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

Enes Özeren*, Yihong Liu*◇, Hinrich Schütze*◇

*LMU Munich ◇Munich Center for Machine Learning (MCML)
enes.oezeren@campus.lmu.de

Abstract

Many pre-trained language models (PLMs) exhibit suboptimal performance on mid- and low-resource languages, largely due to limited exposure to these languages during pre-training. A common strategy to address this is to introduce new tokens specific to the *target* languages, initialize their embeddings, and apply *continual pre-training* on target-language data. Among such methods, OFA (Liu et al., 2024a) proposes a similarity-based subword embedding initialization heuristic that is both effective and efficient. However, OFA restricts target-language token embeddings to be convex combinations of a fixed number of source-language embeddings, which may limit expressiveness. To overcome this limitation, we propose HYPEROFA, a hypernetwork-based approach for more adaptive token embedding initialization. The hypernetwork is trained to map from an *external multilingual word vector space* to the *PLM's token embedding space* using source-language tokens.[1] Once trained, it can generate flexible embeddings for target-language tokens, serving as a good starting point for continual pretraining. Experiments demonstrate that HYPEROFA consistently outperforms random initialization baseline and matches or exceeds the performance of OFA in both continual pre-training convergence and downstream task performance. We make the code publicly available.[2]

1 Introduction

Multilingual PLMs, trained on massive multilingual corpora, have achieved impressive performance across many high-resource languages (Devlin et al., 2019; Artetxe et al., 2020; Liang et al., 2023; Üstün et al., 2024). However, such models often perform suboptimally on languages that are under-resourced in their pre-training data (Wu and Dredze, 2020), and in extreme cases, they perform poorly on entirely unseen languages (Adelani et al., 2024), particularly when there is minimal lexical overlap or shared vocabulary between these unseen languages and the languages covered by the PLM (Muller et al., 2021; Moosa et al., 2023; Liu et al., 2024b; Xhelili et al., 2024).

A common strategy for adapting PLMs to such under-resourced or unseen languages is to introduce new, language-specific tokens, initialize their embeddings, and continually pre-train the model on data from the target languages (Tran, 2020).[3] A key challenge in this process lies in the initialization of these new token embeddings. A naive approach would be random initialization from a given simple distribution, e.g., multivariate Gaussian, (Hewitt, 2021; de Vries and Nissim, 2021; Marchisio et al., 2023). However, such an initialization fails to leverage any lexical or semantical knowledge captured by the original source-language embeddings.

To address this, recent work has explored more informed initialization strategies, using similarity-based heuristics to better align the initialized target embeddings with the existing embedding space, thereby enhancing language adaptation and accelerating continual pre-training (Minixhofer et al., 2022; Dobler and de Melo, 2023; Liu et al., 2024a; Mundra et al., 2024; Yamaguchi et al., 2024a,b). Among this line of work, for example, OFA (Liu et al., 2024a) uses external multilingual word vectors to compute similarities between source and target tokens, then initializes target embeddings as convex combinations of source embeddings, weighted by these similarities. This approach ensures the target embeddings reside in the same vector space as the source ones. However, the

[1] We will use *vector space* and *embedding space* to refer to the two different spaces for convenience.

[2] https://github.com/enesozeren/hyper-ofa

[3] We simply use *source tokens* to refer to tokens belonging to the source languages that are already covered in the PLM vocabulary. Similarly, *target tokens* is used to refer to tokens that belong to the target languages that one wants to adapt to and are usually not covered by the PLM vocabulary.

similarity-based convex combination restricts the relation between embeddings of source tokens and target tokens to be linear, which might not be expressive enough considering the non-linearity of Transformer (Vaswani et al., 2017).

To overcome this limitation, this paper presents HYPEROFA, a hypernetwork-based initialization method designed to enhance the expressiveness and adaptability of embedding initialization. Rather than depending on similarity heuristics, we explicitly learn a mapping from an external vector space to the PLM's embedding space via a hypernetwork. The hypernetwork is trained to predict the embedding of a source token, given external multilingual word vectors of a small set of related words as input. Training proceeds by minimizing the discrepancy between the predicted and actual PLM embeddings of source tokens. Once trained, the hypernetwork is used to generate embeddings for target tokens, providing a robust initialization for continual pre-training on the target languages.

To evaluate HYPEROFA, we follow the experimental setup of OFA, adapting both a monolingual PLM, i.e., RoBERTa (Liu et al., 2019), and a multilingual PLM, i.e., XLM-R (Conneau et al., 2020), to 22 languages covering high-, mid-, and low-resource scenarios. We investigate two research questions: (**1**) *How well do the initialized embeddings perform on their own?* and (**2**) *How effective are they as a starting point for continual pre-training?* To answer these, we evaluate models before and after continual pre-training via zero-shot cross-lingual transfer on downstream tasks, including sentence retrieval and sequence labeling. Our empirical results show that HYPEROFA consistently outperforms the random initialization and achieves competitive or superior performance compared to OFA. Our contributions are as follows:

- We propose HYPEROFA, a hypernetwork-based method for initializing embeddings of new tokens in target languages.
- We extensively evaluate HYPEROFA on adapting RoBERTa and XLM-R to many languages and various downstream tasks.
- We show that HYPEROFA outperforms random initialization and matches or exceeds the performance of its counterpart OFA.

2 Related Work

Tokenizer and Vocabulary Manipulation Manipulating an existing PLM's vocabulary and its accompanying tokenizer is a common approach for adapting it to new languages (Pfeiffer et al., 2021; Alabi et al., 2022; Zeng et al., 2023; Cui et al., 2024) or new domains (Lamproudis et al., 2022; Liu et al., 2023a; Balde et al., 2024). Typically, another tokenizer is trained on the target data using the same tokenization algorithm as used by the original one, such as Byte-Pair Encoding (Gage, 1994; Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012; Wu et al., 2016), and SentencePiece (Kudo and Richardson, 2018; Kudo, 2018). Then, the new tokenizer is merged with the original tokenizer, where unseen tokens are added, resulting in a large vocabulary. Imani et al. (2023) successfully apply such a pipeline to extend the language coverage of XLM-R (Conneau et al., 2020) to more than 500 languages. Similarly, Liu et al. (2025) adapts XLM-R to transliterated data by merging romanized subwords into the vocabulary.

Target Embedding Initialization The embeddings for the new tokens have to be initialized before the model can be used or continually pre-trained. The simplest approach is to randomly initialize the new token embeddings (Artetxe et al., 2020; de Vries and Nissim, 2021; Alabi et al., 2022; Imani et al., 2023). To better leverage the already encoded knowledge in the PLM, some work tries to initialize the new target token embeddings as linear combinations of embeddings of the source tokens, weighted by similarities between target and source tokens. An early work, Tran (2020), induces such similarities from a parallel corpus. More recently, another line of work explores the possibility of directly inducing such similarities from well-aligned external word embeddings (Minixhofer et al., 2022; Dobler and de Melo, 2023; Liu et al., 2024a; Yamaguchi et al., 2024a,b; Ye et al., 2024). However, the similarity-based convex combination might restrict the expressiveness of the new token embeddings. Therefore, this work aims to improve the initialization by breaking the linearity obstacle.

Hypernetworks Hypernetworks are neural networks designed to generate the weights of another network (Ha et al., 2017; Chauhan et al., 2024). A recent survey by Chauhan et al. (2024) highlights their application across various domains such as computer vision (von Oswald et al., 2020) and

natural language processing (NLP) (Volk et al., 2023; Pinter et al., 2017; Schick and Schütze, 2020; Minixhofer et al., 2024). One of the earlier works in initializing embeddings with hypernetworks is MIMICK (Pinter et al., 2017), which focuses on predicting the out-of-vocabulary word embeddings with a hypernetwork. Similarly, Schick and Schütze (2020) integrates a hypernetwork into BERT (Devlin et al., 2019) 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. Our work builds upon the insights from this line of work and designs a hypernetwork to map from the external word vector space to the PLM's embedding space, allowing for wise initialization of the new token embeddings for effective continual pre-training.

3 Methodology

HYPEROFA builds upon certain aspects of OFA (Liu et al., 2024a), e.g., factorized parameterization (cf. §3.2) and external multilingual vector vectors (cf. §3.3). The key differentiator is that we directly predict the target token embeddings using a hypernetwork (cf. §3.4) instead of initialization based on similarity-heuristics. For a clearer understanding, we therefore follow the notations used by Liu et al. (2024a) and introduce HYPEROFA in the following. Figure 1 provides an overview of HYPEROFA.

3.1 Problem Setting

Given a model with a source tokenizer TOK^s with vocabulary V^s, the goal is to replace the source tokenizer with a target tokenizer TOK^t with vocabulary V^t that supports a broader range of tokens across various languages. Typically, $|V^s| < |V^t|$. The core problem is to **initialize the target embeddings** $\boldsymbol{E}^t \in \mathbb{R}^{|V^t| \times D}$, where D is the embedding dimension, which is the same as the dimension of the source embeddings $\boldsymbol{E}^s \in \mathbb{R}^{|V^s| \times D}$.

3.2 Source Embedding Factorization

Since $|V^t| > |V^s|$, the number of embedding parameters grows significantly from $V^s \times D$ to $V^t \times D$ in the target model. This can result in a large ratio of model parameters in the embedding matrix, limiting the efficiency. To address this, Liu et al. (2024a) adopts a factorized parametrization to represent the embeddings, similar to Lan et al. (2020).

Factorization decomposes the $\boldsymbol{E}^s$ into two smaller matrices using the Singular Value Decomposition (SVD) method, such that $\boldsymbol{E}^s \approx \boldsymbol{F}^s \boldsymbol{P}$, where $\boldsymbol{F}^s \in \mathbb{R}^{|V^s| \times D'}$ is the coordinate matrix containing token-specific parameters, and $\boldsymbol{P} \in \mathbb{R}^{D' \times D}$ is the primitive embedding matrix capturing language-agnostic features. When $D' < D$, the total number of parameters of $\boldsymbol{F}^s$ and $\boldsymbol{P}$ is smaller than $\boldsymbol{E}^s$. Since $\boldsymbol{P}$ is expected to be shared across languages, one only needs to initialize the coordinate matrix $\boldsymbol{F}^t \in \mathbb{R}^{|V^t| \times D'}$ for TOK^t while reusing the same $\boldsymbol{P}$. The original dimension can be restored by multiplication: $\boldsymbol{F}^t P \in \mathbb{R}^{|V^t| \times D}$.

3.3 Matching External Word Vectors

OFA (Liu et al., 2024a) takes advantage of external well-aligned multilingual vectors W to induce the similarities between source tokens and target tokens.[4] In contrast, we directly use these vectors to train a hypernetwork to map from the vector space to the embedding space, discarding the similarity-based heuristics. To do this, we first need to create corresponding pairs of tokens in $V^s \cup V^t$ and words in W, which is achieved by a matching operation. Specifically, a token in $V^s \cup V^t$ is matched with a word in W if that word contains the token as a subword (cf. Figure 1). This matching operation results in s_i (resp. t_j), a set of matched words for each token i in V^s (resp. each token j in V^t). We then represent the set of matched word vectors for each token i (resp. j) as $W_{\{s_i\}}$ (resp. $W_{\{t_j\}}$).

3.4 Hypernetwork

To address the main limitation of OFA– use a convex combination of source-token embeddings to initialize the target embeddings – we propose a hypernetwork approach to directly map from the vector space to the embedding space, which introduces non-linearity, and thus is more expressive.

After performing factorization (cf. §3.2) and creating the set of matched words and tokens (cf. §3.3), a hypernetwork HN_θ with parameters θ is introduced. The ultimate aim of the hypernetwork is to generate the target-token embedding $\boldsymbol{F}_j$ by using the matched word vectors $W_{\{t_j\}}$, where $j \in V^t$. Therefore, we need to properly train HN_θ so that it can map from the vector space to the embedding space. To do this, we create a training set for HN_θ. Each item in the training set is a pair: $(W_{\{s_i\}}, \boldsymbol{F}_i^s)$, where $W_{\{s_i\}}$ and $\boldsymbol{F}_i^s$ are the set of

[4] Liu et al. (2024a) use $\overrightarrow{\text{ColexNet}}$+ (Liu et al., 2023b), which are static word vectors that contain over 4M words spanning more than 1K languages. The tokens in V^t are usually subwords of the word types covered by $\overrightarrow{\text{ColexNet}}$+.

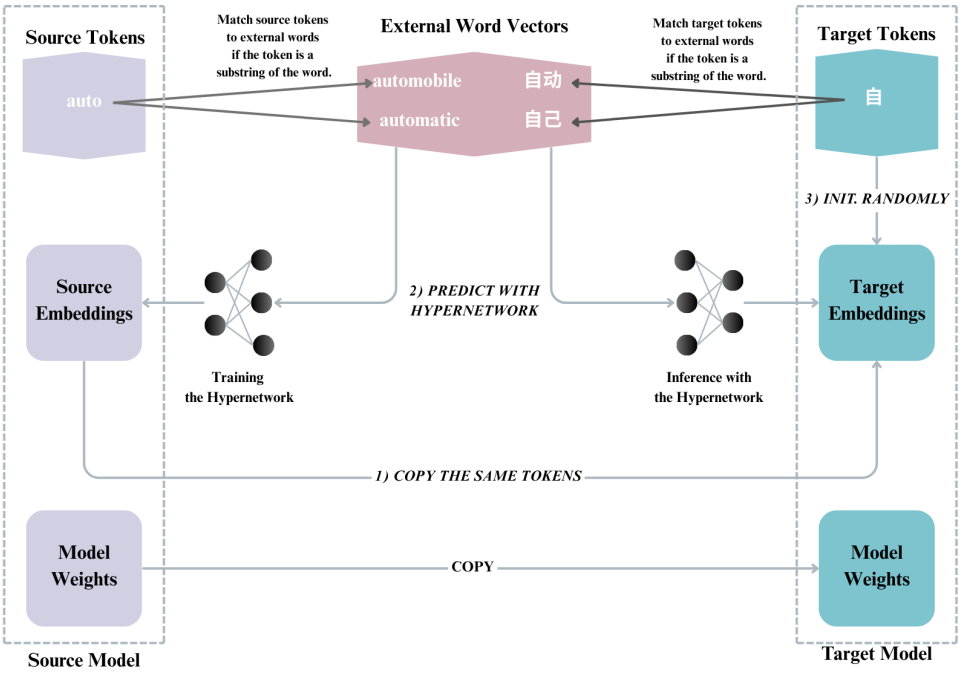

Figure 1: HYPEROFA pipeline. The source model (left) transfers weights to the target model (right). The target embeddings are initialized by first copying embeddings for matching tokens, then generating embeddings via a hypernetwork for tokens with matching external words, and finally randomly initializing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

matched word vectors and coordinate vector in $\boldsymbol{F}^s$ for token i in V^s, respectively.[5] HN_θ then takes $W_{\{s_i\}}$ as input and is trained to predict $\boldsymbol{F}^s_i$.

A custom loss function is proposed for the training, which contains two training objectives: a batch-wise *contrastive loss* $\mathcal{L}_\text{c}$ and a *normalized L1 loss* $\mathcal{L}_\text{L1}$. The contrastive loss $\mathcal{L}_\text{c}$ aims to improve the similarity between the ground-truth coordinate embeddings and the predictions:

$$\mathcal{L}_\text{c} = \mathbb{E}\left[-\log \frac{\exp(\text{sim}(\boldsymbol{F}^s_i, \hat{\boldsymbol{F}}^s_i)/\tau)}{\exp(\text{sim}(\boldsymbol{F}^s_i, \hat{\boldsymbol{F}}^s_i)/\tau) + \text{NEG}}\right]$$

where $\text{NEG} = \sum_{k \neq i} \exp(\text{sim}(\boldsymbol{F}^s_k, \hat{\boldsymbol{F}}^s_i))/\tau)$, sim is cosine similarity, $\hat{\boldsymbol{F}}^s_i = HN_\theta(W_{\{s_i\}})$ and τ is temperature. The normalized L1 loss $\mathcal{L}_\text{L1}$ aims to preserve magnitude consistency:

$$\mathcal{L}_\text{L1} = \mathbb{E}\left[\|\boldsymbol{F}^s_i - \hat{\boldsymbol{F}}^s_i\|_1\right]$$

The final loss is $\mathcal{L}(\theta) = \lambda \cdot \mathcal{L}_\text{c} + (1-\lambda) \cdot \mathcal{L}_\text{L1}$ where λ is a hyperparameter controlling the weight.

When designing the model architecture for HN_θ, there are certain requirements because of the input – a set of vectors. First, the number of matched word vectors may vary for different tokens, meaning the model 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 should be permutation-invariant. Considering these requirements, we used a BiLSTM (Schuster and Paliwal, 1997) for HN_θ despite it not inherently satisfying the permutation-invariance requirement.[6] To address the BiLSTM's sensitivity to input order, data augmentation is implemented by randomly shuffling the order of the word vectors during each training epoch, effectively preventing the model from overfitting to specific sequence arrangements.

3.5 New Token Initialization

The target coordinate embeddings, $\boldsymbol{F}^t$, are initialized in three steps similar to OFA (Liu et al., 2024a) (cf. Figure 1).

1. For tokens in $V^s \cap V^t$, their embeddings in $\boldsymbol{F}^s$ are directly copied to $\boldsymbol{F}^t$.

2. For tokens that have at least one matched word (cf. §3.3), their embeddings are predicted by HN_θ using the set of vectors $W_{\{t_j\}}$ as input.

[5] We exclude $(W_{\{s_i\}}, \boldsymbol{F}^s_i)$ from the training set if $s_i = \emptyset$, i.e., there are no matched words for the concerned token i.

[6] We experimented with both Transformer and BiLSTM architectures for the hypernetwork, but experiments have shown that BiLSTM works better in our study (cf. Appendix §A.1)

3. For the remaining tokens, their embeddings are randomly initialized from a normal distribution $\mathcal{N}(\mathbb{E}[F^s], \text{Var}[F^s])$, similar to OFA.

4 Experimental Setup

4.1 HYPEROFA-Based Models

Following OFA (Liu et al., 2024a), we use the tokenizer of Glot500-m (Imani et al., 2023) as the target tokenizer, which is trained by SentencePiece (Kudo and Richardson, 2018; Kudo, 2018) and has a vocabulary size of 401K. We consider three different dimensions for D': 100, 200, 400 (cf. §3.2). We create 6 models using HYPEROFA as follows:

HYPEROFA-mono-xxx These are RoBERTa models (Liu et al., 2019) with an extended vocabulary (from the original 50K to 401K). "xxx" denotes the embedding dimension of the model (100, 200, 400), and the "mono" suffix indicates that the model is originally monolingual. The new token embeddings are predicted by a hypernetwork trained specifically for each model (cf. §4.2) or randomly initialized as a fallback (cf. §3.5).

HYPEROFA-multi-xxx These are XLM-R models (Conneau et al., 2020) with an extended vocabulary (from the original 250K to 401K). "xxx" denotes the embedding dimension of the model (100, 200, 400), and the "multi" suffix indicates that the model is originally multilingual. The new token embeddings are predicted by a hypernetwork trained specifically for each model (cf. §4.2) or randomly initialized as a fallback (cf. §3.5).

4.2 Hypernetwork Setup

Hypernetwork Training Dataset For HYPEROFA-mono-xxx models, the hypernetwork training dataset consists of **22K pairs** of embeddings of the source tokens and their corresponding sets of matched word vectors, as 22K out of RoBERTa's 50K vocabulary tokens match at least one word in $\overrightarrow{\text{ColexNet}}$+ (cf. §3.4). Similarly, for XLM-R, the training dataset contains **103K pairs**, corresponding to 103K tokens from its 250K vocabulary.

Hypernetwork Training As described in §3.4, we use a BiLSTM architecture for hypernetworks. The hyperparameters of training are explained in the §A.2. Table 1 shows the hypernetwork parameter sizes used for each HYPEROFA-based model. Notably, the hypernetworks have a substantial number of parameters compared to their corresponding models. Preliminary experiments show that larger hypernetworks, when combined with strong regularization (dropout and the data augmentation methods), perform better than smaller hypernetworks. Figure 2 shows a case comparison study, which compares two hypernetworks for HYPEROFA-multi-400 model, one with 210M and one with 8M parameters. During training of the two hypernetworks, the larger one predicts embeddings better than the smaller one, when measuring cosine similarities to the true token embeddings in the validation set. Also, as the dimension of the predicted embedding increases, a hypernetwork with higher capacity is necessary. Therefore, the hidden dimension of the BiLSTM is increased for embeddings with higher dimensions (see Appendix Table 6).

LM	Param	Hypernetwork	Param
HYPEROFA-mono-100	92M	HN-R-100	22M
HYPEROFA-mono-200	97M	HN-R-200	23M
HYPEROFA-mono-400	107M	HN-R-400	87M
HYPEROFA-multi-100	113M	HN-X-100	53M
HYPEROFA-multi-100	138M	HN-X-200	54M
HYPEROFA-multi-400	188M	HN-X-400	210M

Table 1: Number of parameters in HYPEROFA-based models and their associated hypernetworks.

4.3 Baselines

We consider the following baselines for comparison with HYPEROFA. The details of how many tokens are randomly initialized or wisely initialized in each model are shown in Table 2.

OFA-mono-xxx RoBERTa models (Liu et al., 2019) with an extended vocabulary (from the original 50K to 401K) where the new token embeddings are initialized with OFA (Liu et al., 2024a).

OFA-multi-xxx XLM-R models (Conneau et al., 2020) with an extended vocabulary (from the original 250K to 401K) where the new token embeddings are initialized with OFA (Liu et al., 2024a).

Random-mono-xxx RoBERTa models (Liu et al., 2019) with an extended vocabulary (from the original 50K to 401K). Embeddings of all overlapping tokens are directly copied, while embeddings of the remaining tokens are randomly initialized from a Gaussian distribution with mean and standard deviations of the source embeddings.

Random-multi-xxx XLM-R models (Conneau et al., 2020) with an extended vocabulary (from the

original 50K to 401K). Embeddings of all overlapping tokens are directly copied, while embeddings of the remaining tokens are randomly initialized from a Gaussian distribution with mean and standard deviations of the source embeddings.

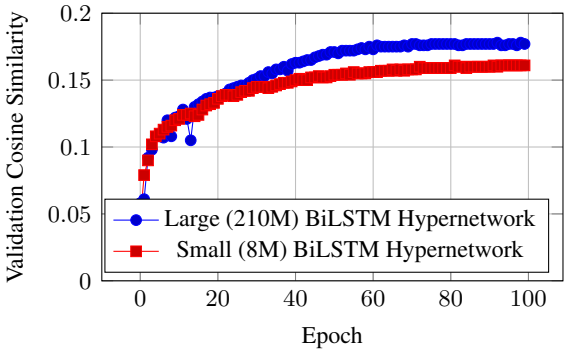

Figure 2: Comparison of large (210M parameters) and small (8M parameters) BiLSTM-based hypernetworks (HN-X-400) in terms of validation cosine similarity between predicted and true embeddings over 100 epochs for creating the HYPEROFA-multi-400 model.

Method	Model	Wise	Random	Total
HYPEROFA	RoBERTa	179K	195K	401K
	XLM-R	84K	62K	401K
OFA	RoBERTa	179K	195K	401K
	XLM-R	84K	62K	401K
Random	RoBERTa	0	374K	401K
	XLM-R	0	146K	401K

Table 2: Distribution of token embeddings initialized using HYPEROFA, OFA, and random initialization methods. The "Wise" column indicates the number of tokens initialized using the respective wise initialization method. The "Random" column indicates tokens initialized randomly. The difference between the total tokens ("Total") and the sum of "Wise" and "Random" columns represents token embeddings directly copied from the source embedding matrix due to vocabulary overlapping. This distribution holds consistently across all variants with different embedding factorization dimensions (100, 200, 400). Many token embeddings in HYPEROFA and OFA are wisely initialized.

4.4 Downstream Tasks

The performances of HYPEROFA-based models and the baselines are evaluated by four datasets in two downstream tasks: sentence retrieval and two sequence labeling, introduced as follows.

Sentence Retrieval Retrieval performance is assessed using the Sentence Retrieval Tatoeba (SR-T) (Artetxe and Schwenk, 2019) and Sentence Retrieval Bible (SR-B) datasets. Following Liu et al. (2024a), Top-10 accuracy is used as the evaluation metric, where the correct translation must be among the ten nearest neighbors of a query English sentence. Sentence-level 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 WikiANN (Pan et al., 2017) and Universal Dependencies (de Marneffe et al., 2021) datasets, respectively. Our evaluation methodology follows Liu et al. (2024a), 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 cross-lingual transfer performance on test sets in other languages. F1 scores are reported for both datasets.

5 Results

To validate the effectiveness of HYPEROFA, we evaluate HYPEROFA-based models and baselines in two scenarios: **before** (cf. §5.1) and **after** (cf. §5.2) the continual pre-training.

5.1 Before Continual Pre-Training

This evaluation aims to directly reflect the quality of the embeddings initialized with HYPEROFA. Since the newly added tokens cover more than 500 languages (we use the Glot500-m tokenizer as the target tokenizer), we evaluate HYPEROFA-based models and baselines on **all** languages in downstream tasks. The results are presented in Table 3.

HYPEROFA and OFA consistently outperform the random baselines, while showing comparable performance to each other across downstream tasks. In all downstream tasks, the models with randomly initialized new embeddings perform the worst. This indicates that randomly initializing the new token embeddings is suboptimal as no encoded knowledge in the original embedding matrix is explicitly leveraged. For the retrieval tasks (SR-B and SR-T), HYPEROFA performs better than OFA on all cases except when the embedding dimension is 400 in the mono setup. We hypothesize this might be because, with a fixed amount of training data (22K pairs for mono models), learning higher-dimensional embeddings becomes more challenging for the hypernetwork. This hypothesis is supported by the fact that when more training instances are included in the multi models (103

Models	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**	24.9	22.8
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**	**10.6**	38.3	48.3
Random-multi-400	5.6	21.0	41.6	53.7
OFA-multi-400	5.9	**21.3**	43.3	**54.6**
HYPEROFA-multi-400	**6.1**	**21.3**	**43.5**	54.1

Table 3: Performance of randomly initialized baselines, OFA and HYPEROFA before continual pre-training. The scores for OFA models are taken from Liu et al. (2024a) directly. SR-B covers **98** languages, SR-T covers **369** languages, NER covers **164** languages, and POS covers **91** languages. Top-10 accuracy is reported for SR-B and SR-T; F1 score is reported for NER and POS. All metrics are average across languages.

pairs), HYPEROFA-mutli-400 models achieve comparable or even better results than OFA-multi-400 models across all downstream tasks.

5.2 After Continual Pre-Training

Continual pre-training is crucial because, even with carefully initialized new token embeddings, the embeddings and the backbone model must be fine-tuned on data containing these new tokens. Therefore, to validate how effective the new embeddings with HYPEROFA are as a starting point for continual pre-training, we select 6 models and continually pre-train them on a diverse set of languages.

Models and Training Due to resource constraints, we select **6** models out of 18 models for continual pre-training. For the mono models, we use Random-mono-100, OFA-mono-100, and HYPEROFA-mono-100; for the multi models, we use Random-multi-400, OFA-multi-400, and HYPEROFA-multi-400. All six models are continually pre-trained using hyperparameters similar to those in Liu et al. (2024a), with some key differences: an effective batch size of 512 instead of 384 and training on 4 NVIDIA H100 GPUs. The training is conducted for 4,000 steps (approx. 1 epoch).

Model	Phase	SRT	SRB	POS	NER
HyperOFA-mono-100	Before	6.0	5.1	30.0	23.5
	After	11.3	**9.9**	56.3	43.4
OFA-mono-100	Before	5.9	5.0	30.2	24.0
	After	**15.2**	9.8	**56.8**	**45.7**
Random-mono-100	Before	4.4	3.6	29.1	23.3
	After	9.5	7.0	51.1	40.0
HyperOFA-multi-400	Before	17.7	9.2	63.7	47.5
	After	**56.1**	42.2	**70.4**	**60.5**
OFA-multi-400	Before	17.9	8.6	62.9	47.2
	After	55.8	**42.3**	**70.4**	60.3
Random-multi-400	Before	17.6	8.1	65.0	45.9
	After	55.3	40.8	70.3	59.8

Table 4: Performance before and after continual pre-training. Evaluation is conducted on the intersection of the 22 continual pre-training languages and those available in each downstream task. Specifically, SR-T and SR-B are evaluated on **20** languages, POS on **9** languages, and NER on **14** languages. Metrics reported are: Top-10 accuracy for SR-T and SR-B, F1 score for POS NER. All metrics are averaged across the respective languages. HYPEROFA achieves consistently better performance than the random baseline and competitive performance compared with OFA.

Training Data Due to constrained computing resources, we are not able to continually train HYPEROFA-based models or other baselines on full Glot500-c (Imani et al., 2023). Therefore, a subset of languages from Glot500-c comprising **22** languages spanning high, mid, and low-resource categories is used for the continual pre-training. The list of languages and their data size can be found in Appendix Table 7. This dataset subset contains 1.1 billion tokens across 36 million sentences.

The benchmark results for before and after continual pre-training for the 6 models are presented in Table 4. The metrics are calculated for the languages that are in the 22 continual pre-training languages. And the training loss curves of the 6 models throughout the continual pre-training are presented in Figure 3.

Multilingual XLM-R models consistently outperform their monolingual RoBERTa counterparts, highlighting the advantages of multilingual pre-training. The first observation is that all models based on XLM-R outperform the

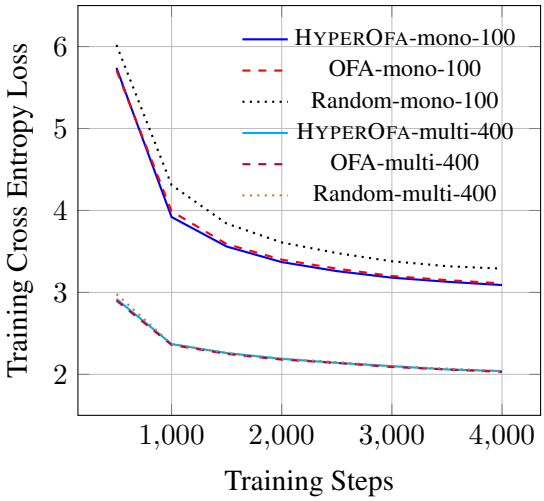

Figure 3: Training loss curves during the continual pre-training of models initialized with HYPEROFA, OFA, or random initialization methods.

RoBERTa-based models. This aligns with our expectations, as XLM-R already sees much multilingual data during its pre-training stage, which helps further adapt to other languages. In contrast, RoBERTa is originally monolingual and therefore lacks enough multilingual knowledge.

Within XLM-R models, the choice of embedding initialization has minimal impact, suggesting inherent robustness to vocabulary extension. Different initialization (random, OFA, or HYPEROFA) methods do not produce substantial performance differences in models based on XLM-R across downstream tasks. The loss curves (cf. Figure 3) also show that different multilingual models show a similar convergence trend throughout continual pre-training progression. This suggests that multilingual models are already quite robust and effective in adapting to new languages even when new token embeddings are randomly initialized.

RoBERTa-based models benefit from wise initialization methods. Models with embeddings initialized using OFA and HYPEROFA show notably improved performance compared to those with the random baseline in RoBERTa-based models across all downstream tasks. Additionally, OFA and HYPEROFA also show faster convergence (at the same training step but a lower loss) than the random baseline, as shown in Figure 3. This highlights the significance of advanced embedding initialization techniques for monolingual models – a better strategy can actively leverage the knowledge encoded in the original embeddings, though monolingual, and can be transferred to other languages.

HYPEROFA and OFA perform comparably across downstream tasks, suggesting both are viable strategies. We observe that HYPEROFA achieves comparable or occasionally better results than OFA. However, the difference is generally small, with neither method showing a decisive advantage overall. This suggests that both approaches are effective, with their relative strengths depending on the specific evaluation metric. However, because of the capability of modeling non-linearity, we expect HYPEROFA-based models can improve when more training data (for hypernetworks and continual pre-training) is available.

6 Conclusion

This study introduces HYPEROFA, a method for expanding the vocabulary of PLMs to new languages and initializing new token embeddings with a hypernetwork. We show the effectiveness of HYPEROFA by evaluating the resulting models both before and after the continual pre-training. The results show that HYPEROFA consistently outperforms the random initialization baseline and performs competitively with OFA. These results highlight HYPEROFA as a promising approach, alongside OFA, for efficient new token embedding initialization towards effective and efficient continual pre-training.

Limitations

This study explores 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), the 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 §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 continual pre-trained dataset used in this study is relatively small compared to that of Liu et al. (2024a) due to computational constraints. Exploring the impact of larger datasets, especially those having more languages, could provide deeper

insights into the strengths and weaknesses of the proposed methods in different settings.

Acknowledgements

We sincerely thank Mina Rezaei for insightful discussions. We also gratefully acknowledge the Leibniz Supercomputing Centre (LRZ) of the Bavarian Academy of Sciences and Humanities and the Munich Center for Machine Learning (MCML) for generously providing computational resources.

References

David Ifeoluwa Adelani, Hannah Liu, Xiaoyu Shen, Nikita Vassilyev, Jesujoba O. Alabi, Yanke Mao, Haonan Gao, and En-Shiun Annie Lee. 2024. SIB-200: A simple, inclusive, and big evaluation dataset for topic classification in 200+ languages and dialects. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 226–245, St. Julian's, Malta. Association for Computational Linguistics.

Jesujoba O. Alabi, David Ifeoluwa Adelani, Marius Mosbach, and Dietrich Klakow. 2022. Adapting pre-trained language models to African languages via multilingual adaptive fine-tuning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4336–4349, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.

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

Gunjan Balde, Soumyadeep Roy, Mainack Mondal, and Niloy Ganguly. 2024. MEDVOC: vocabulary adaptation for fine-tuning pre-trained language models on medical text summarization. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 6180–6188. ijcai.org.

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

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Yiming Cui, Ziqing Yang, and Xin Yao. 2024. Efficient and effective text encoding for chinese llama and alpaca. *Preprint*, arXiv:2304.08177.

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

Wietse de Vries and Malvina Nissim. 2021. As good as new. how to successfully recycle English GPT-2 to make models for other languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 836–846, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. 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.

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

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

John Hewitt. 2021. Initializing new word embeddings for pretrained language models.

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. 2023. Glot500: Scaling multilingual corpora and language models to 500 languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1082–1117, Toronto, Canada. Association for Computational Linguistics.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Anastasios Lamproudis, Aron Henriksson, and Hercules Dalianis. 2022. Vocabulary modifications for domain-adaptive pretraining of clinical language models. In *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2022, Volume 5: HEALTHINF, Online Streaming, February 9-11, 2022*, pages 180–188. SCITEPRESS.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Davis Liang, Hila Gonen, Yuning Mao, Rui Hou, Naman Goyal, Marjan Ghazvininejad, Luke Zettlemoyer, and Madian Khabsa. 2023. XLM-V: Overcoming the vocabulary bottleneck in multilingual masked language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13142–13152, Singapore. Association for Computational Linguistics.

Siyang Liu, Naihao Deng, Sahand Sabour, Yilin Jia, Minlie Huang, and Rada Mihalcea. 2023a. Task-adaptive tokenization: Enhancing long-form text generation efficacy in mental health and beyond. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15264–15281, Singapore. Association for Computational Linguistics.

Yihong Liu, Peiqin Lin, Mingyang Wang, and Hinrich Schuetze. 2024a. OFA: A framework of initializing unseen subword embeddings for efficient large-scale multilingual continued pretraining. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1067–1097, Mexico City, Mexico. Association for Computational Linguistics.

Yihong Liu, Chunlan Ma, Haotian Ye, and Hinrich Schuetze. 2024b. TransliCo: A contrastive learning framework to address the script barrier in multilingual pretrained language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2476–2499, Bangkok, Thailand. Association for Computational Linguistics.

Yihong Liu, Chunlan Ma, Haotian Ye, and Hinrich Schütze. 2025. TransMI: A framework to create strong baselines from multilingual pretrained language models for transliterated data. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 469–495, Abu Dhabi, UAE. Association for Computational Linguistics.

Yihong Liu, Haotian Ye, Leonie Weissweiler, Renhao Pei, and Hinrich Schuetze. 2023b. Crosslingual transfer learning for low-resource languages based on multilingual colexification graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8376–8401, Singapore. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Kelly Marchisio, Patrick Lewis, Yihong Chen, and Mikel Artetxe. 2023. Mini-model adaptation: Efficiently extending pretrained models to new languages via aligned shallow training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5474–5490, Toronto, Canada. Association for Computational Linguistics.

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.

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

Ibraheem Muhammad Moosa, Mahmud Elahi Akhter, and Ashfia Binte Habib. 2023. Does transliteration help multilingual language modeling? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 670–685, Dubrovnik, Croatia. Association for Computational Linguistics.

Benjamin Muller, Antonios Anastasopoulos, Benoît Sagot, and Djamé Seddah. 2021. When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 448–462, Online. Association for Computational Linguistics.

Nandini Mundra, Aditya Nanda Kishore Khandavally, Raj Dabre, Ratish Puduppully, Anoop Kunchukuttan,

and Mitesh M Khapra. 2024. An empirical comparison of vocabulary expansion and initialization approaches for language models. In *Proceedings of the 28th Conference on Computational Natural Language Learning*, pages 84–104, Miami, FL, USA. Association for Computational Linguistics.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. 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, Vancouver, Canada. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112, Copenhagen, Denmark. Association for Computational Linguistics.

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

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

Timo Schick and Hinrich Schütze. 2020. BERTRAM: Improved word embeddings have big impact on contextualized model performance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3996–4007, Online. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*, pages 5149–5152. IEEE.

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

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.

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

Ahmet Üstün, Viraat Aryabumi, Zheng Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. 2024. Aya model: An instruction finetuned open-access multilingual language model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15894–15939, Bangkok, Thailand. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Tomer Volk, Eyal Ben-David, Ohad Amosy, Gal Chechik, and Roi Reichart. 2023. Example-based hypernetworks for multi-source adaptation to unseen domains. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9096–9113, Singapore. Association for Computational Linguistics.

Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. 2020. Continual learning with hypernetworks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and 1 others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Orgest Xhelili, Yihong Liu, and Hinrich Schuetze. 2024. Breaking the script barrier in multilingual pretrained language models with transliteration-based post-training alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11283–11296, Miami, Florida, USA. Association for Computational Linguistics.

Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. 2024a. An empirical study on cross-lingual vocabulary adaptation for efficient language model inference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6760–6785, Miami, Florida, USA. Association for Computational Linguistics.

Atsuki Yamaguchi, Aline Villavicencio, and Nikolaos Aletras. 2024b. How can we effectively expand the vocabulary of llms with 0.01 gb of target language text? *arXiv preprint arXiv:2406.11477*.

Haotian Ye, Yihong Liu, Chunlan Ma, and Hinrich Schütze. 2024. MoSECroT: Model stitching with static word embeddings for crosslingual zero-shot transfer. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 1–7, Mexico City, Mexico. Association for Computational Linguistics.

Qingcheng Zeng, Lucas Garay, Peilin Zhou, Dading Chong, Yining Hua, Jiageng Wu, Yikang Pan, Han Zhou, Rob Voigt, and Jie Yang. 2023. Greenplm: Cross-lingual transfer of monolingual pre-trained language models at almost no cost. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 6290–6298. ijcai.org.

A Experiments for Hypernetwork

A.1 Architecture: BiLSTM vs Setformer

As explained in the §3.4, there are two requirements for the model architecture; variable length input, permutation invariant. To satisfy those requirements, initially, an encoder only transformer model (Vaswani et al., 2017) without positional encoding layers (called as Setformer in this study) was tested. However, after observing poor performance, the approach shifted to a BiLSTM (Bidirectional LSTM) 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 (Table 5).

Table 5 compares the two candidate hypernetwork architectures, Setformer and BiLSTM, for initializing token embeddings for HYPEROFA-mono-100 model. The model initialized with the BiLSTM hypernetwork achieves better SR-T Top 10 accuracy (6.4), outperforming the Setformer variant. This suggests that BiLSTM is more effective than Setformer as a hypernetwork.

We attribute the reason for the poor performance of the Setformer to the need of transformers that require a large amount of data to learn effectively. On the other hand, the BiLSTM architecture was more efficient at learning the task with the available data which is limited by the source vocabulary size.

LM	Hypernetwork	SR-T
HYPEROFA-mono-100	BiLSTM	**6.4**
HYPEROFA-mono-100	Setformer	5.2
Random-mono-100	-	4.6

Table 5: Comparison of Setformer (Transformer encoder without positional encodings) and BiLSTM as hypernetworks both having 22M trainable parameters. They are used for initializing token embeddings in HYPEROFA-mono-100, a RoBERTa-based model with a new vocabulary and factorized embedding dimension of 100 (mono-100). The SR-T Top 10 Accuracy is reported for the without continual pre-training set up. Random initialization baseline performance is given at the last row. BiLSTM performs better as a hypernetwork.

A.2 Hyperparameters

The hypernetworks follow a BiLSTM architecture. All hypernetworks for HYPEROFA-mono-xxx and HYPEROFA-multi-xxx models 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 a healthy training, the hyperparameters in the loss function, as explained in §3.4, were set as follows: $\lambda = 0.1$ for all hypernetworks, and $T = 0.5$ for the hypernetworks of HYPEROFA-mono-xxx, and $T = 0.25$ for the hypernetworks of HYPEROFA-multi-xxx.

All models were trained until the validation loss converged. More details about the training data, model parameter sizes are presented in Table 6.

A.3 Regularization

We applied multiple regularization and data augmentation methods to ensure that hypernetworks do not overfit.

We used high dropout rate of 0.4 since we have seen that the large models with high regularization performs better (see Figure 2). We also applied data augmentation by shuffling word vector order before each epoch to prevent model to overfit to the order of the input word vectors.

Additionally, with 50% probability, the number of word vectors is randomly limited to 50–100% of the available vectors.

LM	Hypernetwork	Training Data	Layers	Hid Dim	Param	Epoch
HYPEROFA-mono-100	HN-R-100	22K	2	800	22M	370
HYPEROFA-mono-200	HN-R-200	22K	2	800	23M	470
HYPEROFA-mono-400	HN-R-400	22K	2	1600	87M	400
HYPEROFA-multi-100	HN-X-100	103K	4	800	53M	120
HYPEROFA-multi-200	HN-X-200	103K	4	800	54M	230
HYPEROFA-multi-400	HN-X-400	103K	4	1600	210M	80

Table 6: Hypernetwork model details for predicting the target embeddings for HYPEROFA-mono-xxx and HYPEROFA-multi-xxx language models with different factorized dimensions. All hypernetworks have the BiLSTM architecture. Epochs column indicated the converged epoch number for the hypernetwork.

B Continual Pre-training Dataset

The continual pre-training dataset was deliberately kept smaller than that used by Liu et al. (2024a) 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 7. For continual pre-training 36M sentences (approx. 1.1B tokens) across 22 languages are used. To categorize source category with respect to the volume of that language in Glot500-c, thresholds used: high (>5M sentences), mid (>500K sentences), and low (<500K sentences).

C Benchmark Language Coverage

In this section, we present the languages used in benchmarks for the tables in our paper.

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,

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 7: Distribution of continued pre-trainig data. The table shows the original Glot500-c volume and sub-sampled volume for each language, grouped by their source category (High, Mid, Low) which is assigned with respect to the volume of that language in Glot500-c.

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 Table 4

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

NER Benchmark Languages:

eng_Latn, tur_Latn, ell_Grek, bul_Cyrl, ces_Latn, kor_Hang, kat_Geor, fry_Latn, khm_Khmr, yue_Hani, tuk_Latn, uig_Arab, gla_Latn, mhr_Cyrl

POS Benchmark Languages:

eng_Latn, tur_Latn, ell_Grek, bul_Cyrl, ces_Latn, kor_Hang, yue_Hani, uig_Arab, gla_Latn

D Performance - Language Breakdown

In this section we show the benchmark results per language before continual pre-training (checkpoint 0) and after (checkpoint 4000) for the 6 models which had continual pre-training (see §5.2).

SR-B for mono-100 Models

	Checkpoint 0			Checkpoint 4000		
	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100
eng_Latn	-	-	-	-	-	-
tur_Latn	5.2	5.6	6.4	8.2	**11.2**	9.4
ell_Grek	3.8	4.6	5.2	6.8	**13.0**	12.6
bul_Cyrl	4.8	5.6	3.8	13.8	**29.2**	28.8
ces_Latn	4.6	7.2	6.4	18.0	17.2	**23.0**
kor_Hang	3.6	6.0	6.4	7.8	10.6	**12.0**
kat_Geor	2.8	4.4	4.6	7.0	8.6	**10.2**
fry_Latn	3.6	5.6	7.2	14.6	**16.8**	15.4
zsm_Latn	3.8	6.6	6.6	11.8	**23.4**	20.2
khm_Khmr	2.8	5.8	4.6	3.8	6.2	**8.0**
jpn_Japn	-	-	-	-	-	-
yue_Hani	1.8	2.4	2.8	4.2	**5.8**	**5.8**
tuk_Latn	4.2	4.8	**6.8**	5.4	6.4	6.0
uig_Arab	2.2	3.2	3.2	**4.0**	3.8	**4.0**
pam_Latn	4.2	5.4	5.6	5.2	6.0	**6.4**
kab_Latn	2.8	2.4	3.6	3.8	**5.2**	4.2
gla_Latn	2.8	3.8	**4.8**	4.4	4.4	4.4
mhr_Cyrl	3.6	6.8	**7.0**	4.2	6.8	6.6
swh_Latn	3.4	**5.0**	**5.0**	3.8	4.8	3.6
cmn_Hani	5.8	5.2	3.8	5.0	**9.0**	8.0
pes_Arab	4.8	**7.0**	6.4	2.8	3.6	4.0
dtp_Latn	1.8	2.2	2.6	**4.6**	3.8	**4.6**

Table 8: Acc at 10 values in SR-B benchmark for Mono 100 models initialized with 3 approaches. Bold values highlight the best metric for each language.

SR-T for Mono 100 Models

	Checkpoint 0			Checkpoint 4000		
	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100
eng_Latn	-	-	-	-	-	-
tur_Latn	3.2	4.2	5.2	9.4	**15.6**	8.6
ell_Grek	2.0	2.3	2.4	4.9	**16.4**	13.5
bul_Cyrl	3.7	4.3	4.4	20.8	**48.5**	42.1
ces_Latn	4.0	4.7	5.3	19.8	**30.4**	19.8
kor_Hang	2.8	4.4	4.1	7.4	**11.3**	8.3
kat_Geor	3.4	5.9	6.2	8.7	**14.3**	11.7
fry_Latn	19.7	23.7	27.8	40.5	**46.8**	35.3
zsm_Latn	5.2	9.8	9.6	13.9	**34.1**	22.3
khm_Khmr	2.6	4.6	4.3	3.9	**9.8**	6.9
jpn_Japn	-	-	-	-	-	-
yue_Hani	1.8	5.3	4.4	4.7	**7.3**	4.9
tuk_Latn	7.4	11.3	7.9	15.3	**18.2**	13.3
uig_Arab	2.1	2.3	2.4	2.3	**2.6**	2.0
pam_Latn	1.6	2.3	3.0	3.4	**3.5**	2.8
kab_Latn	2.0	2.2	2.9	2.9	**3.4**	2.4
gla_Latn	3.4	4.5	4.1	4.1	**4.7**	4.2
mhr_Cyrl	2.4	3.2	2.5	2.8	**4.1**	3.5
swh_Latn	11.3	11.5	11.3	13.1	**15.6**	11.3
cmn_Hani	3.7	4.8	3.7	4.9	**9.4**	7.5
pes_Arab	2.9	4.2	**4.3**	2.6	2.9	2.1
dtp_Latn	3.1	3.3	4.0	3.9	**5.1**	3.5

Table 9: Acc at 10 values in SR-T benchmark for Mono 100 models initialized with 3 approaches. Bold values highlight the best metric for each language.

NER for Mono 100 Models

	Checkpoint 0			Checkpoint 4000		
	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100
eng_Latn	75.9	75.3	75.4	**80.9**	80.5	80.6
tur_Latn	32.0	32.8	32.3	47.7	**55.9**	52.1
ell_Grek	10.7	10.2	9.8	37.0	**47.2**	45.0
bul_Cyrl	19.0	20.5	24.0	54.3	64.7	**65.5**
ces_Latn	36.1	37.8	37.4	59.6	**61.9**	61.4
kor_Hang	11.3	13.8	10.9	17.1	**29.2**	27.3
kat_Geor	11.9	14.6	14.0	25.9	**34.8**	30.9
fry_Latn	29.9	30.2	32.0	68.0	**70.1**	65.9
zsm_Latn	-	-	-	-	-	-
khm_Khmr	17.2	17.4	14.6	30.7	**35.9**	32.6
jpn_Japn	-	-	-	-	-	-
yue_Hani	7.7	7.4	6.0	9.2	**14.3**	12.1
tuk_Latn	24.4	25.2	26.9	**41.7**	40.6	40.0
uig_Arab	14.7	14.6	16.9	**20.9**	16.4	18.7
pam_Latn	-	-	-	-	-	-
kab_Latn	-	-	-	-	-	-
gla_Latn	25.7	24.9	20.3	45.0	**51.5**	39.3
mhr_Cyrl	9.4	11.1	8.6	21.6	**36.2**	36.1
swh_Latn	-	-	-	-	-	-
cmn_Hani	-	-	-	-	-	-
pes_Arab	-	-	-	-	-	-
dtp_Latn	-	-	-	-	-	-

Table 10: F1 scores in NER benchmark for Mono 100 models. Bold values highlight the best metric for the language.

POS for Mono 100 Models

	Checkpoint 0			Checkpoint 4000		
	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100	Random-mono-100	OFA-mono-100	HYPEROFA-mono 100
eng_Latn	94.8	94.9	94.9	**95.8**	**95.8**	**95.8**
tur_Latn	25.7	26.9	26.5	41.9	48.4	**49.2**
ell_Grek	16.8	18.3	17.2	54.0	75.3	**76.5**
bul_Cyrl	21.8	24.2	23.1	77.8	82.8	**83.7**
ces_Latn	25.3	27.0	26.2	78.0	79.4	**80.4**
kor_Hang	19.9	21.9	20.9	35.6	**40.7**	40.1
kat_Geor	-	-	-	-	-	-
fry_Latn	-	-	-	-	-	-
zsm_Latn	-	-	-	-	-	-
khm_Khmr	20.9	20.3	23.1	13.2	**15.3**	10.4
jpn_Japn	-	-	-	-	-	-
yue_Hani	16.8	17.5	17.5	**32.7**	32.6	30.7
tuk_Latn	-	-	-	-	-	-
uig_Arab	-	-	-	-	-	-
pam_Latn	-	-	-	-	-	-
kab_Latn	20.2	20.9	20.8	31.1	**40.7**	39.8
gla_Latn	-	-	-	-	-	-
mhr_Cyrl	-	-	-	-	-	-
swh_Latn	-	-	-	-	-	-
cmn_Hani	-	-	-	-	-	-
pes_Arab	-	-	-	-	-	-
dtp_Latn	-	-	-	-	-	-

Table 11: F1 scores in POS benchmark for Mono 100 models. Bold values highlight the best metric for the language.

SR-B for Multi 400 Models

	Checkpoint 0			Checkpoint 4000		
	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400
eng_Latn	-	-	-	-	-	-
tur_Latn	13.6	13.6	15.8	75.4	**76.0**	**76.0**
ell_Grek	6.2	6.6	8.2	50.0	49.6	**50.8**
bul_Cyrl	16.6	15.4	15.6	**82.4**	82.0	**82.4**
ces_Latn	15.8	18.4	17.8	73.4	**74.6**	74.4
kor_Hang	9.8	9.8	9.8	**63.2**	62.4	62.8
kat_Geor	3.0	4.8	6.2	43.2	**44.2**	43.8
fry_Latn	5.0	5.6	5.6	49.0	50.0	**51.0**
zsm_Latn	17.2	18.2	18.6	80.4	**84.6**	84.4
khm_Khmr	3.6	3.0	4.2	30.8	**31.6**	31.4
jpn_Japn	-	-	-	-	-	-
yue_Hani	3.0	3.2	3.4	**13.6**	13.0	12.8
tuk_Latn	5.6	4.4	5.4	46.0	54.4	**54.6**
uig_Arab	4.6	7.0	6.6	33.8	**34.8**	34.6
pam_Latn	5.2	4.2	4.4	20.4	21.0	**23.2**
kab_Latn	3.0	4.0	3.0	8.0	**10.4**	9.4
gla_Latn	4.0	3.6	4.0	**28.6**	27.4	25.8
mhr_Cyrl	3.2	3.8	3.6	20.0	25.0	**25.2**
swh_Latn	8.2	9.6	8.6	34.8	**40.0**	38.0
cmn_Hani	17.4	17.8	17.2	28.2	**30.0**	28.2
pes_Arab	14.2	16.4	22.2	28.6	**30.4**	29.6
dtp_Latn	2.6	2.6	3.4	**5.2**	**5.2**	4.6

Table 12: Acc@10 values in SR-B benchmark for Multi 400 models initialized with 3 approaches. Bold values highlight the best metric for each language.

SR-T for Multi 400 Models

	Checkpoint 0			Checkpoint 4000		
	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400
eng_Latn	-	-	-	-	-	-
tur_Latn	22.8	22.2	23.0	87.7	**87.8**	87.4
ell_Grek	21.2	21.0	20.3	79.8	**80.8**	80.2
bul_Cyrl	34.4	35.7	36.1	**88.3**	88.1	88.2
ces_Latn	25.3	25.3	25.5	83.2	**84.6**	83.4
kor_Hang	21.1	21.3	21.4	**79.3**	79.1	78.9
kat_Geor	12.1	13.1	12.1	63.5	**64.6**	**64.6**
fry_Latn	35.3	33.5	33.0	84.4	**86.7**	83.8
zsm_Latn	31.4	32.2	32.7	90.5	**91.4**	90.7
khm_Khmr	5.0	4.6	5.3	51.8	**52.6**	52.4
jpn_Japn	-	-	-	-	-	-
yue_Hani	22.1	22.5	22.3	63.8	59.4	**64.9**
tuk_Latn	14.3	15.3	14.3	48.8	**51.2**	**51.2**
uig_Arab	7.0	8.0	7.8	54.2	56.3	**57.2**
pam_Latn	4.4	4.8	4.5	7.0	**7.8**	7.5
kab_Latn	2.5	3.6	3.1	7.9	7.4	**8.9**
gla_Latn	5.4	5.3	5.3	33.2	**36.1**	33.2
mhr_Cyrl	2.8	3.2	3.6	17.8	20.2	**22.5**
swh_Latn	21.0	20.5	20.5	35.4	**36.4**	36.2
cmn_Hani	33.1	33.5	32.9	**65.0**	60.7	62.5
pes_Arab	27.2	28.5	27.6	59.3	57.4	**63.1**
dtp_Latn	3.6	4.1	3.5	5.7	**6.3**	5.4

Table 13: Acc@10 values in SR-T benchmark for Multi 400 models initialized with 3 approaches. Bold values highlight the best metric for each language.

NER for Multi 400 Models

	Checkpoint 0			Checkpoint 4000		
	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400
eng_Latn	78.1	78.4	77.9	**81.3**	81.2	**81.3**
tur_Latn	55.9	59.2	58.3	**72.8**	**72.8**	72.0
ell_Grek	58.1	56.8	59.6	**70.6**	69.3	70.2
bul_Cyrl	63.7	64.4	64.3	**76.9**	76.4	76.0
ces_Latn	61.7	61.2	61.5	75.9	75.8	**76.0**
kor_Hang	39.8	41.2	41.1	48.5	48.8	**49.1**
kat_Geor	48.9	52.1	53.1	62.2	62.3	**62.9**
fry_Latn	56.3	58.8	56.6	78.1	**78.4**	76.9
zsm_Latn	-	-	-	-	-	-
khm_Khmr	36.1	37.3	33.5	45.2	43.5	**47.1**
jpn_Japn	-	-	-	-	-	-
yue_Hani	20.7	20.0	23.8	16.2	**23.8**	21.0
tuk_Latn	30.2	34.3	35.5	56.7	**57.9**	55.7
uig_Arab	28.2	34.6	34.8	**48.2**	47.0	45.5
pam_Latn	-	-	-	-	-	-
kab_Latn	-	-	-	-	-	-
gla_Latn	37.5	39.2	38.0	56.8	55.9	**61.7**
mhr_Cyrl	27.8	23.7	27.5	48.3	51.0	**51.1**
swh_Latn	-	-	-	-	-	-
cmn_Hani	-	-	-	-	-	-
pes_Arab	-	-	-	-	-	-
dtp_Latn	-	-	-	-	-	-

Table 14: F1 scores in NER benchmark for Multi 400 models initialized with 3 approaches. Bold values highlight the best metric for each language.

POS for Multi 400 Models

	Checkpoint 0			Checkpoint 4000		
	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400	Random-multi-400	OFA-multi-400	HYPEROFA-multi 400
eng_Latn	95.3	95.4	95.3	**95.8**	**95.8**	**95.8**
tur_Latn	62.4	61.5	62.4	**71.4**	**71.4**	71.3
ell_Grek	84.6	83.4	84.0	**86.0**	85.9	**86.0**
bul_Cyrl	85.9	85.6	86.1	87.8	**88.0**	**88.0**
ces_Latn	74.3	73.9	73.0	**82.7**	**82.7**	82.5
kor_Hang	52.0	52.2	52.5	52.4	**52.6**	52.5
kat_Geor	-	-	-	-	-	-
fry_Latn	-	-	-	-	-	-
zsm_Latn	-	-	-	-	-	-
khm_Khmr	-	-	-	-	-	-
jpn_Japn	-	-	-	-	-	-
yue_Hani	40.2	25.5	28.6	27.2	**27.3**	27.1
tuk_Latn	-	-	-	-	-	-
uig_Arab	58.8	57.5	57.9	**69.2**	68.9	68.9
pam_Latn	-	-	-	-	-	-
kab_Latn	-	-	-	-	-	-
gla_Latn	31.7	31.4	33.6	60.4	**60.7**	60.6
mhr_Cyrl	-	-	-	-	-	-
swh_Latn	-	-	-	-	-	-
cmn_Hani	-	-	-	-	-	-
pes_Arab	-	-	-	-	-	-
dtp_Latn	-	-	-	-	-	-

Table 15: F1 scores in POS benchmark for Multi 400 models initialized with 3 approaches. Bold values highlight the best metric for the language.