HyperOFA: Expanding LLM Vocabulary to New Languages via Hypernetwork Based Embedding Initialization

Enes Özeren
Department of Statistics, LMU Munich
enes.oezeren@campus.lmu.de

Abstract

Most pre-trained language models are trained primarily for high-resource languages, limiting their usability in mid and low-resource languages. A common approach to adapt these models for other languages involves introducing new tokens specific to these languages and continuing pre-training. However, the method used to initialize these newly introduced tokens significantly impacts the duration and efficiency of continued pre-training. Poor initialization can lead to longer training times and increased computational costs. OFA [Liu et al., 2023a] method provides an effective initialization strategy. Building on this, HyperOFA introduces a hypernetwork-based approach to initialize new tokens. Experimental results show that HyperOFA consistently outperforms the random initialization baseline and performs on par with OFA, sometimes surpassing it and sometimes falling behind.

1 Introduction

Current language models (LMs) require pre-training on vast amounts of text data. As a result, most LMs are trained on high-resource languages, such as English and Chinese, which have a lot of available linguistic data. High-resource languages are those with extensive digital text corpora, while mid- and low-resource languages lack sufficient data for effective model training. Consequently, many LMs are either monolingual or multilingual, but for limited high-resource languages. This limits the usability and accessibility of LMs for speakers of mid or low-resource languages, hindering their potential benefits and widespread adoption. Training a language model from scratch for such languages is costly and often impractical due to limited text data and the large number of model parameters—hence the term Large Language Models (LLMs).

Most LMs consist of two primary components: transformer block parameters, which capture general language reasoning capabilities, and embedding parameters, which encode token representations for specific languages. Extending pre-trained LMs (PLMs) to support additional languages often requires modifying and initializing the embedding parameters for the new languages. Therefore, some research focuses on extending the language capabilities of PLMs by initializing new token embeddings from new languages. Some of these approaches aim to transfer the capabilities of a monolingual model to another language, resulting in a new monolingual model [de Vries and Nissim, 2020, Minixhofer et al., 2021], while others explore expanding existing monolingual or multilingual models to include a broader range of languages [Artetxe et al., 2019, Tran, 2020, Dobler and De Melo, 2023, Liu et al., 2023a].

To initialize the token embeddings from new languages, methods like WECHSEL [Minixhofer et al., 2021] and OFA [Liu et al., 2023a] utilize external information sources such as static multilingual word vectors. These methods identify similar token embeddings from LM's supported languages and generate new token embeddings by combining the similar embeddings linearly. However, combining

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

LM's supported language tokens can be limiting, especially when introducing new tokens for a new language with very different semantic meanings. Additionally, the use of linear combinations may restrict the ability of these methods.

To address these challenges, this study introduces HyperOFA. Like previous methods, HyperOFA utilizes static multilingual word vectors as an external information source. However, it trains a hypernetwork to learn the relationship between the word vectors and the LM's existing token embeddings. During initialization, the hypernetwork generates new token embeddings by taking external word vectors as input. To empirically test the method, the vocabulary size of RoBERTa (50K tokens) ([Liu, 2019]) and XLM-RoBERTa (250K tokens) ([Conneau, 2019]) models were extended to 401K tokens. The results were compared against a random initialization baseline and the OFA method. The HyperOFA code-base is available at `https://github.com/enesozeren/hyper-ofa`.

2 Related Work

Tokenization Language models typically use embedding vectors that encode the semantic information of basic units of text, known as tokens. Tokens can be character level, word level, or more commonly, sub-word level. The process of generating tokens from given text is performed by a tokenizer which is trained on a dataset similar to LM's pre-training dataset but a smaller one ([Radford et al., 2019]). The token set is referred as the LM's vocabulary. This vocabulary consists of units from the languages in the smaller dataset, potentially, limiting the LMs ability to learn new languages through continued pre-training. To address this, Glot500-m ([Imani et al., 2023]) expands the vocabulary of XLM-RoBERTa ([Conneau, 2019]) from 250K tokens to 401K tokens, performs continued pre-training and demonstrates superior performance on both high- and low-resource languages.

Introducing New Tokens To improve the ability of LLMs to handle languages beyond those included in their original pre-training dataset, recent research has proposed various methods. One promising approach is to introduce new tokens for the new languages and then performing continued pre-training. These new tokens' embeddings are initialized in different ways in the literature, which this paper categorizes as either random or wise initialization.

Random Token Initialization The simplest approach for initializing new token embeddings is random initialization. de Vries and Nissim [2020] and Artetxe et al. [2019] randomly initialize the new token embeddings and perform continued pre-training by freezing model weights and training only the embedding layer. Similarly, ([Imani et al., 2023]) employ random initialization but perform continued pre-training across all model parameters. However, random initialization requires more computational resources and training data to achieve good performance compared to wise initialization methods ([Liu et al., 2023a]).

Wise Token Initialization Wise initialization methods generally uses a external source of information to initialize the new embeddings better than random. An early work, Tran [2020] created a bilingual LM by introducing new tokens from a new language by using a parallel or non-parallel corpus. The method learns a sparse word translation matrix to initialize the new token embeddings.

The WECHSEL method ([Minixhofer et al., 2021]) proposes a wise initialization technique by using an external source of information to transform a monolingual model to another monolingual model in a different language. The method utilizes multilingual static word vectors which contain both the source and target language. First, the method matches the words in the external source to the tokens of source and target language if they are contained in the word. Then, using this matching information, both the source and target tokens are projected into the external word vector space. By considering the cosine similarity of the tokens in this shared space, the target token embeddings are initialized as the convex combination of similar source token embeddings. The paper applies this method for nine different languages.

Liu et al. [2023a] introduced the OFA framework that builds upon the token initialization method from the WECHSEL approach. However, OFA expends the vocabulary of both monolingual and multilingual models to cover significantly larger number of languages. To prevent the embedding matrix from becoming disproportionately large compared to the rest of the model, OFA employs a factorization technique which was proposed in ALBERT ([Lan, 2019]). The method leverages

ColexNet+ word vectors [Liu et al., 2023b] as external word vectors and utilizes Glot500-m tokenizer for introducing the new tokens.

Similar to WECHSEL and OFA methods, HyperOFA also utilizes an external source of information, specifically ColexNet+ word vectors, as used in OFA. However, the key distinction of HyperOFA lies in its use of a hypernetwork to directly predict embeddings for new token initialization, whereas the previous two method initialize new tokens by combining embeddings of existing ones. This approach can be particularly advantageous for initializing tokens from different languages which have very different semantics than the model's original languages. This paper directly compares HyperOFA with OFA method under experiment settings similar with those in the OFA paper to evaluate the impact of this different approach.

Hypernetworks Hypernetworks are networks designed to generate the weights of another network. A recent survey by Chauhan et al. [2024] highlights their application accross various domains of deep learning. One of the earlier works in initializing embeddings with hypernetworks is MIMICK ([Pinter et al., 2017]) which focuses on out-of-vocabulary (OOV) words for word vectors. The method uses an RNN architecture for the hypernetwork and predicts the OOV word embeddings. On the other hand, Schick and Schütze [2019] integrates a hypernetwork into BERT ([Devlin, 2018]) to generate embeddings for rare words. More recently, Minixhofer et al. [2024] proposed a hypernetwork-based method for zero-shot tokenizer transfer, enabling a language model to detach from its tokenizer and work with any tokenizer.

3 Methodology

The HyperOFA method builds upon certain aspects of the OFA method. To provide a clear understanding, similar notation used in OFA paper is used. The HyperOFA pipeline can be seen in the Figure 1.

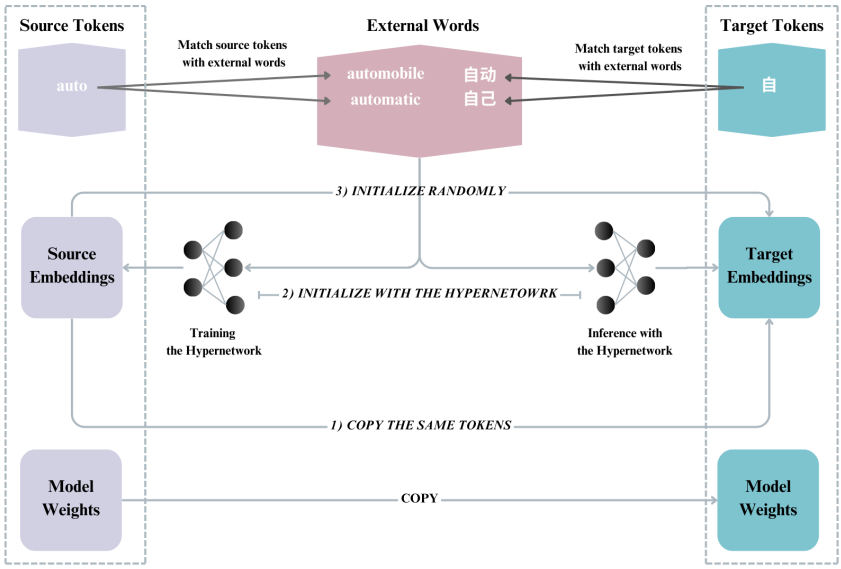


Figure 1: HyperOFA pipeline. The source model (left) transfers weights to the target model (right). The target embeddings are initialized by first copying embeddings for matching tokens, then generating embeddings via a hypernetwork for tokens with matching external words, and finally randomly initializing the rest.

3.1 Problem Setting

Given a model with a source tokenizer TOK^s, the goal is to replace the source tokenizer with a target tokenizer TOK^t which supports broader range of tokens across various languages. The source vocabulary size V^s of TOK^s is smaller than target vocabulary size V^t of TOK^t. Starting from

the source embeddings $E^s \in \mathbb{R}^{|V^s| \times D}$, the HyperOFA method aims to initialize target embeddings $E^t \in \mathbb{R}^{|V^t| \times D}$ wisely by keeping the same embedding dimension D.

3.2 Factorization

Since $V^t > V^s$ the number of embedding parameters grows signiciantly from $V^s \times D$ to $V^t \times D$ in the target model. This can result having large ratio of model parameters in the embedding matrix. To address this, Lan [2019] and Liu et al. [2023a] use factorization method.

Factorization decomposes the E^s into two smaller matrices using the Singular Value Decomposition (SVD) metod, such that $E^s \approx F^s P$ where $F^s \in \mathbb{R}^{|V^s| \times D'}$ is the coordinate matrix containing token-specific parameters, and $P \in \mathbb{R}^{D' \times D}$ is the primitive embedding matrix capturing language-agnostic features. When $D' < D$, the total number of parameters of F^s and P is smaller than E^s. After factorizing E^s, the goal is to initialize the token-specific embeddings in $F^t \in \mathbb{R}^{|V^t| \times D'}$ for the TOK^t while reusing the same P.

Liu et al. [2023a] experiments with different D' values to explore the trade-off between parameter saving and model performance.

3.3 External Word Vectors

As an external source of information for wise initialization, Colexnet+ ([Liu et al., 2023b]), static word vectors $W \in \mathbb{R}^{4million \times 200}$, is used in this method. Colexnet+ contains over four million words spanning more than one thousand languages. To utilize these words vectors, the tokens from TOK^s and TOK^t are matched with the words in W. Specifically, a token will be matched with a word if that word contains the token (see Figure 1). The resulting set of matched word vectors will be represented as $W(s_i)$ for token i from TOK^s and $W(t_j)$ for token j from TOK^t.

3.4 Hypernetwork

After performing factorization and creating the set of matched words and tokens, a hypernetwork HN_θ with parameters θ will be used. The aim of this hypernetwork is to generate the token embedding F_j by using the matched word vectors $W(s_j)$ where $j \in V^t$. The training set for HN_θ is ($W(s_i)$, F_i) in which the inputs are the matched word vectors and the target is the token embedding in the source coordinate matrix F^s. Since the number of matched word vectors varies for each token, at least one vector is always present, though multiple word vectors are typically used to predict the target token embedding.

During training, a custom loss function is employed to ensure that the predicted token embeddings not only align semantically with the target embeddings but also maintain a similar norm. This loss combines contrastive loss, which enforces semantic similarity, and normalized L1 loss, which preserves magnitude consistency.

$$\mathcal{L}(\theta) = \lambda \cdot \mathcal{L}_{\text{contrastive}} + (1 - \lambda) \cdot \mathcal{L}_{\text{L1}}$$

where $\mathcal{L}_{\text{contrastive}} = -\frac{1}{N} \sum_{k=1}^{N} \log \frac{\exp(\text{sim}(F_k^s, \hat{F}_k^s)/T)}{\sum_{l=1}^{N} \exp(\text{sim}(F_l^s, \hat{F}_l^s)/T)}$ and $\mathcal{L}_{\text{L1}} = \frac{1}{N} \sum_{k=1}^{N} \frac{\|F_k^s - HN_\theta(W(s_k))\|_1}{-\frac{1}{N}\sum_{l=1}^{N} \|F_l^s\|}$

Here the N is batch size, sim is cosine similarity, T is temperature and λ is a parameter to balance loss components. The $\mathcal{L}_{\text{L1}}$ component is a normalized by average embedding value to prevent very small loss values.

When designing the model architecture for HN_θ, the input requirements played a crucial role. First, the number of matched word vectors varies for across tokens, meaning the HN_θ architecture must be capable of handling variable-length inputs. Secondly, since the order of the input matched word vectors should not influence the prediction, the model architecture should be permutation-invariant. Initially, an encoder only transformer model ([Vaswani, 2017] without positional encoding layers (called as Setformer in this study) that satisfies both requirements was tested. However, after observing poor performance, the approach shifted to a BiLSTM architecture ([Schuster and Paliwal,

1997]) despite it not inherently satisfying the permutation-invariance requirement. Experimental results demonstrated that BiLSTM works better for this task when compared to a transfomer encoder model without positional encoding layer (see Apendix Table 4 for experiments). To address the BiLSTM's sensitivity to input order, data augmentation was implemented by randomly shuffling the word vectors' order during each training epoch, effectively preventing the model from overfitting to specific sequence arrangements.

The trained BiLSTM-based hypernetwork HN_θ generates target embeddings by taking matched word vectors $W(t_j)$ as inputs, with the network parameters optimized using the specified loss function.

3.5 New Token Initialization

The target embeddings, F^t, are initialized in three steps similar to the OFA method (see Figure 1).

1. For tokens that overlap between TOK^s and TOK^t, the token embeddings in F^s are directly copied to F^t.
2. For tokens with matched words in W, their embeddings are predicted by HN_θ using the matched word vectors $W(t_j)$ as input.
3. For the remaining tokens, the embeddings are randomly initialized from a normal distribution with the same mean and standard deviation as the source embeddings.

4 Hypernetwork Training

In this study, the vocabularies of RoBERTa-base (RoBERTa) [Liu, 2019] (50K tokens) and XLM-RoBERTa-base (XLM-R) [Conneau, 2019] (250K tokens) are extended to match the Glot500-m vocabulary [Imani et al., 2023], which contains 401K tokens.

Before initializing new token embeddings, the factorization method described in Section 3.2 is applied for different dimensions D'. While the original embedding size for both models is 768, factorization is used to create coordinate matrices F^s with dimensions of 100, 200, and 400. The models with reduced embedding sizes are referred to as RoBERTa-100, RoBERTa-200, RoBERTa-400, and similarly, XLM-R-100, XLM-R-200, XLM-R-400.

To compare the performance of HyperOFA with OFA, different hypernetworks are trained to initialize the new token embeddings for these RoBERTa-xxx and XLM-R-xxx models.

Hypernetwork Training Dataset For the RoBERTa model, the hypernetwork training dataset consists of 22K word vector and target embedding pairs, as only 22K out of RoBERTa's 50K vocabulary tokens match with the $\overrightarrow{\text{Colex}}$Net+ words. Similarly, for XLM-R, the training dataset contains 103K pairs, corresponding to the 103K matched tokens from its 250K vocabulary. To mitigate overfitting, data augmentation is applied by shuffling word vectors before each epoch. Additionally, with 50% probability, the number of word vectors is randomly limited to 50–100% of the available vectors. A custom sampler is also employed to ensure that batches contain input sequences of similar length, improving training stability.

Hypernetwork Training Hyperparameters As described in Section 3.4, the hypernetworks follow a BiLSTM architecture. All RoBERTa-xxx and XLM-R-xxx hypernetworks share the same configuration: a maximum context size of 256, a dropout rate of 0.4, and an Adam optimizer. The learning rate starts at 1×10^{-4} and decays linearly by a factor of 0.95 every 10 epochs. Training was conducted on two Nvidia A100 GPUs, with each model requiring approximately 1 to 1.5 hours.

To ensure healthy training, the hyperparameters in the loss function, as explained in Section 3.4, were set as follows: $\lambda = 0.1$ for all hypernetworks, and $T = 0.5$ for the hypernetworks of RoBERTa-xxx, and $T = 0.25$ for the hypernetworks of XLM-R-xxx.

As the dimension of the predicted embedding increased, a hypernetwork with higher capacity was necessary. To address this, the hidden dimension of the BiLSTM was increased for embeddings with 400 dimensions. Further details are provided in Table 1. All models were trained until the validation loss converged.

Notably, the hypernetworks have a substantial number of parameters compared to the LLMs they initialize. Experiments showed that larger hypernetworks, when combined with strong regularization (dropout of 0.4 and the data augmentation methods mentioned earlier), were performing better than smaller hypernetworks. For a detailed comparison, see Appendix Figure 3.

LM	Hypernetwork	Training Data	Layers	Hid Dim	Param	Epoch
RoBERTa-100	HN-R-100	22K	2	800	22.1M	370
RoBERTa-200	HN-R-200	22K	2	800	22.5M	470
RoBERTa-400	HN-R-400	22K	2	1600	87.4M	400
XLM-R-100	HN-X-100	103K	4	800	52.9M	120
XLM-R-200	HN-X-200	103K	4	800	52.3M	230
XLM-R-400	HN-X-400	103K	4	1600	210.3M	80

Table 1: Hypernetwork model details for predicting the new tokens for RoBERTa and XLM-R language models with different factorized dimensions. Epochs column indicated the converged epoch number for the hypernetwork.

5 Results

With the hypernetworks explained in Section 4, RoBERTa and XLM-R models' embeddings are initialized with the HyperOFA method. Also those 2 LMs are initialized with OFA and random initialization methods as well. The models are described as follows:

HyperOFA-mono-xxx These are RoBERTa models with an extended vocabulary, expanding from 50K tokens to 401K tokens. The "xxx" denotes the embedding dimension of the model, and the "mono" suffix indicates that the RoBERTa model is originally monolingual.

HyperOFA-multi-xxx These are XLM-R models with an extended vocabulary, expanding from 250K tokens to 401K tokens. The "xxx" denotes the embedding dimension of the model, and the "multi" suffix indicates that the XLM-R model is originally multilingual.

OFA-mono-xxx RoBERTa models with extened vocabulary with the OFA method by using Liu et al. [2023a] code-base.

OFA-multi-xxx XLM-R models with extened vocabulary with the OFA method by using Liu et al. [2023a] code-base.

Random-mono-xxx RoBERTa models with extened vocabulary with the random initialization method. In this approach all overlapping tokens are copied, while the remaining tokens are randomly initialized with old embedding's mean and standard deviations are used.

Random-multi-xxx XLM-R models with extened vocabulary with the random initialization method. In this approach all overlapping tokens are copied, while the remaining tokens are randomly initialized with old embedding's mean and standard deviations are used.

The details of the volume of tokens initialized with each step explained in Section 3.5 can be found in Table 2.

5.1 Evaluations

The performance of language models (LMs) with extended vocabularies is compared across three approaches: Random, OFA, and HyperOFA. Evaluations are conducted on four benchmarks under two conditions: without continued pre-training and with continued pre-training.

Sentence Retrieval Retrieval performance is assessed using the Sentence Retrieval Tatoeba (SR-T) [Artetxe and Schwenk, 2019] and Sentence Retrieval Bible (SR-B) benchmarks. Following Liu et al.

Method	Model	Copy	HyperOFA	OFA	Random	Total
HyperOFA	RoBERTa	27K	179K	0	195K	401K
	XLM-R	255K	84K	0	62K	401K
OFA	RoBERTa	27K	0	179K	195K	401K
	XLM-R	255K	0	84K	62K	401K
Random	RoBERTa	27K	0	0	374K	401K
	XLM-R	255K	0	0	146K	401K

Table 2: Distribution of tokens initialized using HyperOFA, OFA, and random initialization methods. The "Copy" column represents the number of tokens shared between the old and new vocabularies, which are directly copied. This distribution holds for all variants with different embedding factorization dimensions (100, 200, 400).

[2023a], Top-10 accuracy is used as the evaluation metric, where the correct translation must be among the ten nearest neighbors of an English sentence. Sentence representations are obtained by averaging contextualized word embeddings from the model's 8th layer.

Sequence Labeling For sequence labeling, named entity recognition (NER) and part-of-speech tagging (POS) are evaluated using the WikiANN [Pan et al., 2017] and Universal Dependencies [De Marneffe et al., 2021] datasets, respectively. The methodology follows Liu et al. [2023a], where models are fine-tuned on the English training set. The best checkpoint, selected based on validation performance, is then used to report zero-shot performance on test sets in other languages. F1 scores are reported for both benchmarks.

5.1.1 Performance before continued pre-training

The models' embedding matrices are extended, and the newly added tokens are initialized using three different methods: Random, OFA, and HyperOFA. Test metrics for the benchmarks are then computed without any continued pre-training. The benchmark results are presented in Table 3.

The evaluation of vocabulary extension methods is conducted across all available languages in the benchmarks. SR-B covers 98 languages, SR-T covers 369 languages, NER covers 164 languages, and POS covers 91 languages.

The results without continued pre-training in Table 3 show that HyperOFA and OFA methods outperforms the Random initialization method in all cases. For the retrievel tasks (SR-B and SR-T), HyperOFA performs better than OFA on all cases except mono-400 models. This is because increasing the embedding dimension (from 100 to 200 and 400) leads to a decline in HyperOFA's performance. With a fixed amount of training data, learning higher-dimensional embeddings becomes more challenging for the hypernetwork. In mono models, where the hypernetwork had only 20K training observations, HyperOFA-mono-400 underperforms compared to OFA-mono-400 across all four benchmarks. However, HyperOFA-multi-400 performs competitively with OFA-multi-400, as the hypernetwork had access to 100K training observations in the multi setting. For sequence labeling tasks (NER and POS), OFA generally outperforms HyperOFA, though HyperOFA occasionally achieves better results.

5.1.2 Performance after continued pre-training

For continued pre-training, a subset of languages from Glot500-c ([Imani et al., 2023]), comprising 22 languages spanning high, mid, and low-resource categories is used. The full list of languages and their data volumes can be found in Appendix - Table 5. This dataset contains 1.1 billion tokens across 36 million sentences.

Continued pre-training is crucial because, even with carefully initialized embeddings, they must be fine-tuned on data from the new language tokens. Additionally, the model parameters need to adapt to these languages. To achieve this, six models were continued pre-training: RoBERTa-100 and XLM-R-400, each initialized with three different methods—HyperOFA, OFA, and Random initialization.

LM	SR-B	SR-T	NER	POS
Random-mono-100	3.5	4.6	23.4	22.5
OFA-mono-100	4.5	6.2	**25.0**	**23.5**
HyperOFA-mono-100	**5.0**	**6.4**	24.3	22.5
Random-mono-200	3.7	5.2	24.9	23.1
OFA-mono-200	4.5	7.2	**25.7**	**23.4**
HyperOFA-mono-200	**4.8**	**7.5**	25.3	**23.4**
Random-mono-400	4.1	5.3	25.8	23.0
OFA-mono-400	**4.8**	**7.2**	**26.1**	**24.5**
HyperOFA-mono-400	4.7	6.3	25.8	23.0
Random-multi-100	5.1	7.2	34.7	41.5
OFA-multi-100	5.1	7.5	36.3	**42.3**
HyperOFA-multi-100	**5.2**	**7.6**	**37.6**	**42.3**
Random-multi-200	5.7	10.0	38.1	47.3
OFA-multi-200	5.7	10.4	**40.2**	**48.6**
HyperOFA-multi-200	**6.0**	1**0.6**	38.3	48.3
Random-multi-400	5.6	21.0	41.4	53.7
OFA-multi-400	5.9	**21.3**	**43.3**	**54.6**
HyperOFA-multi-400	**6.1**	**21.3**	42.6	53.8

Table 3: Performance comparison of OFA and HyperOFA in without continued pre-training setting. The values for OFA models are taken from Liu et al. [2023a] directly. SR-B covers 98 languages, SR-T covers 369 languages, NER covers 164 languages, and POS covers 91 languages. For SR-B and SR-T top 10 accuracy, for NER and POS f1 score is presented. All metrics are average across languages.

All six models were continued pre-trained using hyperparameters similar to those in Liu et al. [2023a], with three key differences: smaller dataset, an effective batch size of 512 instead of 384 and training on four Nvidia H100 GPUs. The continued pre-training was conducted for 4,000 steps (approx. 1 epoch) and required at most 1.5 hours to complete.

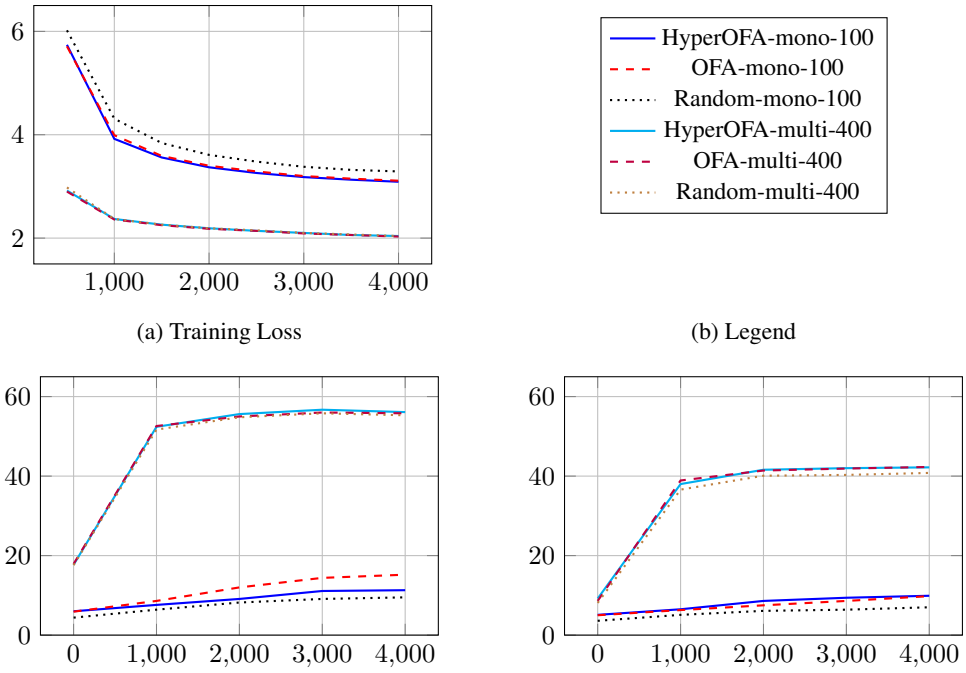
The results of training loss and retrieval benchmark evaluations are presented in Figure 2. The first observation is that all XLM-R-based models outperform the RoBERTa-based models. This aligns with expectations, as XLM-R is a multilingual model, whereas RoBERTa is English only. Hence, XLM-R is better at adaptation to new languages. Secondly, models with embeddings initialized using the HyperOFA and OFA methods achieve superior performance compared to those with randomly initialized embeddings. Finally, the performance trends between HyperOFA and OFA remain consistent across all cases, with neither method showing a decisive advantage overall. While OFA performs better on the SR-T benchmark, HyperOFA achieves slightly higher scores on the SR-B benchmark. This suggests that both approaches are effective, with their relative strengths depending on the specific evaluation metric.

6 Limitations

In this study, as well as in Liu et al. [2023a], HyperOFA and OFA were explored for initializing new embeddings in encoder-only models. While both methods are theoretically applicable to decoder-only models like GPT [Radford et al., 2019] and encoder-decoder models like T5 [Raffel et al., 2020], their effectiveness in these settings remains untested, presenting an open research direction.

Another limitation concerns the embedding dimensions used in this study. Due to the embedding matrix factorization described in Section 3.2, the dimensions are relatively low compared to those in modern LLMs. While this approach reduces computational costs, it leaves open the question of how HyperOFA would perform with much higher-dimensional embeddings.

Finally, the continued pre-training dataset used in this study was relatively small compared to that of Liu et al. [2023a] due to disk quota limitations in this study. Exploring the impact of larger continued pre-training datasets, especially those having multiple languages, could provide deeper insights into the strengths and weaknesses of both methods in different settings.



(a) Training Loss

(b) Legend

(c) SR-T Top 10 Accuracy

(d) SR-B Top 10 Accuracy

Figure 2: Training losses and evaluation metrics for models with initialized embeddings with HyperOFA, OFA, and random initialization during continued pre-training. (a) shows the training loss values, (b) provides the legend for all three plots, (c) presents SR-T average Top-10 accuracy for languages included in the continued pre-training set (20 languages out of 22 continued pre-training languages), and (d) shows SR-B average Top-10 accuracy for same languages as in SR-T.

7 Conclusion

This study introduced HyperOFA as a method for expanding the vocabulary of LLMs to new languages using a hypernetwork model. HyperOFA leverages external static multilingual word vectors as an external information source to initialize new token embeddings. Experiments across four different benchmarks demonstrated that HyperOFA consistently outperforms the random initialization baseline and performs competitively with the OFA method. These results highlight HyperOFA as a promising approach, alongside OFA, for efficient new token embedding initialization.

Acknowledgements

I sincerely thank Mina Rezaei for generously providing access to computational resources and constructive feedback. I'm also deeply grateful to Yihong Liu for the thought-provoking discussions, invaluable insights, and assistance with the datasets.

References

Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the association for computational linguistics*, 7: 597–610, 2019.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. *arXiv preprint arXiv:1910.11856*, 2019.

Vinod Kumar Chauhan, Jiandong Zhou, Ping Lu, Soheila Molaei, and David A Clifton. A brief review of hypernetworks in deep learning. *Artificial Intelligence Review*, 57(9):250, 2024.

A Conneau. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

Marie-Catherine De Marneffe, Christopher D Manning, Joakim Nivre, and Daniel Zeman. Universal dependencies. *Computational linguistics*, 47(2):255–308, 2021.

Wietse de Vries and Malvina Nissim. As good as new. how to successfully recycle english gpt-2 to make models for other languages. *arXiv preprint arXiv:2012.05628*, 2020.

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Konstantin Dobler and Gerard De Melo. Focus: Effective embedding initialization for monolingual specialization of multilingual models. *arXiv preprint arXiv:2305.14481*, 2023.

Ayyoob Imani, Peiqin Lin, Amir Hossein Kargaran, Silvia Severini, Masoud Jalili Sabet, Nora Kassner, Chunlan Ma, Helmut Schmid, André Martins, François Yvon, and Hinrich Schütze. Glot500: Scaling multilingual corpora and language models to 500 languages. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1082–1117, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.61. URL `https://aclanthology.org/2023.acl-long.61/`.

Zhenzhong Lan. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Yihong Liu, Peiqin Lin, Mingyang Wang, and Hinrich Schütze. Ofa: A framework of initializing unseen subword embeddings for efficient large-scale multilingual continued pretraining. *arXiv preprint arXiv:2311.08849*, 2023a.

Yihong Liu, Haotian Ye, Leonie Weissweiler, and Hinrich Schuetze. Transfer learning for low-resource languages based on multilingual colexification graphs. *arxiv*, 2023b. URL `https://arxiv.org/abs/2305.12818`.

Yinhan Liu. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364, 2019.

Benjamin Minixhofer, Fabian Paischer, and Navid Rekabsaz. Wechsel: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. *arXiv preprint arXiv:2112.06598*, 2021.

Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. *arXiv preprint arXiv:2405.07883*, 2024.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 1946–1958, 2017.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. Mimicking word embeddings using subword rnns. *arXiv preprint arXiv:1707.06961*, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Timo Schick and Hinrich Schütze. Bertram: Improved word embeddings have big impact on contextualized model performance. *arXiv preprint arXiv:1910.07181*, 2019.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

Ke Tran. From english to foreign languages: Transferring pre-trained language models. *arXiv preprint arXiv:2002.07306*, 2020.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

A Experiments for Hypernetwork

A.1 Large vs Small Hypernetwork

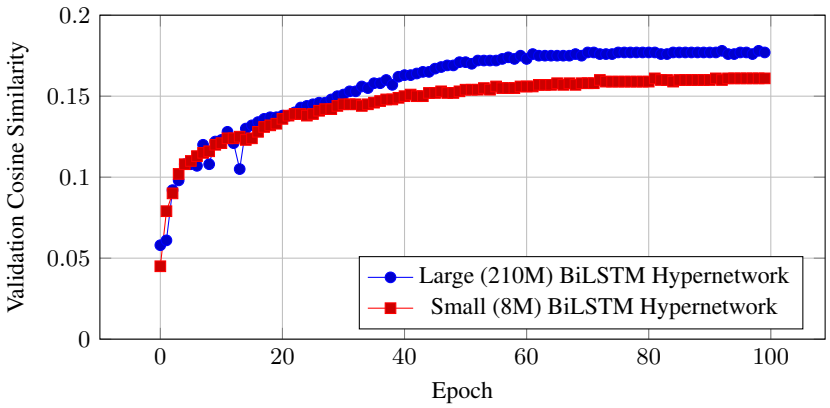


Figure 3: Comparison of large (210M parameters) and small (8M parameters) BiLSTM-based hypernetwork models in terms of validation cosine similarity between predicted and true embeddings over 100 epochs for the XLM-R embeddings.

A.2 BiLSTM vs Setformer

Table 4 compares two candidate hypernetwork architectures for initializing token embeddings in the RoBERTa model with factorized embeddings (100 dimensional). Setformer (a Transformer encoder without positional encodings) and a BiLSTM (Bidirectional LSTM) are evaluated with two different parameter sizes.

Results show that LLMs initialized with smaller hypernetworks (8M parameters) perform similarly and better than a randomly initialized baseline. However, increasing the hypernetwork size to 22M parameters leads to a notable performance boost. Specifically, the LLM initialized with the larger BiLSTM hypernetwork achieves the highest SR-T Top 10 accuracy (6.4), outperforming both Setformer variants. This suggests that BiLSTM is more effective than Setformer as a hypernetwork, especially when given sufficient capacity.

LM	Hypernetwork	Hypernetwork Param.	LLM SR-T Top 10 Acc
HyperOFA-mono-100	Setformer	8M	5.7
HyperOFA-mono-100	BiLSTM	7M	5.6
HyperOFA-mono-100	Setformer	22M	5.2
HyperOFA-mono-100	BiLSTM	22M	**6.4**
Random-mono-100	Random Init	-	4.6

Table 4: Comparison of Setformer (Transformer encoder without positional encodings) and BiLSTM as hypernetworks for initializing token embeddings in HyperOFA-mono-100, a RoBERTa-based model with a new vocabulary and factorized embedding dimension of 100. The SR-T Top 10 Accuracy is reported without continued pre-training. Also random initialization baseline performance is given at the last row.

B Continued Pre-training Dataset

The continued pre-training dataset was deliberately kept smaller than that used by Liu et al. [2023a] due to disk quota limitations in the HyperOFA study. The languages, their original sentence counts in Glot500-c [Imani et al., 2023] dataset and the sentence counts used in this study is listed in Table 5. For continued pre-training 36M sentences (approx. 1.1B tokens) across 22 languages are used. To categorize source category, thresholds used: high (>5M sentences), mid (>500K sentences), and low (<500K sentences).

Source Category	Language	Glot500-c Sentence Count	Subsampled Sentence Count
High	eng_Latn	36,121,560	5,000,000
	tur_Latn	29,182,577	5,000,000
	ell_Grek	22,031,905	5,000,000
	bul_Cyrl	21,822,051	5,000,000
	ces_Latn	20,374,860	5,000,000
	kor_Hang	6,348,091	5,000,000
Mid	kat_Geor	990,785	990,785
	fry_Latn	925,801	925,801
	zsm_Latn	849,033	849,033
	khm_Khmr	565,794	565,794
	jpn_Japn	507,538	507,538
Low	yue_Hani	483,750	483,750
	tuk_Latn	312,480	312,480
	uig_Arab	298,694	298,694
	pam_Latn	292,293	292,293
	kab_Latn	166,953	166,953
	gla_Latn	124,953	124,953
	mhr_Cyrl	91,557	91,557
	swh_Latn	43,876	43,876
	cmn_Hani	57,500	57,500
	pes_Arab	18,762	18,762
	dtp_Latn	1,355	1,355
	Total Sentence Count	141,612,168	35,731,124

Table 5: Distribution of continued pre-trainig data across. The table shows the original Glot500-c volume and sub sampled volume for each language, grouped by their source category (High, Mid, Low).

C Benchmark Language Coverage

C.1 For Benchmark Performances in Table 3

SR-B Benchmark Languages:

mal_Mlym, aze_Latn, guj_Gujr, ben_Beng, kan_Knda, tel_Telu, mlt_Latn, fra_Latn, spa_Latn, fil_Latn, nob_Latn, rus_Cyrl, deu_Latn, tur_Latn, pan_Guru, mar_Deva, por_Latn, nld_Latn, zho_Hani, ita_Latn, ind_Latn, ell_Grek, bul_Cyrl, swe_Latn, ces_Latn, isl_Latn, pol_Latn, ron_Latn, dan_Latn, hun_Latn, tgk_Cyrl, srp_Latn, fas_Arab, ceb_Latn, heb_Hebr, hrv_Latn, fin_Latn, slv_Latn, vie_Latn, mkd_Cyrl, slk_Latn, nor_Latn, est_Latn, ltz_Latn, eus_Latn, lit_Latn, kaz_Cyrl, lav_Latn, epo_Latn, cat_Latn, tha_Thai, ukr_Cyrl, tgl_Latn, sin_Sinh, gle_Latn, hin_Deva, kor_Hang, ory_Orya, urd_Arab, sqi_Latn, bel_Cyrl, afr_Latn, nno_Latn, tat_Cyrl, hau_Latn, sna_Latn, msa_Latn, som_Latn, srp_Cyrl, mlg_Latn, zul_Latn, arz_Arab, nya_Latn, tam_Taml, hat_Latn, uzb_Latn, sot_Latn, uzb_Cyrl, als_Latn, amh_Ethi, sun_Latn, war_Latn, yor_Latn, fao_Latn, uzn_Cyrl, smo_Latn, bak_Cyrl, ilo_Latn, tso_Latn, mri_Latn, asm_Beng, hil_Latn, nso_Latn, ibo_Latn, kin_Latn, hye_Armn, lin_Latn, tpi_Latn, twi_Latn, kir_Cyrl, pap_Latn, nep_Deva, bcl_Latn, xho_Latn, cym_Latn, gaa_Latn, ton_Latn, lat_Latn, srn_Latn, ewe_Latn, bem_Latn, efi_Latn, bis_Latn, haw_Latn, hmo_Latn, kat_Geor, pag_Latn, loz_Latn, fry_Latn, mya_Mymr, nds_Latn, run_Latn, rar_Latn, fij_Latn, ckb_Arab, ven_Latn, zsm_Latn, chv_Cyrl, sag_Latn, guw_Latn, bre_Latn, toi_Latn, che_Cyrl, pis_Latn, oss_Cyrl, nan_Latn, tuk_Latn, tir_Ethi, yua_Latn, min_Latn, khm_Khmr, tum_Latn, lug_Latn, tzo_Latn, mah_Latn, jav_Latn, jpn_Jpan, lus_Latn, crs_Latn, ndo_Latn, snd_Arab, yue_Hani, kua_Latn, hin_Latn,

kal_Latn, tdt_Latn, mfe_Latn, mos_Latn, kik_Latn, cnh_Latn, gil_Latn, pon_Latn, ori_Orya, luo_Latn, nzi_Latn, gug_Latn, bar_Latn, bci_Latn, chk_Latn, yap_Latn, ssw_Latn, quz_Latn, sah_Cyrl, tsn_Latn, quy_Latn, bbc_Latn, wal_Latn, uig_Arab, pam_Latn, seh_Latn, zai_Latn, gym_Latn, bod_Tibt, nde_Latn, fon_Latn, nbl_Latn, kmr_Latn, guc_Latn, mam_Latn, nia_Latn, nyn_Latn, cab_Latn, top_Latn, mco_Latn, tzh_Latn, plt_Latn, iba_Latn, kek_Latn, sop_Latn, kac_Latn, qvi_Latn, cak_Latn, kbp_Latn, ctu_Latn, kri_Latn, mau_Latn, tyv_Cyrl, btx_Latn, nch_Latn, ncj_Latn, pau_Latn, toj_Latn, pcm_Latn, dyu_Latn, kss_Latn, quc_Latn, yao_Latn, kab_Latn, tuk_Cyrl, ndc_Latn, san_Deva, qug_Latn, arb_Arab, mck_Latn, arn_Latn, pdt_Latn, gla_Latn, kmr_Cyrl, nav_Latn, ksw_Mymr, mxv_Latn, hif_Latn, wol_Latn, sme_Latn, gom_Latn, bum_Latn, mgr_Latn, ahk_Latn, tsz_Latn, bzj_Latn, udm_Cyrl, cce_Latn, meu_Latn, cbk_Latn, bhw_Latn, ngu_Latn, nyy_Latn, naq_Latn, toh_Latn, nse_Latn, alz_Latn, mhr_Cyrl, djk_Latn, gkn_Latn, grc_Grek, swh_Latn, alt_Cyrl, miq_Latn, kaa_Cyrl, lhu_Latn, lzh_Hani, cmn_Hani, kjh_Cyrl, mgh_Latn, rmy_Latn, srm_Latn, gur_Latn, yom_Latn, cfm_Latn, lao_Laoo, qub_Latn, ote_Latn, ldi_Latn, ayr_Latn, bba_Latn, aln_Latn, leh_Latn, ban_Latn, ace_Latn, pes_Arab, ary_Arab, hus_Latn, glv_Latn, mai_Deva, dzo_Tibt, ctd_Latn, nnb_Latn, sxn_Latn, mps_Latn, gkp_Latn, acr_Latn, dtp_Latn, lam_Latn, poh_Latn, quh_Latn, tob_Latn, ach_Latn, npi_Deva, myv_Cyrl, tih_Latn, gor_Latn, ium_Latn, teo_Latn, kia_Latn, crh_Cyrl, enm_Latn, mad_Latn, cac_Latn, hnj_Latn, ikk_Latn, sba_Latn, zom_Latn, bqc_Latn, bim_Latn, mdy_Ethi, bts_Latn, gya_Latn, agw_Latn, knv_Latn, giz_Latn, hui_Latn, hif_Deva

SR-T Benchmark Languages:

mal_Mlym, aze_Latn, ben_Beng, tel_Telu, fra_Latn, spa_Latn, nob_Latn, rus_Cyrl, deu_Latn, tur_Latn, mar_Deva, por_Latn, nld_Latn, ara_Arab, ita_Latn, ind_Latn, ell_Grek, bul_Cyrl, swe_Latn, ces_Latn, isl_Latn, pol_Latn, ron_Latn, dan_Latn, hun_Latn, srp_Latn, ceb_Latn, heb_Hebr, hrv_Latn, glg_Latn, fin_Latn, slv_Latn, vie_Latn, mkd_Cyrl, slk_Latn, est_Latn, eus_Latn, lit_Latn, kaz_Cyrl, bos_Latn, epo_Latn, cat_Latn, tha_Thai, ukr_Cyrl, tgl_Latn, gle_Latn, hin_Deva, kor_Hang, urd_Arab, sqi_Latn, bel_Cyrl, afr_Latn, nno_Latn, tat_Cyrl, ast_Latn, mon_Cyrl, arz_Arab, tam_Taml, uzb_Cyrl, amh_Ethi, war_Latn, fao_Latn, hye_Armn, oci_Latn, xho_Latn, cym_Latn, lat_Latn, kat_Geor, fry_Latn, nds_Latn, zsm_Latn, bre_Latn, tuk_Latn, khm_Khmr, jpn_Jpan, yue_Hani, gsw_Latn, lvs_Latn, kur_Latn, ido_Latn, uig_Arab, pam_Latn, pms_Latn, wuu_Hani, yid_Hebr, ina_Latn, kab_Latn, gla_Latn, cbk_Latn, hsb_Latn, mhr_Cyrl, swh_Latn, cmn_Hani, pes_Arab, dtp_Latn, lfn_Latn, ile_Latn, csb_Latn.

NER Benchmark Languages:

hbs_Latn, mal_Mlym, aze_Latn, guj_Gujr, ben_Beng, kan_Knda, tel_Telu, mlt_Latn, fra_Latn, spa_Latn, eng_Latn, rus_Cyrl, deu_Latn, tur_Latn, pan_Guru, mar_Deva, por_Latn, nld_Latn, ara_Arab, zho_Hani, ita_Latn, ind_Latn, ell_Grek, bul_Cyrl, swe_Latn, ces_Latn, isl_Latn, pol_Latn, ron_Latn, dan_Latn, hun_Latn, tgk_Cyrl, fas_Arab, ceb_Latn, heb_Hebr, hrv_Latn, glg_Latn, fin_Latn, slv_Latn, vie_Latn, mkd_Cyrl, slk_Latn, nor_Latn, est_Latn, ltz_Latn, eus_Latn, lit_Latn, kaz_Cyrl, lav_Latn, bos_Latn, epo_Latn, cat_Latn, tha_Thai, ukr_Cyrl, tgl_Latn, sin_Sinh, gle_Latn, hin_Deva, kor_Hang, urd_Arab, swa_Latn, sqi_Latn, bel_Cyrl, afr_Latn, nno_Latn, tat_Cyrl, ast_Latn, mon_Cyrl, msa_Latn, som_Latn, srp_Cyrl, mlg_Latn, arz_Arab, tam_Taml, uzb_Latn, cos_Latn, als_Latn, amh_Ethi, sun_Latn, war_Latn, div_Thaa, yor_Latn, fao_Latn, bak_Cyrl, ilo_Latn, mri_Latn, asm_Beng, ibo_Latn, kin_Latn, hye_Armn, oci_Latn, lin_Latn, kir_Cyrl, nep_Deva, cym_Latn, lat_Latn, kat_Geor, fry_Latn, mya_Mymr, nds_Latn, pnb_Arab, ckb_Arab, chv_Cyrl, que_Latn, bre_Latn, pus_Arab, che_Cyrl, oss_Cyrl, nan_Latn, lim_Latn, tuk_Latn, min_Latn, khm_Khmr, jav_Latn, vec_Latn, jpn_Jpan, snd_Arab, yue_Hani, sco_Latn, ori_Orya, arg_Latn, kur_Latn, bar_Latn, roh_Latn, aym_Latn, sah_Cyrl, lmo_Latn, ido_Latn, vol_Latn, uig_Arab, bod_Tibt, pms_Latn, wuu_Hani, yid_Hebr, scn_Latn, ina_Latn, xmf_Geor, san_Deva, gla_Latn, mwl_Latn, diq_Latn, cbk_Latn, szl_Latn, hsb_Latn, vls_Latn, mhr_Cyrl, grn_Latn, lzh_Hani, mzn_Arab, nap_Latn, ace_Latn, frr_Latn, eml_Latn, vep_Latn, sgs_Latn, lij_Latn, crh_Latn, ksh_Latn, zea_Latn, csb_Latn, jbo_Latn, bih_Deva, ext_Latn, fur_Latn.

POS Benchmark Languages:

mal_Mlym, ben_Beng, tel_Telu, mlt_Latn, fra_Latn, spa_Latn, eng_Latn, rus_Cyrl, deu_Latn, tur_Latn, mar_Deva, por_Latn, nld_Latn, ara_Arab, zho_Hani, ita_Latn, ind_Latn, ell_Grek, bul_Cyrl, swe_Latn, ces_Latn, isl_Latn, pol_Latn, ron_Latn, dan_Latn, hun_Latn, srp_Latn, fas_Arab, ceb_Latn, heb_Hebr, hrv_Latn, glg_Latn, fin_Latn, slv_Latn, vie_Latn, slk_Latn, nor_Latn, est_Latn, eus_Latn, lit_Latn, kaz_Cyrl, lav_Latn, cat_Latn, tha_Thai, ukr_Cyrl, tgl_Latn, sin_Sinh, gle_Latn, hin_Deva, kor_Hang, urd_Arab, sqi_Latn, bel_Cyrl, afr_Latn, tat_Cyrl, tam_Taml, amh_Ethi, yor_Latn, fao_Latn, hye_Armn, cym_Latn, lat_Latn, nds_Latn, bre_Latn, hyw_Armn, jav_Latn, jpn_Jpan, yue_Hani, gsw_Latn, sah_Cyrl, uig_Arab, kmr_Latn, pcm_Latn, quc_Latn, san_Deva, gla_Latn, wol_Latn, sme_Latn, hsb_Latn, grc_Grek, hbo_Hebr, grn_Latn, lzh_Hani, ajp_Arab, nap_Latn, aln_Latn, glv_Latn, lij_Latn, myv_Cyrl, bam_Latn, xav_Latn.

C.2 For Benchmark Performances in Figure 2

SR-T Benchmark Languages:

tur_Latn, ell_Grek, bul_Cyrl, ces_Latn, kor_Hang, zsm_Latn, kat_Geor, fry_Latn, khm_Khmr, yue_Hani, tuk_Latn, uig_Arab, pam_Latn, kab_Latn, gla_Latn, mhr_Cyrl, swh_Latn, cmn_Hani, pes_Arab, dtp_Latn

SR-B Benchmark Languages:

tur_Latn, ell_Grek, bul_Cyrl, ces_Latn, kor_Hang, zsm_Latn, kat_Geor, fry_Latn, khm_Khmr, yue_Hani, tuk_Latn, uig_Arab, pam_Latn, kab_Latn, gla_Latn, mhr_Cyrl, swh_Latn, cmn_Hani, pes_Arab, dtp_Latn