Adapting Language Models via Token Translation

Zhili Feng
Carnegie Mellon University

Tanya Marwah
Carnegie Mellon University

Lester Mackey
Microsoft Research

David Alvarez-Melis
Microsoft Research

Nicolò Fusi
Microsoft Research

Abstract

Modern large language models use a fixed tokenizer to effectively compress text drawn from a source domain. However, applying the same tokenizer to a new target domain often leads to inferior compression, more costly inference, and reduced semantic alignment. To address this deficiency, we introduce Sparse Sinkhorn Token Translation (S2T2). S2T2 trains a tailored tokenizer for the target domain and learns to translate between target and source tokens, enabling more effective reuse of the pre-trained next-source-token predictor. In our experiments with finetuned English language models, S2T2 improves both the perplexity and the compression of out-of-domain protein sequences, outperforming direct finetuning with either the source or target tokenizer. In addition, we find that token translations learned for smaller, less expensive models can be directly transferred to larger, more powerful models to reap the benefits of S2T2 at lower cost.

1 Introduction

Modern large language models (LLMs) are typically trained in two stages. First a tokenizer is trained to map commonly occurring character sequences in the training data into vocabulary units known as *tokens*. Next, all training text is tokenized, i.e., translated into this token vocabulary, and a model is trained to predict the next token given a context of preceding tokens. The tokenizer can be viewed as an initial compressor of input bytes [Gage, 1994] that significantly shortens text drawn from the training domain and arguably improves the training dynamics [Rajaraman et al., 2024]. Despite its widespread adoption, this two-stage procedure suffers from a key failing: When faced with text from a new target domain, compression quality drops, context length and inference costs increase, and learned semantic alignment deteriorates. This effect is especially evident when modern LLMs (trained predominantly on English and code) are used to reason about molecular sequences like proteins. Such sequences are commonly represented using the Latin-script alphabet, but the meaning and frequency of each substring differ significantly their natural language counterparts, resulting in semantic misalignment.

To tackle the analogous alignment problem for low-resource languages, Remy et al. [2024] proposed to use `fast_align` [Dyer et al., 2013], an expectation-maximization algorithm that requires parallel data from the training and target domains.

This approach shows promising results, but for many target domains, parallel training data is difficult or impossible to gather. For example, there is no agreed-upon parallel translation between protein sequences and natural language.

In this work, we propose a ***S**parse **S**inkhorn **T**oken **T**ranslation (S2T2)* algorithm that does not require parallel data. Instead, S2T2 learns a translation between training domain tokens and new target domain tokens just using a sample data from the target domain and the pretrained LLM weights. After training a tokenizer on the target domain, S2T2 translates each target-domain token into a (sparse)

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

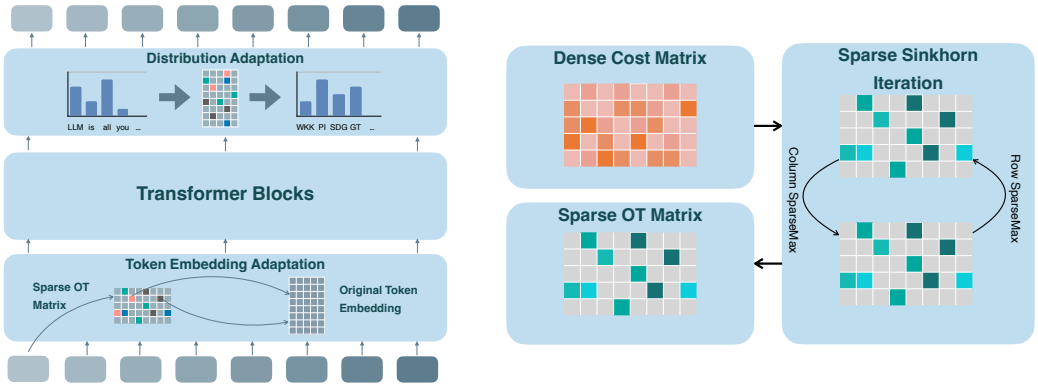

Figure 1: Overview of S2T2. **Left**: S2T2 injects a weight-tied sparse optimal transport (OT) layer in both the token embedding and language model head. The input tokens will be encoded based on a sparse convex combination of the original token embeddings and decoded by a sparse combination of the original language model head. **Right**: The sparse OT matrix is obtained by iteratively projecting a dense cost matrix along its rows and columns. The dense cost matrix is updated by backpropogation.

distribution over training-domain tokens, uses the pretrained LLM to predict the next training-domain token, and translates that training-domain token back into a (sparse) distribution over target-domain tokens. In our experiments with English LLMs, we find that

1. S2T2 provides an effective initialization for continual finetuning on protein sequences, yielding both better compression and better perplexity than direct finetuning of the pretrained model, and
2. S2T2 enables *weak-to-strong model transferability*: Translations learned for smaller, less expensive models can be transferred to larger, more powerful models to reap the benefits at lower cost.

2 Translating Tokens with Sparse Sinkhorn

Consider a pretrained LLM $\mathcal{M}$ with vocabulary size v, embedding matrix $\mathbf{E} \in \mathbb{R}^{v \times d}$, and language model head $\mathbf{L} \in \mathbb{R}^{v \times d}$. For a given input sequence encoded as a matrix $\mathbf{X} \in \{0,1\}^{s \times v}$ in which each row is a one-hot vector representing a training-domain token, $\mathbf{XE} \in \mathbb{R}^{s \times d}$ represents the sequence of (soft) embeddings, and the predicted next token is given by

$$\mathcal{M}(\mathbf{X}) = \arg\max_{i \in [v]} \operatorname{softmax}(\mathbf{L}h(\mathbf{XE}))_i \in \{0,1\}^v \tag{1}$$

where $h : \mathbb{R}^{s \times d} \to \mathbb{R}^d$ maps an embedding sequence into a single vector, the internal representation of the next token.

Consider also a dataset D drawn from a new target domain, and let u be the vocabulary size of a new tokenizer trained on D. For given marginal distributions over training and target tokens $\boldsymbol{\mu} \in \Delta_{v-1}$ and $\boldsymbol{\nu} \in \Delta_{u-1}$, we define the constraint set $C(\boldsymbol{\mu}, \boldsymbol{\nu}) = \{\mathbf{P} \in [0,1]^{v \times u} : \mathbf{P}\mathbf{1} = \boldsymbol{\mu}, \mathbf{P}^\top \mathbf{1} = \boldsymbol{\nu}\}$. S2T2 finds a joint probability matrix $\mathbf{P} \in C(\boldsymbol{\mu}, \boldsymbol{\nu})$ and defines a new target-domain LLM $\mathcal{M}'$ with embedding matrix $\mathbf{E}' = \left(\mathbf{P}^\top \odot (1/\boldsymbol{\mu})\right)\mathbf{E} \in \mathbb{R}^{u \times d}$ and language head $\mathbf{L}' = \left(\mathbf{P} \odot (1/\boldsymbol{\nu})\right)^\top \mathbf{L} \in \mathbb{R}^{u \times d}$ substituted for $(\mathbf{E}, \mathbf{L})$ in (1). Here, $\mathbf{A} \odot \boldsymbol{v}$ represents a Hadamard product broadcasted along the last dimension. It is crucial to perform such a Hadamard product, since we want the new token embedding and old token embedding to be on the same scale. More generally, one could use different $\mathbf{P}$ matrices to translate $\mathbf{E}$ and $\mathbf{L}$, but we focus on a single $\mathbf{P}$ here for simplicity. An overview of S2T2 can be in Fig. 1.

2.1 Finding P via Sparse Sinkhorn

Since it is difficult to directly parameterize a joint probability matrix $\mathbf{P} \in C(\boldsymbol{\mu}, \boldsymbol{\nu})$, we instead maintain a dense weight matrix $\mathbf{C} \in \mathbb{R}^{v \times u}$ and recover $\mathbf{P}$ as the solution to the following two equivalent optimization problems.

$$\min_{\mathbf{P}'} \quad \frac{1}{2}\|\mathbf{P}' - \mathbf{C}\|_F^2 \qquad (2) \qquad \min_{\mathbf{P}'} \quad \langle -\mathbf{C}, \mathbf{P}' \rangle + \frac{1}{2}\|\mathbf{P}'\|_F^2 \qquad (3)$$
$$\text{s.t.} \quad \mathbf{P}' \in C(\boldsymbol{\mu}, \boldsymbol{\nu}) \qquad\qquad \text{s.t.} \quad \mathbf{P}' \in C(\boldsymbol{\mu}, \boldsymbol{\nu})$$

Notice that (3) is the ℓ_2 constrained optimal transport problem, which is known to generate sparse solutions [Essid and Solomon, 2018, Peyré et al., 2019]. Moreover, since $C = C_1 \cap C_2$ for the convex sets $C_1 = \{\mathbf{P} \in \mathbb{R}_+^{v \times u}, \mathbf{P}\mathbf{1} = \boldsymbol{\mu}\}$ and $C_2 = \{\mathbf{P} \in \mathbb{R}_+^{v \times u}, \mathbf{P}^\top \mathbf{1} = \boldsymbol{\nu}\}$, these problems can be solved using iterative Dykstra's projections [Boyle and Dykstra, 1986], a Sinkhorn-like algorithm via with guaranteed convergence (see Algorithm 1).

In every Sinkhorn iteration, we solve a set of ℓ_2 projections onto a probability simplex. This optimization problem enjoys an efficient backpropogation computation [Martins and Astudillo, 2016]. A small caveat is that we are not always projecting onto the unit simplex but rather onto a scaled simplex, so the optimization is modified accordingly in Algorithm 2.

Algorithm 1 SPARSE SINKHORN ITERATION

Require: Weight matrix $\mathbf{C} \in \mathbb{R}^{v \times u}$
$\mathbf{P} \leftarrow \mathbf{0}^{v \times u}$, $\mathbf{Q} \leftarrow \mathbf{0}^{v \times u}$, $\mathbf{X}_0 \leftarrow \mathbf{C}$
for $k = 0, \ldots, n$ **do**
  $\mathbf{Y}_k \leftarrow \mathcal{P}_{C_1}(\mathbf{X}_k + \mathbf{P}_k)$, where $\mathcal{P}_{C_1}$ applies SPARSEMAX with scale $\boldsymbol{\mu}_i$ to each row i.
  $\mathbf{P}_{k+1} \leftarrow \mathbf{X}_k + \mathbf{P}_k - \mathbf{Y}_k$
  $\mathbf{X}_{k+1} \leftarrow \mathcal{P}_{C_2}(\mathbf{Y}_k + \mathbf{Q}_k)$, where $\mathcal{P}_{C_2}$ applies SPARSEMAX with scale $\boldsymbol{\nu}_j$ to each column j.
  $\mathbf{Q}_{k+1} \leftarrow \mathbf{Y}_k + \mathbf{Q}_k - \mathbf{X}_{k+1}$
end for
return $\mathbf{X}_{n+1}$

Algorithm 2 SPARSEMAX

Require: $\boldsymbol{z} \in \mathbb{R}^K$, scale α
Sort $\boldsymbol{z}$ as $\boldsymbol{z}_{(1)} \geq \cdots \geq \boldsymbol{z}_{(K)}$
Find $k(\boldsymbol{z}) = \max\left\{k \in [K] : \alpha + k\boldsymbol{z}_{(k)} > \sum_{j \leq k} \boldsymbol{z}_{(j)}\right\}$
Let $\tau(\boldsymbol{z}) = \frac{\sum_{j \leq k} \boldsymbol{z}_{(j)} - \alpha}{k(\boldsymbol{z})}$
return $\boldsymbol{p}$ where $\boldsymbol{p}_i = \max\{\boldsymbol{z}_i - \tau(\boldsymbol{z}), 0\}$

To learn our token translation, we initialize the weight matrix $\mathbf{C}$ by setting each entry to be $1/v$, obtain the joint probability matrix $\mathbf{P}$ by applying Algorithm 1 to $\mathbf{C}$, and perform a normal forward pass using $\mathbf{P}$. During the backward pass, we differentiate through the Sinkhorn iteration and update $\mathbf{C}$ directly. In practice, we find that iterating 3 times is enough to generate an effective sparse $\mathbf{P}$.

3 Experiment

We conduct experiments on the UniRef50 [Suzek et al., 2015] protein sequence dataset using the OLMo-1B English LLM [Groeneveld et al., 2024] with batch size 16 and context length of 512. The training domain tokens in our experiment are bytes (single characters), and the target domain tokenizer is a new Byte-Pair Encoding (BPE) tokenizer [Gage, 1994] trained on UniRef50 with vocabulary size 512. The new tokenizer reduces the length our protein sequences by a factor of $1.82\times$ on average. This will in turn have sizable impact on the standard measure of model compression, bits-per-byte (BpB) [see Biderman et al., 2024, for details on calculating BpB]. To control the sparsity level of $\mathbf{P}$, we add an entropy regularizer $\alpha H(\mathbf{P})$ to the next token prediction loss with larger α encouraging smaller entropy and hence sparser $\mathbf{P}$. Unless otherwise specified, $\alpha = 0$.

We compare with four baseline methods: 1. Training an *unconstrained* translator $\mathbf{P}$ followed by whole-model finetuning. 2. Training a dense probabilistic translator $\mathbf{P}$ (using SOFTMAX in place of SPARSEMAX) followed by whole-model finetuning. 3. Finetuning the model directly using the original OLMo tokenizer. 4. Finetuning the model with the new tokenizer, resizing the embedding matrix $\mathbf{E}$ and language model head $\mathbf{L}$ by truncation.

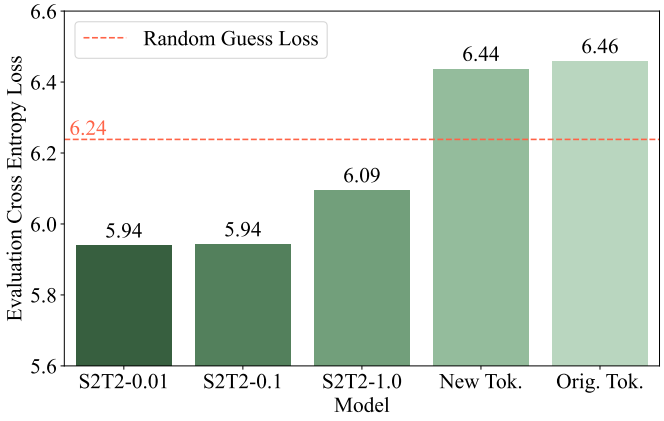

Figure 2: Evaluation loss after initializing OLMo-7B with token translator $\mathbf{P}$ learned from OLMo-1B. Along the x-axis, S2T2-α represent S2T2 with the α-entropy regularizer that controls the sparsity of $\mathbf{P}$. **New Tok.** is OLMo-7B with the new tokenizer and truncated $\mathbf{E}, \mathbf{L}$; **Orig Tok.** is OLMo-7B with the original tokenizer. The red dashed line is the loss when you **randomly guess** the next token.

Training details. We always train with AdamW [Loshchilov and Hutter, 2019]. When training $\mathbf{P}$, we use a learning rate of 10^{-3} (except for our model transfer experiments, which use 2×10^{-5}) and no weight decay; when finetuning the whole model, we always use learning rate of 2×10^{-5} with 0.01 weight decay. We follow the convention of training with BFloat16, $\beta_1 = 0.9, \beta_2 = 0.95$, and $\varepsilon = 10^{-5}$. We always use the cosine annealing scheduler with 20% linear warm-up steps and decay to 10% of the learning rate. We train $\mathbf{P}$ and finetune the whole model with 2000 steps.

Remarkably, Table 1 shows that simply initializing with S2T2 produces better language model quality (as measured by perplexity) and compression (as measured by BpB) than whole-model finetuning with the original tokenizer (baseline 3). Note that baseline 3 has much worse BpB due to its longer sequence length, further motivating the usage of a tailored tokenizer. In addition, S2T2 initialization outperforms both dense Sinkhorn and unconstrained token translation in both metrics. Moreover, after finetuning, S2T2 also improves upon the perplexity and BpB of baseline 4, direct finetuning with a new tokenizer. Fig. 2 shows that the translator $\mathbf{P}$ learned using OLMo-1B can also be directly transferred to the more expensive model, OLMo-7B, yielding significantly better performance than random guessing or OLMo-7B with its original tokenizer or the new tokenizer with truncated embedding matrix and language model head.

Table 1: Performance on UniRef50 evaluation set, measured by perplexity (**perp.**) and bits-per-byte (**BpB**). **Plain P**: Unconstrained $\mathbf{P}$. **CFT**: Continual finetuning, initialized from the learned $\mathbf{P}$. **FT orig. tok.**: Finetuning with the original tokenizer. **FT new tok.**: Finetuning with the new tokenizer.

	Plain P	**+ CFT**	**Sinkhorn P**	**+ CFT**	**S2T2**	**+ CFT**	**FT orig. tok.**	**FT new tok.**
Perp.	174.20	130.44	167.74	136.12	144.03	118.78	151.05	130.56
BpB	4.09	3.86	4.06	3.89	3.94	3.78	7.24	3.86

4 Conclusion

We proposed S2T2 as a token translation technique for continual finetuning of LLMs on out-of-distribution data and demonstrate its effectiveness on protein sequence modeling. As a next step, we plan to expand this framework to adapt to other modalities such as code and images. Another natural extension is to combine the training and target token vocabularies to produce an effective "multidomain" LLM.

5 Acknowledgement

This work was done during Zhili Feng's and Tanya Marwah's internship at Microsoft Research New England.

References

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.

James P Boyle and Richard L Dykstra. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in Order Restricted Statistical Inference: Proceedings of the Symposium on Order Restricted Statistical Inference held in Iowa City, Iowa, September 11–13, 1985*, pages 28–47. Springer, 1986.

Chris Dyer, Victor Chahuneau, and Noah A Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 644–648, 2013.

Montacer Essid and Justin Solomon. Quadratically regularized optimal transport on graphs. *SIAM Journal on Scientific Computing*, 40(4):A1961–A1986, 2018.

Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models. *Preprint*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. PMLR, 2016.

Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. Toward a theory of tokenization in llms. *arXiv preprint arXiv:2404.08335*, 2024.

François Remy, Pieter Delobelle, Hayastan Avetisyan, Alfiya Khabibullina, Miryam de Lhoneux, and Thomas Demeester. Trans-tokenization and cross-lingual vocabulary transfers: Language adaptation of LLMs for low-resource NLP. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=sBxvoDhvao.

Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.