks. We also validate the quality of learned token embeddings on standard word embedding benchmarks, including Stanford Rare Words (Luong et al., 2013) and Word Analogy (Mikolov et al., 2013). Results presented in Table 2 highlight the superiority of TriEmbed, demonstrating our hypothesis that ancestor-descendant relations can serve as a foundation for learning other linguistic relations more effectively.

5 Conclusions

In this work, we review the gap between text space and token index space inherent in the current two-component language modeling pipeline. We introduce TriEmbed, a reparameterization method for the embedding module that incorporates morphological information from the tokenizer algorithm. Extensive experiments demonstrate the effectiveness of TriEmbed particularly in terms of scalability, suggesting a promising direction for future large-scale pretraining of language models.

Limitations

The gap between token index space and text space is inherent to the current two-component pipeline paradigm. While TriEmbed partially mitigates this issue, it cannot fully close the gap without abandoning the tokenization process.

Additionally, our experiments are limited to pretraining on small-scale models with fewer than 3 billion parameters due to resource limits. While results of our experiments have already demonstrated clear advantages of TriEmbed, scaling up the experiments to larger models and training corpus would provide more robust validation, which we leave as future work.

References

1986. *Chapter III: Word formation in generative morphology*, pages 37–56. De Gruyter Mouton, Berlin, Boston.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. *Preprint*, arXiv:2304.01373.

Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. 2021. Too Much in Common: Shifting of Embeddings in Transformer Language Models and its Implications. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5117–5130, Online. Association for Computational Linguistics.

Konstantin Dobler and Gerard de Melo. 2023. FOCUS: Effective Embedding Initialization for Monolingual Specialization of Multilingual Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13440–13454, Singapore. Association for Computational Linguistics.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2019. Representation Degeneration Problem in Training Natural Language Generation Models. *Preprint*, arXiv:1907.12009.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2018. FRAGE: Frequency-Agnostic Word Representation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Manfred Görlach. 2003. *7. Morphology and word formation*, pages 75–92. John Benjamins Publishing Company.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. *Preprint*, arXiv:2203.15556.

Andrej Karpathy. 2024. Let's build the gpt tokenizer.

Ha Young Kim, Niranjan Balasubramanian, and Byungkon Kang. 2024. On Initializing Transformers with Pre-trained Embeddings. *Preprint*, arXiv:2407.12514.

Thang Luong, Richard Socher, and Christopher Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Preprint*, arXiv:1301.3781.

Benjamin Minixhofer, Fabian Paischer, and Navid Rekabsaz. 2022. WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3992–4006, Seattle, United States. Association for Computational Linguistics.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-Top: Simple and Effective Postprocessing for Word Representations. In *International Conference on Learning Representations*.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Preprint*, arXiv:2406.17557.

Amanda Pounder. 2000. *Process and Paradigms in Word-Formation Morphology*. De Gruyter Mouton, Berlin, New York.

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, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Timo Schick and Hinrich Schütze. 2020. Rare Words: A Major Problem for Contextualized Embeddings and How to Fix it by Attentive Mimicking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8766–8774.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

L. Tunstall, L. von Werra, and T. Wolf. 2022. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly Media.

Dilin Wang, Chengyue Gong, and Qiang Liu. 2019a. Improving Neural Language Modeling via Adversarial Training. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6555–6565. PMLR.

Lingxiao Wang, Jing Huang, Kevin Huang, Ziniu Hu, Guangtao Wang, and Quanquan Gu. 2019b. Improving Neural Language Generation with Spectrum Control. In *International Conference on Learning Representations*.

Sangwon Yu, Jongyoon Song, Heeseung Kim, Seongmin Lee, Woo-Jong Ryu, and Sungroh Yoon. 2022. Rare Tokens Degenerate All Tokens: Improving Neural Text Generation via Adaptive Gradient Gating for Rare Token Embeddings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 29–45, Dublin, Ireland. Association for Computational Linguistics.

A BPE Algorithm

Algorithm 1 Byte-pair Encoding

Input: Training corpus D, Target vocab size k, Base vocabulary V
Output: Final vocabulary V
while $|V|<k$ **do**
  $(t_l, t_r) \leftarrow$ most frequent bigram in D.
  $t_{new} \leftarrow \text{concat}(t_l, t_r)$
  $V \leftarrow V + \{t_{new}\}$
  Replace all occurrence of (t_l, t_r) with t_new in D
end
return V

B Extended Related Work

Gao et al. (2019) observe that language modeling trained with next-token-prediction loss using gradient descent commonly leads to token embeddings degenerating to a narrow cone. This phenomenon, named the representation degeneration problem, indicates an overall similarity among embeddings, leading to decreased expressiveness of token embeddings. Therefore, it is difficult for the model to learn linguistic relationships between the tokens and to generate high quality texts. Existing studies addressing this problem by applying post-processing (Mu and Viswanath, 2018; Biś et al., 2021) or regularization (Gao et al., 2019; Wang et al., 2019b,a) directly to constraint the embedding space. Our proposed TriEmbed doesn't explicitly constraint the embedding space. The reparameterization automatically change the gradient dynamics, mitigating this problem.

Frequency bias in embedding space is another problem. Gong et al. (2018) finds that rare words and frequent words commonly occupy different sub-regions of the embedding space. The reason stems from the gradient descent with softmax. Because of the low sampling rates of rare words, their token embeddings are merely updates, leading to under-estimation. According to their observation, the moving distance of the embedding for a popular word is twice longer than that of a rare word during training. Yu et al. (2022) mitigate this problem by gating a specific part of the gradient of rare words. Similarly, TriEmbed also mitigates the problem via altering gradient dynamics. Since the token embeddings are now dependent, rare token embeddings are more actively updated by the gradients of their descendants.

C Experimental Details

Our experiments are conduct on two different datasets, including FineWeb-Edu (Penedo et al., 2024) and CodeParrot (Tunstall et al., 2022). FineWeb-Edu is a popular English pretraining dataset filtered by an educational quality classifier for our main experiments. CodeParrot is also a commonly used code pretraining dataset consist of 5,361,373 Python files crawled from Github. For both datasets, we chunk the corpus into samples of 512 sequence length. In the data scaling experiments, we randomly sampled five subsets with 1B, 0.75B, 0.5B, 0.375B and 0.25B tokens respectively.

We consider three different series of language

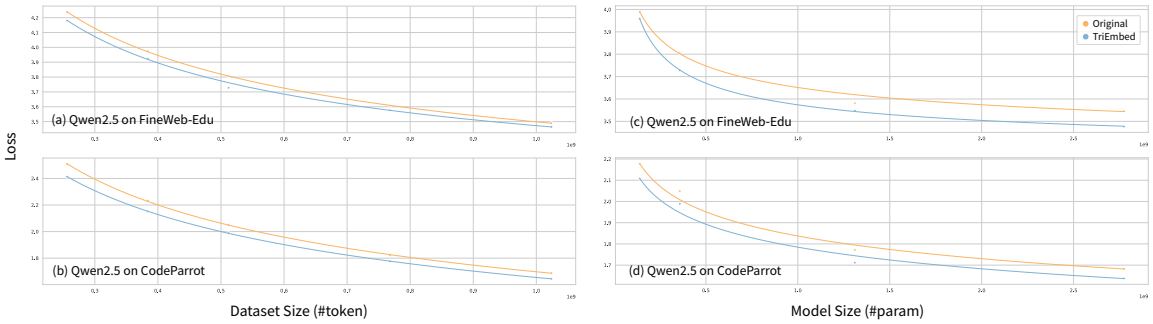

Figure 4: Scalability of different embeddings. (a)(b) Qwen-2.5 trained on different size of dataset. (c)(d) Different scales of Qwen-2.5 trained on 0.5B tokens. The blue and orange curves represent the scaling laws of TriEmbed and original embedding correspondingly.

model architectures including Pythia (Biderman et al., 2023), GPT2 (Radford et al., 2019) and Qwen2.5 (Qwen et al., 2025). All models are weight tied by default.

For all experiments, we train the model from scratch. The training batch size is 128 and the training epoch is 1. We set the learning rate to 1e-4. All loss reported in this work is calculated on a test dataset of 5,000,000 tokens.

D Scalibility of TriEmbed on Qwen2.5

The scaling experimental results of Qwen2.5 series are shown in Figure 4, which present similar patterns to those of GPT2 and Pythia. Specifically, we add a Qwen-0.25B configuration by scaling down Qwen-0.5B.