TRANSLICO: A Contrastive Learning Framework to Address the Script Barrier in Multilingual Pretrained Language Models

Yihong Liu[*], **Chunlan Ma**[*], **Haotian Ye**[*], **and Hinrich Schütze**

Center for Information and Language Processing, LMU Munich
Munich Center for Machine Learning (MCML)
{yihong, chunlan, yehao}@cis.lmu.de

Abstract

The world's more than 7000 languages are written in at least 293 scripts.[1] Due to various reasons, many closely related languages use different scripts, which poses a difficulty for multilingual pretrained language models (mPLMs) in learning crosslingual knowledge through lexical overlap. As a consequence, mPLMs are faced with a script barrier: representations from different scripts are located in different subspaces, which can result in crosslingual transfer involving languages of different scripts performing suboptimally. To address this problem, we propose **TRANSLICO**, a framework that optimizes the Transliteration Contrastive Modeling (TCM) objective to fine-tune an mPLM by contrasting sentences in its training data and their transliterations in a unified script (in our case Latin[2]), which enhances uniformity in the representation space for different scripts. Using Glot500-m (ImaniGooghari et al., 2023), an mPLM pretrained on over 500 languages, as our source model, we fine-tune it on a small portion (5%) of its training data, and refer to the resulting model as **FURINA**. We show that FURINA not only better aligns representations from distinct scripts but also outperforms the original Glot500-m on various zero-shot crosslingual transfer tasks. Additionally, we achieve consistent improvement in a case study on the Indic group where the languages exhibit areal features but use different scripts. We make our code and models publicly available.[3]

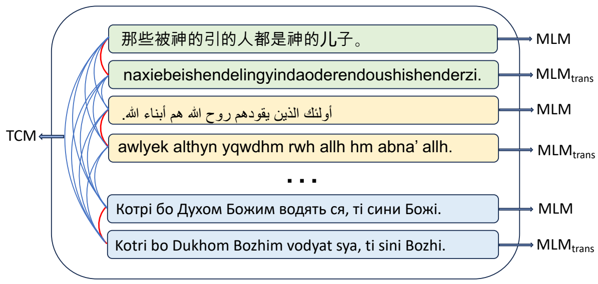


Figure 1: An illustration of applying TRANSLICO to a single batch of data during fine-tuning. The training data is used by the two training objectives in TRANSLICO: Masked Language Modeling (MLM) and Transliteration Contrastive Modeling (TCM). MLM is applied to both the original sentences and their Latin transliterations. TCM is used to learn better-aligned cross-script representations by contrasting the positive pairs (paired data connected with red lines) against the negative pairs (the remaining samples connected with blue lines).

1 Introduction

In recent years, mPLMs have made impressive progress in various crosslingual transfer tasks (Conneau et al., 2018; Hu et al., 2020; Liang et al., 2020). Such achievement is mainly due to the availability of monolingual corpora of many languages (Costa-jussà et al., 2022; Adebara et al., 2023; ImaniGooghari et al., 2023), the amelioration of model architectures suitable for scaling up (Vaswani et al., 2017; Peng et al., 2023), as well as the advancement of self-supervised learning objectives (Devlin et al., 2019; Raffel et al., 2020; Brown et al., 2020). Despite the fact that mPLMs present attractive performance in high-resource languages, those models often gain unsatisfactory results for low-resource languages, especially when the writing systems or *scripts* are different from the transfer source languages (Muller et al., 2021).

This undesired behavior is related to the script barrier in the representation space, where different scripts are located in different subspaces (Wen-Yi and Mimno, 2023). To tackle this problem, transliteration or romanization[4] is leveraged in some recent work (Dhamecha et al., 2021; Muller et al.,

[*]Equal contribution.

[1]https://worldswritingsystems.org/

[2]Throughout this paper we use Latin to refer to the Latin script, not the Latin language.

[3]https://github.com/cisnlp/TransliCo

[4]Romanization is a specific type of transliteration that involves converting non-Latin scripts into the Latin script.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2476–2499
August 11-16, 2024

2021; Moosa et al., 2023; Purkayastha et al., 2023): all languages from different scripts are converted into one common script and the language model is pretrained or adapted with transliterated data. For testing and inference, the queries also need to be transliterated, as the model only supports one script, the pretraining or adaptation script of the model.

However, this line of approaches presents two limitations. First, it does not break the script barrier, rather, it circumvents it. The representations from different scripts are still not aligned. Second, for some tasks, e.g., question answering, it is necessary to transliterate the response back to the original script because we cannot assume that end users know the common script. Unfortunately, transliteration and transliterating back to the original script is not immune to information loss (Amrhein and Sennrich, 2020). The romanized words in many languages, e.g., Chinese, Japanese, and Korean, can be converted to different words in their original scripts, which unfortunately leads to ambiguity.

In this paper, we present TRANSLICO, a contrastive learning framework to address the script barrier in the representation space of mPLMs in a way that overcomes the limitations of prior work. To start with, a small portion of the data from the pretraining corpus of an mPLM is used to generate Latin transliteration, using `Uroman`[5] (Hermjakob et al., 2018). Then we create paired data using sentences in their original script and their transliterations. The data is subsequently used by the two objectives: Masked Language Modeling (MLM) and Transliteration Contrastive Modeling (TCM). MLM is applied to both the original sentences and their transliterations; we use TCM to learn better-aligned representations by contrasting the positive pairs (paired data) against negative pairs (the remaining in-batch samples) as shown in Figure 1.

Using Glot500-m (ImaniGooghari et al., 2023) as our source model, we evaluate TRANSLICO both "globally" and "locally". Specifically, we fine-tune Glot500-m on 5% of its pretraining data of all languages and refer to the resulting model as FURINA. We show that FURINA aligns representations from different scripts better and it generally outperforms the baselines on sentence retrieval, sequence labeling, and text classification tasks for different script groups. Our ablation study indicates MLM and TCM in TRANSLICO are both important for achieving good crosslingual performance. We additionally conduct a case study on Indic languages, a group of languages that show areal features and use different scripts. $\text{FURINA}_{\text{Indic}}$ fine-tuned by TRANSLICO using the data from Indic languages shows consistent improvement over the baseline.

The main contributions of this work are summarized as follows: (i) We present TRANSLICO, a simple but effective framework, to address the script barrier in the representation space of mPLMs. (ii) We conduct extensive and controlled experiments on a variety of crosslingual tasks and show TRANSLICO boosts performance. (iii) We show the framework encourages the representations from different scripts to be better aligned. (iv) In a case study on Indic languages, we demonstrate that TRANSLICO also works for areal languages that have shared vocabulary but use distinct scripts.

2 Related Work

Transliteration refers to converting languages from one script into another script (Wellisch et al., 1978). Transliteration can increase lexical overlap, and therefore it has been shown to substantially improve the performance of neural machine translation for low-resource languages of different scripts (Gheini and May, 2019; Goyal et al., 2020; Amrhein and Sennrich, 2020). Several studies also demonstrate that transliteration can enhance the crosslinguality of mPLMs across various dimensions. For instance, Dhamecha et al. (2021) transliterate seven Indo-Aryan family languages into Devanagari and show that common-script representations facilitate fine-tuning in a multilingual scenario. Purkayastha et al. (2023) show that, by transliterating into Latin, better performance is achieved when adapting mPLMs to new languages, particularly for low-resource languages. More recently, Moosa et al. (2023) focus on Indic languages and show that models directly pretrained on transliterated corpora in the Latin script achieve better performance. However, the downside is that the model only supports one script and loses the ability to deal with the scripts in which the languages were originally written. This is not optimal when we expect predictions or generations in the original scripts. In contrast to this line of work, we aim to directly break the script barrier, instead of circumventing it by limiting the model to one common script. We use transliterations in our fine-tuning framework to improve the alignment across different scripts: the model after fine-tuning still

[5] https://github.com/isi-nlp/uroman

supports the scripts it originally did.

Contrastive learning is a method for learning meaningful representations by contrasting positive pairs against negative pairs (Chopra et al., 2005; Hadsell et al., 2006). This type of approach has achieved great success in learning visual representations (Schroff et al., 2015; Oord et al., 2018; Chen et al., 2020; He et al., 2020). Contrastive learning also demonstrates its effectiveness in NLP, especially for learning sentence representations (Gao et al., 2021; Zhang et al., 2022; Wu et al., 2022b; Zhang et al., 2023). One major problem in contrastive learning is how to construct contrastive pairs. For a monolingual scenario, depending on specific downstream tasks, the positive pairs are usually constituted through data transformation or data augmentation strategies (Zhang et al., 2020; Yan et al., 2021; Wu et al., 2022a; Xu et al., 2023a) whereas the negative pairs are typically the remaining in-batch samples (Xu et al., 2023b). In a multilingual scenario where parallel data is available, translations from different languages can be used to construct the positive pairs (Reimers and Gurevych, 2019; Pan et al., 2021b; Chi et al., 2021; Wei et al., 2021). Unfortunately, large parallel corpora are mostly available for high-resource languages. Therefore, aiming to improve crosslinguality, especially for under-represented languages, we start from the perspective of the script and construct positive pairs by using sentences in their original script and their Latin transliterations that can be easily obtained. Then we fine-tune an mPLM with our contrastive framework TRANSLICO. In this way, our work also resembles some post-pretraining alignment approaches (Pan et al., 2021a; Feng et al., 2022; Ji et al., 2023) that fine-tune a PLM using token-level or sentence-level translations.

3 Methodology

We present **TRANSLICO**, a simple framework to address the script barrier by fine-tuning a PLM on a small portion of the data that is used to pretrain the model. The framework consists of two training objectives: Masked Language Modeling (Devlin et al., 2019) and Transliteration Contrastive Modeling. We illustrate our framework in Figure 2 and introduce our training objectives in the following.

3.1 Masked Language Modeling

The MLM training objective is to take an input sentence $X = [x_1, x_2, \cdots, x_n]$, randomly replace a certain percentage (15% in our case) of tokens by [mask] tokens, and then train the model to predict the original tokens using an MLM head. Formally, let $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_n]$ be the contextualized representations at the last layer of the Transformer model (Vaswani et al., 2017) (the output of the last Transformer block of the model) given the input sentence X. Following the notations used by Meng et al. (2021), we compute the MLM loss as follows:

$$\mathcal{L}_{\text{MLM}} = \mathbb{E}\left[-\sum_{i \in \mathcal{M}} \log p_{\text{MLM}}(x_i|\boldsymbol{h}_i)\right]$$

where $\mathcal{M}$ is the set of masked positions in the input sentence X and $p_{\text{MLM}}(x_i|\boldsymbol{h}_i)$ is the probability of outputting the original token giving $\boldsymbol{h}_i$ from the vocabulary V, computed by the MLM head.

Instead of only performing MLM for the sentences in their original scripts, we also perform MLM for their transliterations in Latin script: $X^{\text{trans}} = [x_1, x_2, \cdots, x_m]$, which are obtained by using `Uroman` (Hermjakob et al., 2018). The MLM loss for transliteration data is referred to as $\mathcal{L}^{\text{trans}}_{\text{MLM}}$. By doing this, we can improve the crosslinguality of the model across related languages that use different scripts, as transliteration has shown to be effective in capturing morphological inflection (Murikinati et al., 2020) and generating shared subwords between related languages (Muller et al., 2021; Dhamecha et al., 2021; Moosa et al., 2023). The intuition is that, as all sentences are consistently transliterated into Latin using the same tool, this can bring about vocabularies that have more shared subwords that are originally in different scripts. The improved lexical overlap therefore encourages the model to generate more crosslingual representations through MLM objective.

3.2 Transliteration Contrastive Modeling

Modeling a sentence and its transliterations separately does not necessarily lead to a good alignment between two types of scripts. Therefore, we propose to learn more similar and robust representations of a pair of sentences in different scripts using the contrastive learning objective. Sentence-level contrastive learning generally aims to align a positive pair of sentences by distinguishing them from negative or unrelated samples of sentences (Gao et al., 2021; Meng et al., 2021; Zhang et al., 2023). In our framework, we simply let a positive pair be a sentence in its original script and its transliteration in the Latin script, and other sentences in a training batch be the negative samples to contrast with.

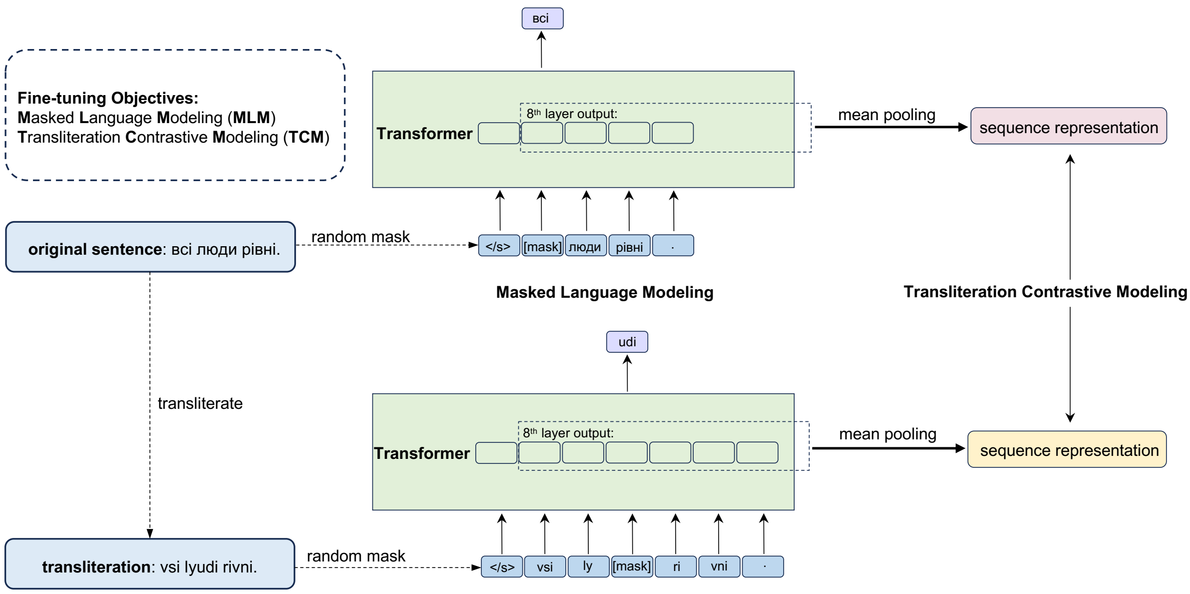


Figure 2: Overview of TRANSLICO. We perform **Masked Language Modeling** for a sentence in its original script and its transliteration in the Latin script. Meanwhile, we calculate the sequence representations of the paired input by mean pooling their 8th layer output (ignoring the special token except for [mask] token). We then perform **Transliteration Contrastive Modeling** on the paired representations against negative pairs (not shown) in a batch.

Formally, let a training batch for TCM objective be $B = \{X_1^{\text{orig}}, X_1^{\text{latn}}, \cdots, X_N^{\text{orig}}, X_N^{\text{latn}}\}$, where N is the batch size, X_i^{orig} is the ith sentence in its original script and X_i^{latn} is its Latin transliteration. Similar to the setting of Meng et al. (2021), a positive pair (X, X^+) consists of a symmetrical contrast pair, i.e., both $(X_i^{\text{orig}}, X_i^{\text{latn}})$ and $(X_i^{\text{latn}}, X_i^{\text{orig}})$, whereas the negative samples are all the remaining sentences in their original scripts and their transliterations in the training batch using a slightly abusing notation: $B^- = B \setminus \{X, X^+\}$. Then the contrastive loss is defined as follows:

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

where $\text{NEG} = \sum_{X^- \in B^-} \exp(\text{sim}(\boldsymbol{h}, \boldsymbol{h}^-)/\tau)$, $\text{sim}(\cdot)$ is the similarity measure (cosine similarity is used), τ is the temperature (set to 1 by default), $\boldsymbol{h}$, $\boldsymbol{h}^+$ and $\boldsymbol{h}^-$ are the representations of X, X^+ and X^- respectively, which are computed by mean pooling the output of the 8th layer. Choosing the 8th layer is based on previous empirical findings, as the first layers are weak in terms of crosslinguality whereas the last layers are too specialized on the pretraining task (Jalili Sabet et al., 2020; Chang et al., 2022). The numerator improves the *alignment*, i.e., encouraging the model to assign similar representations to similar samples, while the denominator improves the *uniformity*, i.e., encouraging the representations to be uniformly distributed on the unit hypersphere (Wang and Isola, 2020).

Since all sentences are expected to have representations similar to their Latin transliterations by the contrast training objective, this implicitly encourages sentences in different scripts to be in the same subspace. Intuitively, the Latin script acts as a bridge to connect all other scripts, and therefore all other scripts are better aligned. We show better script-neutrality is achieved in the representation space compared with the original mPLM in §5.2.

3.3 Overall Training

The overall training objective of TRANSLICO is then the sum of the MLM loss (from the original data and the transliteration data) and the TCM loss:

$$\mathcal{L}_{\text{training}} = \lambda_1 \mathcal{L}_{\text{MLM}} + \lambda_2 \mathcal{L}_{\text{MLM}}^{\text{trans}} + \lambda_3 \mathcal{L}_{\text{TCM}}$$

where λ_1, λ_2 and λ_3 are the weights for each loss. Following (Meng et al., 2021), we set $\lambda_1 = \lambda_2 = \lambda_3 = 1$, as we also find the initial losses from each part during training are in similar magnitude. By fine-tuning an mPLM with this overall training objective, the model is expected to **(1)** not forget the language modeling ability gained in its pretraining phase; **(2)** be able to model sentences in both their original scripts and in their Latin transliterations and **(3)** learn to better align representations from different scripts in the same subspace.

4 Experiments

4.1 Setups

We use Glot500-m (ImaniGooghari et al., 2023), a continued pretrained model from XLM-R (Conneau et al., 2020), as our source mPLM. The training data of Glot500-m is Glot500-c, which contains 1.5B sentences from 511 languages and 30 scripts (534 language-scripts[6] in total). For each language-script, we randomly select 5% sentences from its training set in Glot500-c as the training data. We then concatenate these sentences from all language-scripts and use `Uroman` (Hermjakob et al., 2018), a tool for universal romanization, to transliterate them into the Latin script. Finally, our training data consists of around 75M pairs (a pair is a sentence in the original script and its Latin transliteration). Examples of `Uroman` transliteration are shown in Table 1. Note that we also include the sentences originally in Latin script and perform transliteration for them. This is because we want to (1) preserve the model's ability to model languages in Latin script (2) increase lexical overlap by including data where diacritics are removed (done by `Uroman`) and (3) improve the overall robustness of the model. We show in §5.1 how the model fine-tuned in this setting outperforms the model fine-tuned without Latin data. The fine-tuned model by using the proposed TRANSLICO framework is referred to as **FURINA**. See §A for detailed hyperparameters. Except for the evaluation performed on the original datasets discussed in the main content below, we also evaluate the resulting models on the transliterated datasets. That is, we use `Uroman` to transliterate the datasets from the original script to the Latin script, and then perform evaluation on the new datasets. The performance on transliterated datasets is shown and discussed in §D.

4.2 Downstream Tasks

Sentence Retrieval. Two datasets are considered: Tatoeba (Artetxe and Schwenk, 2019) (SR-T) and Bible (SR-B). We select up to 1,000 English-aligned sentences for SR-T, following the same setting used by Hu et al. (2020); and up to 500 sentences for SR-B. We report the top-10 accuracy for both tasks. Following Jalili Sabet et al. (2020), the similarity is calculated by using the average of contextualized word embeddings at the 8th layer.

[6] A language-script is a combination of the ISO 639-3 code and the script.

Those who are led by the Spirit of God are children of God.		
Chinese	original	那些 被 神 的 引 的 人 都 是 神 的 儿子。
	transliteration	naxie bei shen de ling yindao de ren dou shi shen de rzi.
Arabic	original	أولئك الذين يقودهم روح الله هم أبناء الله.
	transliteration	awlyek althyn yqwdhm rwh allh hm abna' allh.
Ukrainian	original	Котрі бо Духом Божим водять ся, ті сини Божі.
	transliteration	Kotri bo Dukhom Bozhim vodyat sya, ti sini Bozhi.
Slovak	original	Tí, ktorých vedie Boží Duch, sú Božími synmi.
	transliteration	Ti, ktorych vedie Bozi Duch, su Bozimi synmi.

Table 1: Examples of `Uroman` transliteration. We select sentences (translations of the sentence "Those who are led by the Spirit of God are children of God.") from four languages that uses different scripts and transliterate them using `Uroman`. We notice some important characteristics of `Uroman`: tones (for Hani script) are not included and diacritics (for Latin script) are removed.

Text Classification. Taxi1500 (Ma et al., 2023), a multilingual 6-class text classification dataset available in more than 1,500 languages, is used. We select a subset of language-scripts supported by the model for evaluation. We report the zero-shot crosslingual performance (in macro F1 scores) using the English train set for fine-tuning and selecting the best model on the English dev set.

Sequence Labeling. Two types of tasks are considered: named entity recognition (NER) and Part-Of-Speech (POS) tagging. We use WikiANN (Pan et al., 2017) for NER and Universal Dependencies (de Marneffe et al., 2021), version v2.11, for POS. We report the zero-shot performance (in macro F1 scores) for both tasks.

4.3 Results and Discussion

To better illustrate how the proposed TRANSLICO framework can influence the performance of different scripts, we group all language-scripts by their scripts and report the average performance for each group. The results of XLM-R, Glot500-m, and FURINA are shown in Table 2. We see an overall improvement for FURINA compared with Glot500-m in each task except for SR-T. We conjecture that the sub-optimal performance on SR-T can be related to the domain shift and the small set of languages supported by Tatoeba. Our fine-tuning data consists of sentences of many low-resource languages that come from genre-specific corpora such as the Bible, which can be quite different from Tatoeba, which is more modern in terms of the genre and mostly only supports high-resource languages (70 out 98 languages are high-resource languages).

FURINA performs surprisingly well on ST-B. An overall improvement of 10.9 (from 47.2 to 58.1) is achieved. We also see a consistently large increase for each script group of languages: e.g.,

	SR-B			SR-T			Taxi1500			NER			POS		
	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA
Latn	16.2	<u>45.1</u>	**57.4**	55.7	<u>69.1</u>	**73.0**	22.5	<u>52.6</u>	**59.8**	60.3	<u>66.1</u>	**67.3**	68.1	<u>74.4</u>	**75.7**
Cyrl	25.5	<u>60.3</u>	**69.0**	55.5	**74.4**	<u>69.7</u>	30.2	<u>59.8</u>	**63.6**	51.8	<u>65.3</u>	**66.2**	66.7	<u>79.3</u>	**79.5**
Hani	30.4	**43.4**	<u>39.8</u>	<u>62.0</u>	**80.5**	47.7	66.6	<u>68.2</u>	**70.1**	**23.1**	<u>22.2</u>	21.9	<u>22.2</u>	**35.5**	18.2
Arab	36.3	<u>56.4</u>	**61.4**	53.6	**71.8**	<u>56.3</u>	48.5	<u>60.8</u>	**66.5**	45.0	<u>53.4</u>	**57.7**	65.8	<u>68.8</u>	**69.3**
Deva	32.1	<u>60.3</u>	**66.8**	68.6	**81.8**	<u>71.9</u>	49.5	<u>66.6</u>	**73.2**	<u>56.9</u>	56.2	**58.9**	58.3	<u>59.8</u>	**60.8**
Other	33.8	<u>49.0</u>	**53.6**	<u>59.7</u>	**71.1**	57.6	49.5	<u>59.5</u>	**65.2**	45.2	**50.4**	**50.4**	65.9	**68.8**	<u>67.1</u>
All	19.3	<u>47.2</u>	**58.1**	56.6	**70.7**	<u>68.8</u>	26.7	<u>54.3</u>	**61.0**	55.3	<u>61.6</u>	**62.8**	65.6	<u>71.8</u>	**71.9**

Table 2: Performance of FURINA and baselines on five downstream tasks across 5 seeds. We report the average performance for groups of languages using one of the five major scripts in the fine-tuning data: **Latn** (Latin), **Cyrl** (Cyrillic), **Hani** (Hani), **Arab** (Arabic), and **Deva** (Devanagari). We collect the remaining languages in the group "**Other**". In addition, we also report the average over all languages (group "**All**"). FURINA generally performs better than other baselines except on SR-T. **Bold** (<u>underlined</u>): best (second-best) result for each task in each group.

	SR-B			SR-T			Taxi1500			NER			POS		
	Latn	Other	All	Latn	Other	All	Latn	Other	All	Latn	Other	All	Latn	Other	All
FURINA$_{\text{-Latn}}$	**52.1**	**58.6**	**53.5**	**65.9**	<u>62.0</u>	**64.6**	**56.6**	**65.2**	**58.2**	66.1	<u>53.6</u>	<u>61.5</u>	**73.4**	<u>66.7</u>	**70.9**
- w/o TCM	<u>41.2</u>	<u>58.5</u>	<u>44.9</u>	<u>57.6</u>	**67.8**	<u>61.2</u>	<u>52.9</u>	<u>63.4</u>	<u>54.9</u>	**66.2**	**54.4**	**61.9**	<u>72.7</u>	65.5	70.0
- w/o MLM	31.2	26.3	30.2	40.2	25.4	35.1	44.8	52.8	46.4	<u>66.4</u>	48.3	59.8	72.6	**66.8**	<u>70.4</u>
FURINA$_{\text{+Latn}}$	**57.4**	**60.8**	**58.1**	**73.0**	<u>60.9</u>	**68.8**	**59.8**	**65.8**	**61.0**	**67.3**	**54.9**	**62.8**	<u>75.7</u>	<u>65.5</u>	**71.9**
- w/o TCM	<u>45.5</u>	<u>58.4</u>	<u>48.3</u>	<u>65.9</u>	**67.1**	<u>66.3</u>	<u>57.8</u>	<u>64.4</u>	<u>59.1</u>	<u>67.2</u>	<u>54.2</u>	<u>62.4</u>	**75.9**	64.5	<u>71.6</u>
- w/o MLM	41.2	35.9	40.0	63.4	41.6	55.9	50.8	55.2	51.7	66.6	47.2	59.5	73.9	**66.5**	71.1

Table 3: Ablation study. We investigate the effect of incorporating Latin script data and their `Uroman` transliteration (models are therefore classified into two groups: FURINA$_{\text{-Latn}}$ and FURINA$_{\text{+Latn}}$). We also explore the influence of the MLM and TCM objectives. The model generally performs worse when any one of the objectives is missing. In addition, including Latin script data can improve the overall performance for both Latin script languages and languages using other scripts on all tasks. **Bold** (<u>underlined</u>): best (second-best) result per controlled group.

12.3 for Latin script languages and 8.7 for Cyrillic languages. However, for Hani script (Chinese characters) languages, we see a sudden drop in performance, which can also be seen in other tasks such as SR-T, NER, and POS. We hypothesize that `Uroman` is suboptimal for Hani script because a great deal of important information is lost in the transliteration: both due to the removal of tones and due to the conflation of semantically different characters that are pronounced identically (e.g., 氦 (helium) v.s 害 (harm), both pronounced hài in Mandarin). In addition, Chinese characters are logograms. Even if the transliteration contains correct tones, the transliterated words potentially lose semantic or contextual nuances and are more prone to ambiguity (Amrhein and Sennrich, 2020), thus resulting in a performance drop. However, note that Hani languages are high-resource languages, so this result does not diminish our hypothesis that transliteration helps low-resource languages.

For other types of tasks, we also see a consistent improvement. In both NER and POS, FURINA achieves better performance than Glot500-m in 4 out of 5 script groups (except for Hani as discussed above). Compared with token-level classification (NER and POS), we see a more prominent increase in sequence-level classification (Taxi1500): the overall F1 score is increased by 6.7 (more than 10%) compared with Glot500-m, and FURINA achieves substantially better performance for each script group. The results indicate that the proposed contrastive learning framework TRANSLICO boosts crosslingual transfer learning, especially for sequence-level tasks, e.g., sentence retrieval and sentence classification. We also evaluate both Glot500m and FURINA under the common script scenario, i.e., transliterating the evaluation dataset to Latin script. See the evaluation in §D.

5 Analysis

5.1 Ablation Study

We conduct ablation experiments on all five tasks, using the same hyperparameters and Glot500-m as the source model. Specifically, we explore the influence of MLM and TCM objectives in TRANSLICO. In addition, we investigate the importance of incorporating data from Latin script languages and classify the model variants into two groups (with Latin script languages or without). In our preliminary experiments, we also explore the weight of the TCM loss (e.g., 0.1, 0.5, and 1) and find the

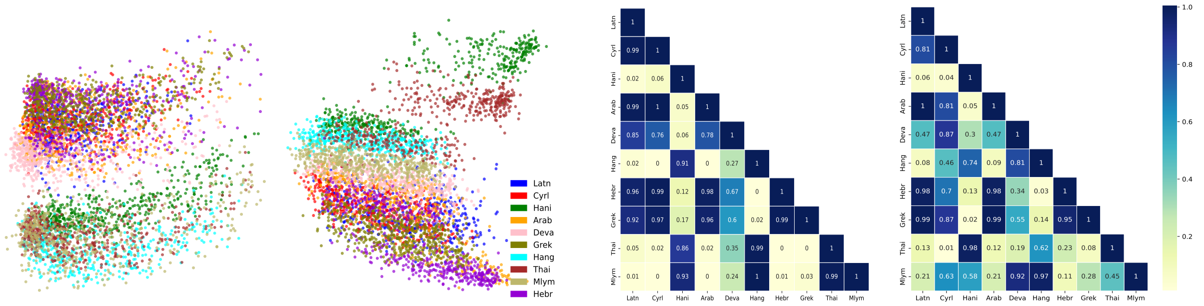


Figure 3: (i) PCA of sentence representations from layer 8 (mean-pooling of contextualized token embeddings, dim=768) of FURINA (Subfigure 1) and Glot500-m (Subfigure 2). Points are sentence representations. Colors indicate scripts. (ii) Pairwise cosine similarity between centroids of scripts of FURINA (Subfigure 3) and Glot500-m (Subfigure 4). FURINA better represents scripts in several cases, e.g., it better aligns related scripts Latn and Cyrl, and, it better separates the unrelated scripts Cyrl and Mlym compared to Glot500-m.

influence is very small as the TCM loss quickly goes to a small value for different weights, possibly due to the simplicity of the contrastive task (the initial magnitude is close though). We report the best result for each model variant in Table 3.

The model generally performs worse when any one of the training objectives is missing. This holds for each group. For example, the overall accuracy in SR-B decreases by 8.6 and 13.3 for FURINA$_{\text{-Latn}}$ when TCM or MLM is missing. The only exception is the accuracy for "Other" in SR-T. We notice when TCM is missing, the result increases by 5.8 (FURINA$_{\text{-Latn}}$) and 6.2 (FURINA$_{\text{+Latn}}$). The possible reason is as follows. The sentences from languages using different scripts in SR-T (Tatoeba sentences are quite simple) are already aligned pretty well. Additionally fine-tuning the model with TCM on a small portion of data (for many low-resource languages the data is related to the Bible) whose domain is different from the domain of SR-T hurts the performance of underrepresented languages.

Interestingly, we find the introduction of TCM and MLM objectives has a more prominent impact on sequence-level (SR-B, SR-T, Taxi1500) than token-level tasks (NER, POS). For example, for FURINA$_{\text{-Latn}}$, all three variants achieve competitive overall performance on token-level tasks: around 60 and 70 F1 scores on NER and POS respectively. This might suggest that TRANSLICO has a relatively small effect on individual token representations as we use sentence-level contrastive learning. In addition, in sequence labeling, the model may be able to transfer prevalent classes such as *verb* and *noun* (ImaniGooghari et al., 2023; Liu et al., 2023) to some extent even without the explicit crosslingual constraint imposed by TCM and MLM.

We also see a consistent improvement when the Latin script data is incorporated into the fine-tuning data. Although there is an occasional small decrease for languages using other scripts (e.g., on SR-T and POS) when comparing FURINA$_{\text{+Latn}}$ and FURINA$_{\text{-Latn}}$, the increase for Latin script languages is prominent for each task. This is expected, as FURINA$_{\text{-Latn}}$ can catastrophically forget the knowledge gained in its pretraining phase (Kirkpatrick et al., 2017). By incorporating the Latin script data and their `Uroman` transliteration, we can further increase lexical overlap and make the model more robust, since `Uroman` has a unified mechanism for romanization and removes all diacritics. For example, the word "salón" in Czech will be the same as the French word "salon" after `Uroman` removing the diacritics on "ó".

5.2 Representation Visualization

To explore how TRANSLICO manipulates the representation space, we visualize the sentence representations from languages that use different scripts. Specifically, we feed Glot500-m and FURINA with 500 sentences from the SR-B task. To facilitate comparison, we only select 10 high-resource languages. Each language uses one of the 10 dominant scripts in the vocabulary of the models (we use GlotScript (Kargaran et al., 2024) to detect the script of the tokens). The languages are English (Latin), Russian (Cyrl), Chinese (Hani), Arabic (Arab), Hindi (Deva), Greek (Grek), Korean (Hang), Malayalam (Mlym), and Hebrew (Hebrew). We obtain the sentence representations by mean-pooling the contextualized token embeddings. We visualize the representations of the 8th layer (also used for SR-B and SR-T) by projecting them to two dimensions with principal component analysis (PCA) in Figure 3 (1st and 2nd subfigures). We

Language	pan	hin	ben	ori	asm	guj	mar	kan	tel	mal	tam	avg
Named Entity Recognition (F1-Score)												
FURINA$_{\text{Indic}}$	92.5	**97.1**	98.6	96.5	**94.5**	**95.7**	**97.2**	**93.9**	**96.7**	**97.2**	**97.0**	**96.1**
Glot500-m	**92.7**	**97.1**	**98.7**	**96.6**	93.5	95.4	97.1	93.5	**96.7**	97.0	**97.0**	96.0
ALBERT$_{US}$	85.4	92.9	97.3	93.5	89.1	80.2	90.6	-	-	-	-	89.9
Wikipedia Section Title Prediction (Accuracy)												
FURINA$_{\text{Indic}}$	81.2	**83.9**	**85.8**	**83.7**	**85.2**	**84.8**	**85.6**	**84.3**	**96.6**	**83.8**	**83.8**	**85.3**
Glot500-m	**81.5**	83.6	85.5	83.2	**85.2**	84.4	84.7	83.9	**96.6**	83.1	83.6	85.0
ALBERT$_{US}$	77.6	82.2	84.4	81.5	81.7	82.4	82.7	-	-	-	-	81.8
Cloze Style Question Answering (Accuracy)												
FURINA$_{\text{Indic}}$	**40.6**	**46.8**	**44.9**	**45.2**	**47.3**	**85.9**	**51.7**	**39.7**	**32.3**	**38.2**	**37.7**	**46.4**
Glot500-m	29.8	28.8	27.7	29.4	31.8	82.7	32.1	28.9	26.3	27.5	29.3	34.0
ALBERT$_{US}$	32.8	38.5	36.4	36.0	37.4	70.2	39.5	-	-	-	-	41.5

Table 4: Evaluation of NER, WSTP, and CSQA (zero-shot) from IndicGLUE Benchmark. FURINA$_{\text{Indic}}$ consistently outperforms Glot500-m in most languages on three tasks. **Bold**: best result for each language in each task. We also show ALBERT$_{US}$ model trained by Moosa et al. (2023) using only romanized data for reference. ALBERT$_{US}$ cannot be compared with the other two models directly, because it uses different pretraining data.

also compute the pairwise normalized cosine similarity between the centroid of each script group (3rd and 4th subfigures). Additionally, we visualize representations from every layer in Appendix E.

It can be seen that the representations from each script roughly form an individual single cluster for Glot500-m. In contrast, representations only form two major clusters for FURINA, where each cluster has certain related scripts. This is evidence that Glot500-m encodes script-sensitive information and distinct but related scripts are not well-aligned, whereas FURINA has learned better script-neutrality for the representations of related scripts. The pairwise similarity also supports our argument: e.g., FURINA has higher similarity scores among (1) Latn, Cyrl, and Deva, and (2) Hani, Thai and Hang, where any script in each group is related to the rest of the scripts. This phenomenon indicates that TRANSLICO is effective in addressing the script barrier in the representation space of mPLMs by improving the similarity of the related scripts. Interestingly, we observe linguistic features and geographical proximity can also be related to the representation subspaces. Hani, Thai and Hang (Hangul) are in the same cluster, which might be explained by the fact that Thai, Korean, and Chinese are spoken in adjacent areas and their vocabularies are mutually influenced. With TRANSLICO, such similarity is further exploited and thus their representations are encouraged to be closer.

5.3 Case Study: Indic Group

To further explore TRANSLICO's performance, we conduct a case study on 12 languages that exhibit areal features such as shared vocabulary. These languages are mostly from the Indo-Aryan group and are mutually influenced with each other linguistically, historically, and phonologically, but use different scripts (Moosa et al., 2023).

Lang.	Sub-family	Script	# Sentence
asm	Eastern Indo-Aryan	Bengali	188.0k
bhi	Eastern Indo-Aryan	Devanagari	4351.4k
guj	Western Indo-Aryan	Gujarati	3.0k
guj	Western Indo-Aryan	Devanagari	4.7k
mai	Eastern Indo-Aryan	Devanagari	4573.8k
nep	Northern Indo-Aryan	Devanagari	704.5k
pan	Northwestern Indo-Aryan	Gurmukhi	5.3k
sin	Insular Indo-Aryan	Sinhala	2874.8k
ben	Eastern Indo-Aryan	Bengali	131.5k
hin	Central Indo-Aryan	Devanagari	40.9k
mar	Southern Indo-Aryan	Devanagari	2905.1k
ori	Eastern Indo-Aryan	Oriya	16.4k
sam	Sanskrit	Devanagari	729.2k

Table 5: The 12 languages in the Indic group used for fine-tuning FURINA$_{\text{Indic}}$. The number of sentences shown is the result of randomly sampling 10% of sentences from Glot500-c for each language.

Similar to the previous settings, we use Glot500-m as our source mPLM with the training data as the only difference. Specifically, we randomly sample 10% of sentences from Glot500-c (Imani-Googhari et al., 2023) for each of the 12 languages. The details of the resulting fine-tuning dataset are shown in Table 5. We then use `Uroman` to transliterate these sentences into the Latin script. Subsequently, we create positive paired data using these sentences and their Latin script transliterations. This results in our training data consisting of 16.5M sentence pairs. We use TRANSLICO to fine-tune Glot500-m on the data and refer to the final model as FURINA$_{\text{Indic}}$. Following the similar setting employed by Moosa et al. (2023), we evaluate FURINA$_{\text{Indic}}$ with three downstream tasks from IndicGLUE (Kakwani et al., 2020): Wikipedia Section Title Prediction (WSTP), Cloze Style Question Answering (CSQA)) and a Named Entity Recognition (NER) task. One major difference between our evaluation and Moosa et al. (2023) is that we always evaluate the languages in their **original** scripts, whereas Moosa et al. (2023) evaluate on the transliterated data in a unified script (Latin). The details for hyperparameters of each task are reported in §A.4. To test the crosslingual transfer ability of FURINA$_{\text{Indic}}$ on the Indic group languages, for WSTP and NER, we fine-tune the model on all languages at once and then evaluate the model on the test set for each language. We use CSQA to test the zero-shot capability of FURINA$_{\text{Indic}}$. We evaluate on F1 for NER and on accuracy for WSTP and CSQA. The results are shown in Table 4.

FURINA$_{\text{Indic}}$ outperforms Glot500-m on 8 out of 11 languages on the NER task, with a 0.1 slightly higher average score than Glot500-m. We see a similar small improvement on WSTP, where FURINA$_{\text{Indic}}$ beats Glot500-m on most of the languages except for Panjabi. We assume the reason for the small improvement in these two tasks is that the model is fine-tuned on the train set of **all** languages (not zero-shot crosslingual transfer), which could overshadow the benefits from TRANSLICO. Nevertheless, the general consistent improvement shows the effectiveness of TRANSLICO. For the zero-shot task CSQA, we see a large increase. FURINA$_{\text{Indic}}$ improves the average performance by more than 12 points (from 34.0 to 46.4). Although another competitive model, ALBERT$_{US}$ (Moosa et al., 2023), beats Glot500-m (source model of FURINA$_{\text{Indic}}$), FURINA$_{\text{Indic}}$ outperforms ALBERT$_{US}$ consistently in all languages. This indicates that TRANSLICO enjoys the largest improvements in zero-shot scenarios, which is exactly the desideratum for most low-resource languages. Overall, the consistent improvement demonstrates TRANSLICO not only works well "globally" on all languages but also "locally" on a small group of languages that have lexical overlap but use different scripts.

6 Conclusion

In this work, we propose a novel framework TRANSLICO to fine-tune an mPLM only using a small portion of sentences in its pretraining corpus and their Latin transliteration. The framework contrasts the sentences in their original script and their transliterations to tackle the script barrier problem. Using Glot500-m as our source model, we fine-tune it using the proposed framework and present the resulting model: FURINA. Through extensive experiments, we show FURINA better aligns the representations from different scripts into a common space and therefore outperforms Glot500-m on a wide range of crosslingual transfer tasks. In addition, we conduct a case study on Indic group languages that are known to be mutually influenced by each other but use different scripts. We show TRANSLICO can also boost the performance for the Indic group languages. We hope this framework can inspire more future work leveraging transliteration to improve crosslinguality of mPLMs.

Limitations

We propose a simple contrastive learning framework TRANSLICO that aims to address the script barrier. We show the effectiveness by using Glot500-m as the source model and fine-tuning it on a small portion (5%) of its pretraining data. We would assume the proposed framework can also be used directly for pretraining, through which the model might benefit further from seeing more data. However, due to a limited computation budget, we aren't able to pretrain a model from scratch or continued pretrain a model using the full Glot500-c corpus. We would leave out how the proposed framework can be integrated into efficient pretraining or continued pretraining for future work.

We see the proposed framework TRANSLICO works "globally" when fine-tuning the model on data of all languages scripts, and also "locally" for a case study when only fine-tuning on the Indic-group languages that are mutually influenced and have extensive lexical overlap. Unfortunately, we didn't validate the framework further by trying more language groups, which could further demonstrate the usage of TRANSLICO. However, this is beyond the scope of this paper. Nevertheless, we hope our framework can inspire more work that applies a similar framework and focus "locally" on groups of languages of interest for future research.

Acknowledgements

This research was supported by DFG (grant SCHU 2246/14-1) and The European Research Council (NonSequeToR, grant #740516). We appreciate Ayyoob Imani's suggestions for designing a case study and Lixi Liu's suggestions for the pairwise similarity graph. We also thank the anonymous reviewers for their constructive feedback.

References

Ife Adebara, AbdelRahim Elmadany, Muhammad Abdul-Mageed, and Alcides Alcoba Inciarte. 2023. SERENGETI: Massively multilingual language models for Africa. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1498–1537, Toronto, Canada. Association for Computational Linguistics.

Chantal Amrhein and Rico Sennrich. 2020. On Romanization for model transfer between scripts in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2461–2469, 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.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Tyler Chang, Zhuowen Tu, and Benjamin Bergen. 2022. The geometry of multilingual language model representations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 119–136, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2021. InfoXLM: An information-theoretic framework for cross-lingual language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3576–3588, Online. Association for Computational Linguistics.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*, pages 539–546. IEEE Computer Society.

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.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Marta R Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, et al. 2022. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*.

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

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.

Tejas Dhamecha, Rudra Murthy, Samarth Bharadwaj, Karthik Sankaranarayanan, and Pushpak Bhattacharyya. 2021. Role of Language Relatedness in Multilingual Fine-tuning of Language Models: A Case Study in Indo-Aryan Languages. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8584–8595, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mozhdeh Gheini and Jonathan May. 2019. A universal parent model for low-resource neural machine translation transfer. *arXiv preprint arXiv:1909.06516*.

Vikrant Goyal, Sourav Kumar, and Dipti Misra Sharma. 2020. Efficient neural machine translation for low-resource languages via exploiting related languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 162–168, Online. Association for Computational Linguistics.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, pages 1735–1742. IEEE Computer Society.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. Computer Vision Foundation / IEEE.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Ayyoob ImaniGooghari, 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.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Yixin Ji, Jikai Wang, Juntao Li, Hai Ye, and Min Zhang. 2023. Isotropic representation can improve zero-shot cross-lingual transfer on multilingual language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8104–8118, Singapore. Association for Computational Linguistics.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Amir Hossein Kargaran, François Yvon, and Hinrich Schütze. 2024. GlotScript: A resource and tool for low resource writing system identification. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7774–7784, Torino, Italy. ELRA and ICCL.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fenfei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, Xiaodong Fan, Ruofei Zhang, Rahul Agrawal, Edward Cui, Sining Wei, Taroon Bharti, Ying Qiao, Jiun-Hung Chen, Winnie Wu, Shuguang Liu, Fan Yang, Daniel Campos, Rangan Majumder, and Ming Zhou. 2020. XGLUE: A new benchmark dataset for cross-lingual pre-training, understanding and generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6008–6018, Online. Association for Computational Linguistics.

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

Chunlan Ma, Ayyoob ImaniGooghari, Haotian Ye, Ehsaneddin Asgari, and Hinrich Schütze. 2023.

Taxi1500: A multilingual dataset for text classification in 1500 languages. *arXiv preprint arXiv:2305.08487*.

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. COCO-LM: correcting and contrasting text sequences for language model pretraining. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 23102–23114.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

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.

Nikitha Murikinati, Antonios Anastasopoulos, and Graham Neubig. 2020. Transliteration for cross-lingual morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 189–197, Online. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Lin Pan, Chung-Wei Hang, Haode Qi, Abhishek Shah, Saloni Potdar, and Mo Yu. 2021a. Multilingual BERT post-pretraining alignment. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 210–219, Online. Association for Computational Linguistics.

Xiao Pan, Mingxuan Wang, Liwei Wu, and Lei Li. 2021b. Contrastive learning for many-to-many multilingual neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 244–258, Online. 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.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077, Singapore. Association for Computational Linguistics.

Sukannya Purkayastha, Sebastian Ruder, Jonas Pfeiffer, Iryna Gurevych, and Ivan Vulić. 2023. Romanization-based large-scale adaptation of multilingual language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7996–8005, Singapore. Association for Computational Linguistics.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR.

Xiangpeng Wei, Rongxiang Weng, Yue Hu, Luxi Xing, Heng Yu, and Weihua Luo. 2021. On learning universal representations across languages. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Hans H Wellisch, Richard Foreman, Lee Breuer, and Robert Wilson. 1978. The conversion of scripts, its nature, history, and utilization.

Andrea Wen-Yi and David Mimno. 2023. Hyperpolyglot LLMs: Cross-lingual interpretability in token embeddings. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1124–1131, Singapore. Association for Computational Linguistics.

Qiyu Wu, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, and Daxin Jiang. 2022a. PCL: Peer-contrastive learning with diverse augmentations for unsupervised sentence embeddings. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12052–12066, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xing Wu, Chaochen Gao, Yipeng Su, Jizhong Han, Zhongyuan Wang, and Songlin Hu. 2022b. Smoothed contrastive learning for unsupervised sentence embedding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4902–4906, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Jiahao Xu, Wei Shao, Lihui Chen, and Lemao Liu. 2023a. SimCSE++: Improving contrastive learning for sentence embeddings from two perspectives. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12028–12040, Singapore. Association for Computational Linguistics.

Lingling Xu, Haoran Xie, Zongxi Li, Fu Lee Wang, Weiming Wang, and Qing Li. 2023b. Contrastive learning models for sentence representations. *ACM Trans. Intell. Syst. Technol.*, 14(4):67:1–67:34.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. ConSERT: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5065–5075, Online. Association for Computational Linguistics.

Junlei Zhang, Zhenzhong Lan, and Junxian He. 2023. Contrastive learning of sentence embeddings from scratch. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3932, Singapore. Association for Computational Linguistics.

Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method by mutual information maximization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1601–1610, Online. Association for Computational Linguistics.

Yuhao Zhang, Hongji Zhu, Yongliang Wang, Nan Xu, Xiaobo Li, and Binqiang Zhao. 2022. A contrastive framework for learning sentence representations from pairwise and triple-wise perspective in angular space. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4892–4903, Dublin, Ireland. Association for Computational Linguistics.

A Hyperparameters

A.1 Fine-tuning on Glot500-c

We fine-tune Glot500-m (ImaniGooghari et al., 2023) on a small portion of its training data, Glot500-c. Specifically, from each language-script, we randomly select 5% sentences. Note that we also consider languages that use Latin scripts when constructing our training data (another variant is only considering languages that do not use Latin scripts, which we do for the ablation study described in §5.1). Then we construct paired data by generating the Latin transliterations for each sentence using `Uroman` (Hermjakob et al., 2018). The paired data is then used to fine-tune the model using the proposed contrastive learning framework. For the MLM objective, we use the standard mask rate of 15% for both sentences in their original scripts and their `Uroman` transliterations. The weights for all objectives are set to 1 by default. We use Adam optimizer (Kingma and Ba, 2015) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon =$ 1e-6. The initial learning rate is set to 1e-5. The effective batch size is set to 768. Each batch contains sentence pairs (one in its original script and one in Latin transliteration) randomly picked from all languages. We set the per-GPU batch to 24, the gradient accumulation to 8, and train on four RTX A6000 GPUs ($24 \times 8 \times 4 = 768$). We use FP16 training (mixed precision (Micikevicius et al., 2018)) by default. We store checkpoints for the model every 2K steps and apply early stopping with the best average performance on downstream tasks. The model is fine-tuned for a maximum of 3 days (roughly 1 epoch).

	indo1319	atla1278	aust1307	turk1311	sino1245	maya1287	afro1255	other	all
SR-B	93	69	55	23	23	15	12	79	369
SR-T	54	2	7	7	3	0	5	20	98
Taxi1500	87	68	51	18	22	15	11	79	351
NER	94	5	12	12	7	0	6	28	164
POS	54	2	4	5	3	1	6	16	91

Table 6: The number of languages in each language family in downstream tasks.

	#class	measure (%)
SR-B	-	top-10 Acc.
SR-T	-	top-10 Acc.
Taxi1500	6	F1 score
NER	7	F1 score
POS	18	F1 score

Table 7: Information of downstream tasks. #class: the number of the categories if it is a sequence-level or token-level classification task.

	Latn	Cyrl	Hani	Arab	Deva	other	all
SR-B	290	28	4	11	8	28	369
SR-T	64	10	3	5	2	14	98
Taxi1500	281	25	4	8	7	26	351
NER	104	17	4	10	5	24	164
POS	57	8	3	5	3	15	91

Table 8: The number of languages in each script group in downstream tasks.

A.2 Fine-tuning on Downstream tasks

The basic information of each downstream task dataset is shown in Table 7. The number of languages in language families and script groups for each downstream task is shown in Table 6 and 8 respectively. We introduce the detailed hyperparameters settings in the following.

For sequence-level retrieval tasks, i.e., **SR-B** and **SR-T**, we use English-aligned sentences (up to 500 and 1000 for SR-B and SR-T respectively) from languages that are supported by the model. Different from SR-T, most of the languages in SR-B are low-resource languages. In addition, many languages in SR-B use non-Latin scripts. The retrieval task is performed without any training: we directly use the model to encode all sentences, where each sentence is represented as the average of the contextual embedding at the **8th** layer. We then compute the top-10 accuracy for each pair.

For sequence-level classification tasks, i.e., **Taxi1500**, we fine-tune the model with a 6-classes classification head on the English train set and select the best checkpoint using the English development set. We train a model using Adam optimizer for 40 epochs with early stopping. The learning rate is set to 1e-5 and the effective batch size is set to 16 (batch size of 8 and gradient accumulation of 2). The training is done on a single GTX 1080 Ti GPU. We then evaluate the performance in a zero-shot transfer setting by evaluating the fine-tuned model on the test sets of all other languages. The Macro F1 score is reported for each language.

For token-level classification tasks, i.e., **NER** and **POS**, we fine-tune the model with a suitable classification head (7 for NER and 18 for POS) on the English train set and select the best checkpoint using the English development set. We train each model using Adam optimizer for a maximum of 10 epochs with early stopping. The learning rate is set to 2e-5 and the effective batch size is set to 32 (batch size of 8 and gradient accumulation of 4). The training is done on a single GTX 1080 Ti GPU. We then evaluate the performance in a zero-shot transfer setting by evaluating the fine-tuned model on the test sets of all other languages. The Macro F1 score is reported for each language.

A.3 Fine-tuning on Indic Group

We fine-tune Glot500-m (ImaniGooghari et al., 2023) on a part of indic group languages. Specifically, we randomly sample 10% of sentences from Glot500-c (ImaniGooghari et al., 2023) for 12 indic languages (shown in table 5). Then we create paired data by transliterating each sentence into Latin using `Uroman` (Hermjakob et al., 2018). The other settings are the same in Appendix A.1.

A.4 Evaluation on Indic Group

The information of three downstream tasks WSTP, NER and CSQA is represented in Table 10. The three tasks are following the same setting of Moosa et al. (2023). For the sentence classification task WSTP, we fine-tune the model with 4-class head on 11 indic languages all at once. We train the model using Adam optimizer for 20 epochs. The learning rate is set to 2e-5 and the effective batch size is set to 256 (batch size 0f 64 and gradient accumulation of 4). The training is done on four GTX 1080 Ti GPUs. We then evaluate the performance in a

	SR-B			SR-T			Taxi1500			NER			POS		
	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA	XLM-R	Glot500	FURINA
indo1319	41.9	61.6	**71.5**	63.4	75.6	**77.5**	48.4	61.4	**67.7**	61.0	66.0	**67.5**	75.4	78.0	**78.7**
atla1278	5.5	45.2	**56.3**	29.6	50.2	**52.6**	13.3	48.2	**58.3**	46.5	59.9	**60.6**	24.1	60.1	**62.6**
aust1307	14.5	47.2	**61.6**	35.3	51.0	**52.3**	23.4	56.0	**62.9**	49.7	**57.6**	56.8	70.1	74.6	**75.8**
turk1311	22.3	63.3	**71.3**	41.6	**70.2**	65.8	30.9	62.2	**67.0**	50.7	61.9	**63.3**	57.3	72.2	**73.0**
sino1245	9.0	39.2	**46.9**	62.0	**80.5**	47.7	21.9	57.4	**61.7**	26.4	**37.4**	35.4	22.2	**35.5**	18.2
maya1287	3.8	20.3	**39.6**	-	-	-	11.1	47.8	**56.1**	-	-	-	28.7	62.0	**64.3**
afro1255	13.0	34.3	**47.0**	41.4	**53.0**	40.0	19.3	41.4	**48.5**	47.5	54.0	**58.1**	54.0	**67.2**	66.3
Other	14.1	36.9	**46.2**	56.7	**69.4**	64.3	20.9	50.9	**56.0**	50.9	56.3	**57.5**	54.1	60.8	**61.4**
All	19.3	47.2	**58.1**	56.6	**70.7**	68.8	26.7	54.3	**61.0**	55.3	61.6	**62.8**	65.6	71.8	**71.9**

Table 9: Aggregated performance of FURINA and baselines for each major language family. We report the average performance for language families: **indo1319** (Indo-European), **atla1278** (Atlantic-Congo), **aust1307** (Austronesian), **turk1311** (Turkic), **sino1245** (Sino-Tibetan), **maya1287** (Mayan), and **afro1255** (Afro-Asiatic). We collect the remaining languages in the group "**Other**". In addition, we also report the average over all languages (group "**All**"). **Bold** (underlined): best (second-best) result for each task in each group. FURINA generally performs better than other baselines except on Sino-Tibetan and SR-T.

	|lan|	|rows|	#class	measure (%)
WSTP	11	403k	4	Accuracy
NER	11	119k	7	F1 score
CSQA	9	135k	4	Accuracy

Table 10: Information of downstream tasks on Indic Group languages. |lan|: languages we evaluate from IndicGlue; #class: the number of the categories if it is a sequence-level or token-level classification task.

	|L|	Glot500-m is better	FURINA is better
SR-B	369	33	**336**
SR-T	98	43	**55**
Taxi1500	351	46	**305**
NER	164	55	**109**
POS	91	40	**51**

Table 11: Number of languages in each task that benefits from the proposed TRANSLICO framework. |L| is the total number of languages for each task.

crosslingual transfer setting by evaluating the fine-tuned model on the test set of 11 indic languages. The accuracy is reported for each language.

For token level task NER, we fine-tune the model with a 7 classification head on 11 indic languages all at once with 20 epochs. The learning rate is set to 2e-5 and the effective batch size is set to 32 (batch size 0f 8 and gradient accumulation of 4). The training is done on a single GTX 1080 Ti GPU. We then evaluate the performance in a crosslingual transfer setting by evaluating the fine-tuned model on the test set of 11 indic languages. The Macro F1 score is reported for each language.

B Futher Fine-grained Analysis

To further analyze how TRANSLICO can influence the crosslinguality of the multilingual model, we additionally report the aggregated results for each language family in Table 9 and the number of languages that benefit from TRANSLICO in Table 11. We see similar improvement as we observe for different script groups.

C Complete Results

We report the complete results for all tasks and language-scripts in Table 17, 18 (**SR-B**), Table 19 (**SR-T**), Table 20, 21 (**Taxi1500**), Table 22 (**NER**), and Table 23 (**POS**).

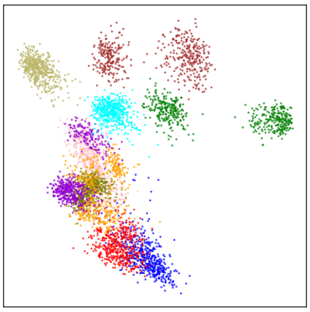
D Evaluation on Transliterated Data

We evaluate both Glot500m and FURINA under the common script scenario. Specifically, we transliterate all the data (including data of language written in Latin script, which is consistent with how we fine-tune the model with TRANSLICO) from downstream tasks to Latin script. Per-task performance for each script group is shown in Table 12 (**SR-B**), Table 13 (**SR-T**), Table 14 (**Taxi1500**), Table 15 (**NER**) and Table 16 (**POS**).

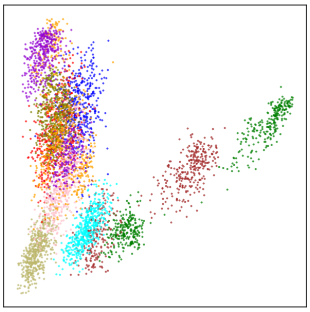
The results indicate that the models consistently perform better when the languages are in their original script instead of in Latin (the common script). We hypothesize the major reason is that the models are not being manipulated with vocabulary extension for transliterated data. Nevertheless, we observe from the results that TRANSLICO-uni generally outperforms Glot500m-uni across all tasks. This indicates our TRANSLICO framework is also effective for common script scenarios.

E Representation Visualization

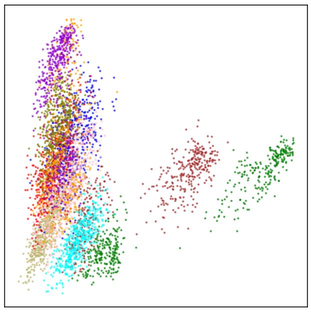
We visualized sentence representations from all layers of two models using PCA. Figure 5 and Figure 4 present the sentence representations of Glot500-m and FURINA respectively.



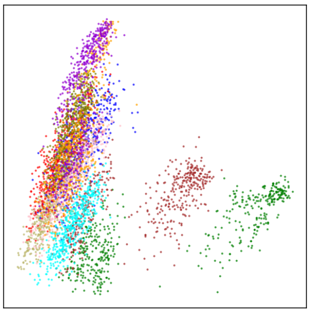
(a) layer 1



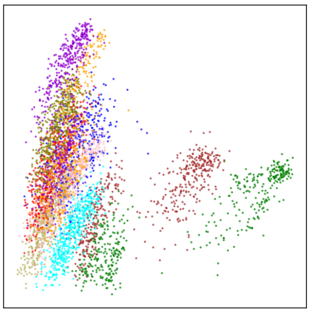
(b) layer 2



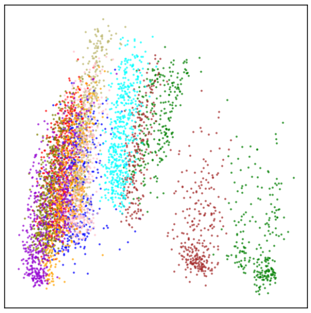
(c) layer 3



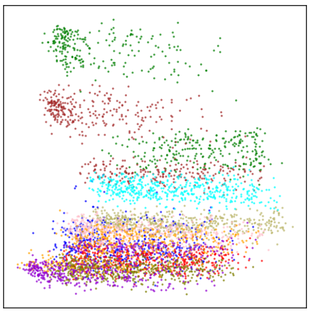
(d) layer 4



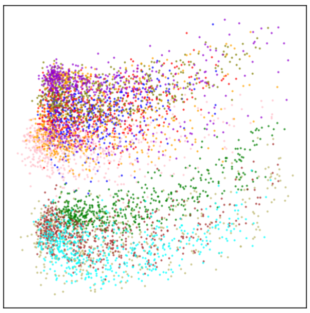
(e) layer 5



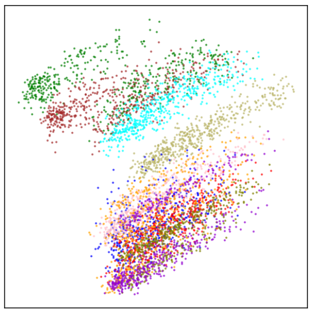
(f) layer 6



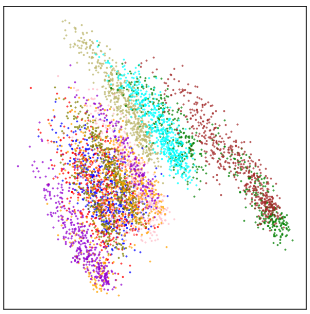
(g) layer 7



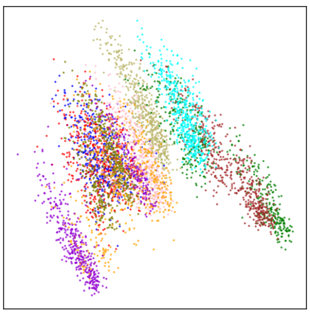
(h) layer 8



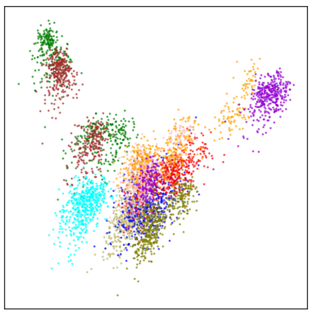
(i) layer 9



(j) layer 10

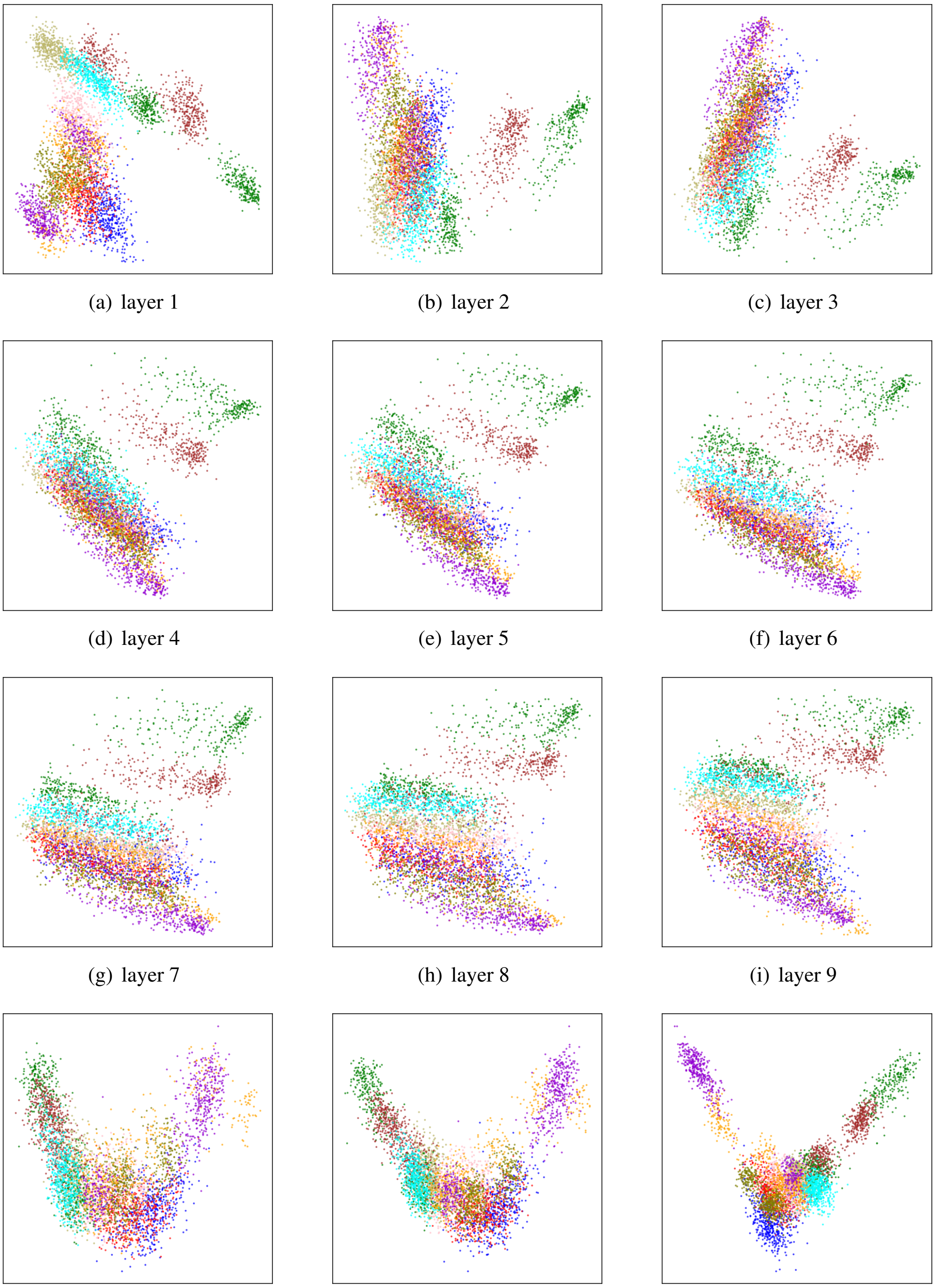


(k) layer 11



(l) layer 12

Figure 4: Visualizations of sentence representations from all layers (mean-pooling the contextualized token embeddings) of FURINA. The original dimension is 768 and we use PCA to select the first two principal components. Each point corresponds to a sentence. Different colors indicate distinct scripts.



(a) layer 1 (b) layer 2 (c) layer 3

(d) layer 4 (e) layer 5 (f) layer 6

(g) layer 7 (h) layer 8 (i) layer 9

(j) layer 10 (k) layer 11 (l) layer 12

Figure 5: Visualizations of sentence representations from all layers (mean-pooling the contextualized token embeddings) of Glot500-m. The original dimension is 768 and we use PCA to select the first two principal components. Points indicate sentence representations and different colors indicate distinct scripts.

	Glot500-m-uni	Glot500-m	FURINA-uni	FURINA
Latn	38.7	45.1	<u>53.8</u>	**57.4**
Cyrl	19.2	<u>60.3</u>	47.0	**69.0**
Hani	5.7	**43.4**	10.4	<u>39.8</u>
Arab	5.9	<u>56.4</u>	22.9	**61.4**
Deva	9.9	<u>60.3</u>	32.7	**66.8**
Other	5.4	<u>49.0</u>	18.0	**53.6**
All	32.7	47.2	<u>48.7</u>	**58.1**

Table 12: Performance Glot500-m and FURINA on the original and transliterated (into Latin) evaluation dataset of **SR-B**. We use Glot500-m and FURINA (resp. Glot500-m-uni and FURINA-uni) to refer to the model performing evaluation on the original-script (resp. transliterated) dataset. We group the performance by scripts (for Glot500-m-uni and FURINA-uni, the script indicates the original script of the languages).

	Glot500-m-uni	Glot500-m	FURINA-uni	FURINA
Latn	58.7	<u>69.1</u>	68.8	**73.0**
Cyrl	28.0	**74.4**	50.4	<u>69.7</u>
Hani	4.5	**80.5**	6.2	<u>47.7</u>
Arab	6.7	**71.8**	16.1	<u>56.3</u>
Deva	16.0	**81.8**	37.4	<u>71.9</u>
Other	7.0	**71.1**	15.7	<u>57.6</u>
All	43.0	**70.7**	54.1	<u>68.8</u>

Table 13: Performance Glot500-m and FURINA on the original and transliterated (into Latin) evaluation dataset of **SR-T**. We use Glot500-m and FURINA (resp. Glot500-m-uni and FURINA-uni) to refer to the model performing evaluation on the original-script (resp. transliterated) dataset. We group the performance by scripts (for Glot500-m-uni and FURINA-uni, the script indicates the original script of the languages).

	Glot500-m-uni	Glot500-m	FURINA-uni	FURINA
Latn	50.5	<u>52.6</u>	48.4	**59.8**
Cyrl	29.6	<u>59.8</u>	30.6	**63.6**
Hani	6.9	<u>68.2</u>	5.2	**70.1**
Arab	15.4	<u>60.8</u>	15.6	**66.5**
Deva	21.6	<u>66.6</u>	24.1	**73.2**
Other	11.5	<u>59.5</u>	14.7	**65.2**
All	44.2	<u>54.3</u>	42.9	**61.0**

Table 14: Performance Glot500-m and FURINA on the original and transliterated (into Latin) evaluation dataset of **Taxi1500**. We use Glot500-m and FURINA (resp. Glot500-m-uni and FURINA-uni) to refer to the model performing evaluation on the original-script (resp. transliterated) dataset. We group the performance by scripts (for Glot500-m-uni and FURINA-uni, the script indicates the original script of the languages).

	Glot500-m-uni	Glot500-m	FURINA-uni	FURINA
Latn	64.3	66.1	<u>66.2</u>	**67.3**
Cyrl	49.5	<u>65.3</u>	57.5	**66.2**
Hani	10.6	**22.2**	10.0	<u>21.9</u>
Arab	14.5	<u>53.4</u>	21.2	**57.7**
Deva	14.0	<u>56.2</u>	29.1	**58.9**
Other	16.6	**50.4**	24.6	**50.4**
All	49.9	<u>61.6</u>	54.0	**62.8**

Table 15: Performance Glot500-m and FURINA on the original and transliterated (into Latin) evaluation dataset of **NER**. We use Glot500-m and FURINA (resp. Glot500-m-uni and FURINA-uni) to refer to the model performing evaluation on the original-script (resp. transliterated) dataset. We group the performance by scripts (for Glot500-m-uni and FURINA-uni, the script indicates the original script of the languages).

	Glot500-m-uni	Glot500-m	FURINA-uni	FURINA
Latn	70.0	<u>74.4</u>	72.5	**75.7**
Cyrl	51.8	<u>79.3</u>	63.1	**79.5**
Hani	22.7	**35.5**	<u>23.4</u>	18.2
Arab	28.1	<u>68.8</u>	46.0	**69.3**
Deva	33.7	<u>59.8</u>	46.4	**60.8**
Other	32.5	**68.8**	41.2	<u>67.1</u>
All	57.1	<u>71.8</u>	62.6	**71.9**

Table 16: Performance Glot500-m and FURINA on the original and transliterated (into Latin) evaluation dataset of **POS**. We use Glot500-m and FURINA (resp. Glot500-m-uni and FURINA-uni) to refer to the model performing evaluation on the original-script (resp. transliterated) dataset. We group the performance by scripts (for Glot500-m-uni and FURINA-uni, the script indicates the original script of the languages).

Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA
ace_Latn	4.4	53.4	**72.0**	ach_Latn	4.4	40.0	**50.0**	acr_Latn	2.6	25.4	**48.2**
afr_Latn	76.8	69.4	**81.4**	agw_Latn	5.8	36.0	**52.8**	ahk_Latn	3.0	3.2	**8.0**
aka_Latn	5.0	**57.0**	53.4	aln_Latn	67.8	67.6	**73.0**	als_Latn	51.4	55.8	**56.4**
alt_Cyrl	12.6	50.8	**59.2**	alz_Latn	4.6	34.6	**44.6**	amh_Ethi	35.4	**52.8**	43.8
aoj_Latn	5.0	20.4	**32.8**	arb_Arab	7.0	14.6	**35.4**	arn_Latn	4.8	28.4	**36.4**
ary_Arab	2.8	15.2	**34.2**	arz_Arab	5.4	24.8	**45.4**	asm_Beng	26.2	66.6	**73.4**
ayr_Latn	4.8	52.8	**63.8**	azb_Arab	7.4	72.4	**74.2**	aze_Latn	71.0	73.0	**76.0**
bak_Cyrl	5.4	65.2	**70.0**	bam_Latn	3.4	60.2	**65.0**	ban_Latn	9.0	33.0	**61.0**
bar_Latn	13.4	40.8	**67.0**	bba_Latn	3.8	36.8	**43.0**	bbc_Latn	7.8	57.2	**71.4**
bci_Latn	4.4	13.2	**33.2**	bcl_Latn	10.2	79.8	**85.8**	bel_Cyrl	67.2	55.8	**69.6**
bem_Latn	6.6	58.2	**59.0**	ben_Beng	46.4	52.6	**71.8**	bhw_Latn	4.4	47.8	**56.0**
bim_Latn	4.2	52.2	**63.4**	bis_Latn	7.0	48.6	**65.6**	bod_Tibt	2.0	33.2	**52.4**
bqc_Latn	3.4	**38.8**	33.8	bre_Latn	17.6	32.8	**61.4**	bts_Latn	6.0	56.4	**70.8**
btx_Latn	11.0	59.6	**71.4**	bul_Cyrl	81.2	76.4	**85.8**	bum_Latn	4.8	38.0	**45.0**
bzj_Latn	7.8	75.0	**84.4**	cab_Latn	5.8	17.4	**28.4**	cac_Latn	3.6	14.8	**35.0**
cak_Latn	3.4	21.4	**43.6**	caq_Latn	3.2	30.2	**45.6**	cat_Latn	**86.6**	76.4	84.2
cbk_Latn	31.8	54.6	**76.2**	cce_Latn	5.2	51.8	**68.2**	ceb_Latn	14.2	68.0	**81.4**
ces_Latn	**75.2**	58.0	71.6	cfm_Latn	4.6	46.8	**59.4**	che_Cyrl	3.4	14.0	**30.8**
chk_Latn	5.4	41.2	**59.2**	chv_Cyrl	4.6	56.2	**56.8**	ckb_Arab	4.0	47.2	**62.4**
cmn_Hani	39.2	**41.8**	31.4	cnh_Latn	4.8	55.6	**63.0**	crh_Cyrl	8.8	75.2	**76.0**
crs_Latn	7.4	80.6	**84.2**	csy_Latn	3.8	50.0	**64.4**	ctd_Latn	4.2	59.4	**63.6**
ctu_Latn	2.8	21.6	**35.6**	cuk_Latn	5.0	22.2	**40.6**	cym_Latn	38.8	42.4	**60.4**
dan_Latn	71.6	63.2	**76.2**	deu_Latn	78.8	66.6	**82.6**	djk_Latn	4.6	40.4	**56.8**
dln_Latn	5.2	66.4	**68.4**	dtp_Latn	5.4	24.2	**41.2**	dyu_Latn	4.2	50.2	**63.0**
dzo_Tibt	2.2	36.4	**52.0**	efi_Latn	4.4	54.0	**59.0**	ell_Grek	52.6	48.6	**56.6**
enm_Latn	39.8	66.0	**75.6**	epo_Latn	64.6	56.2	**75.0**	est_Latn	**72.0**	56.4	69.8
eus_Latn	26.2	23.0	**36.8**	ewe_Latn	4.6	49.0	**54.2**	fao_Latn	24.0	73.4	**82.2**
fas_Arab	78.2	**89.2**	71.6	fij_Latn	3.8	36.4	**43.8**	fil_Latn	60.4	72.0	**84.8**
fin_Latn	**75.6**	53.8	68.6	fon_Latn	2.6	33.4	**58.0**	fra_Latn	88.6	79.2	**90.8**
fry_Latn	27.8	44.0	**72.6**	gaa_Latn	3.8	47.0	**68.6**	gil_Latn	5.6	36.8	**52.6**
giz_Latn	6.2	41.0	**53.2**	gkn_Latn	4.0	32.2	**54.2**	gkp_Latn	3.0	20.6	**31.6**
gla_Latn	25.2	43.0	**59.4**	gle_Latn	35.0	40.0	**54.8**	glv_Latn	5.8	47.4	**58.4**
gom_Latn	6.0	42.8	**58.8**	gor_Latn	3.8	26.0	**43.0**	grc_Grek	17.4	**54.8**	47.6
guc_Latn	3.4	13.0	**21.8**	gug_Latn	4.6	36.0	**37.4**	guj_Gujr	53.8	**71.4**	70.0
gur_Latn	3.8	27.0	**45.6**	guw_Latn	4.0	59.4	**69.4**	gya_Latn	3.6	41.0	**51.0**
gym_Latn	3.6	18.0	**29.4**	hat_Latn	6.0	68.2	**81.2**	hau_Latn	28.8	54.8	**69.8**
haw_Latn	4.2	38.8	**61.6**	heb_Hebr	**25.0**	21.8	21.0	hif_Latn	12.2	39.0	**73.2**
hil_Latn	11.0	76.2	**89.2**	hin_Deva	67.0	**76.6**	75.4	hin_Latn	13.6	43.2	**64.6**
hmo_Latn	6.4	48.2	**60.0**	hne_Deva	13.4	75.0	**84.6**	hnj_Latn	2.8	54.2	**64.0**
hra_Latn	5.2	52.2	**58.0**	hrv_Latn	**79.8**	72.6	75.6	hui_Latn	3.8	28.0	**32.4**
hun_Latn	**76.4**	56.2	70.8	hus_Latn	3.6	17.6	**39.2**	hye_Armn	30.8	**75.2**	68.6
iba_Latn	14.4	66.0	**71.4**	ibo_Latn	5.0	30.4	**48.4**	ifa_Latn	4.4	39.2	**52.2**
ifb_Latn	4.8	36.6	**52.0**	ikk_Latn	3.0	50.6	**62.2**	ilo_Latn	6.2	55.0	**73.6**
ind_Latn	**82.6**	72.2	77.8	isl_Latn	62.6	66.0	**75.8**	ita_Latn	75.4	70.0	**79.2**
ium_Latn	3.2	24.8	**38.6**	ixl_Latn	4.0	18.4	**32.8**	izz_Latn	2.8	25.6	**45.8**
jam_Latn	6.6	67.8	**85.2**	jav_Latn	25.4	47.4	**68.0**	jpn_Jpan	**65.0**	64.2	62.2
kaa_Cyrl	17.6	73.8	**80.0**	kaa_Latn	9.2	43.4	**73.0**	kab_Latn	3.4	20.6	**35.6**
kac_Latn	3.6	26.4	**45.8**	kal_Latn	3.4	**23.2**	22.8	kan_Knda	51.2	48.6	**56.0**
kat_Geor	**54.2**	51.4	51.0	kaz_Cyrl	61.4	56.8	**70.2**	kbp_Latn	2.6	36.0	**49.6**
kek_Latn	5.0	26.4	**50.2**	khm_Khmr	28.4	47.2	**51.6**	kia_Latn	4.0	33.2	**49.8**
kik_Latn	3.2	53.4	**63.2**	kin_Latn	5.0	59.4	**69.0**	kir_Cyrl	54.8	66.6	**70.6**
kjb_Latn	4.0	29.6	**53.8**	kjh_Cyrl	11.0	53.8	**57.2**	kmm_Latn	4.8	42.4	**55.0**
kmr_Cyrl	4.0	42.4	**66.6**	kmr_Latn	35.8	63.0	**70.8**	knv_Latn	2.8	9.0	**21.2**
kor_Hang	**64.0**	61.2	48.2	kpg_Latn	5.2	51.8	**61.6**	krc_Cyrl	9.2	63.0	**72.8**
kri_Latn	2.8	62.8	**78.4**	ksd_Latn	7.0	42.6	**53.6**	kss_Latn	2.2	6.0	**13.4**
ksw_Mymr	1.6	31.8	**47.6**	kua_Latn	4.8	43.8	**54.4**	lam_Latn	4.6	27.4	**37.2**
lao_Laoo	31.4	49.6	**54.8**	lat_Latn	52.2	49.6	**57.0**	lav_Latn	**74.2**	58.8	68.0
ldi_Latn	5.4	25.2	**45.2**	leh_Latn	5.6	58.2	**67.4**	lhu_Latn	2.0	5.0	**14.6**
lin_Latn	6.6	65.4	**70.0**	lit_Latn	**74.4**	62.4	67.8	loz_Latn	6.8	49.2	**67.6**
ltz_Latn	9.8	73.8	**83.8**	lug_Latn	4.6	**49.4**	**49.4**	luo_Latn	6.4	40.8	**53.2**
lus_Latn	3.8	54.4	**66.0**	lzh_Hani	25.0	**63.4**	36.8	mad_Latn	7.6	44.4	**63.6**
mah_Latn	4.8	35.6	**50.6**	mai_Deva	6.4	59.2	**75.4**	mal_Mlym	49.4	**56.6**	41.8
mam_Latn	3.8	12.8	**30.2**	mar_Deva	66.2	**74.8**	61.0	mau_Latn	2.4	3.6	**7.8**
mbb_Latn	3.0	33.6	**50.4**	mck_Latn	5.2	57.4	**64.4**	mcn_Latn	6.0	39.2	**44.6**
mco_Latn	2.6	7.0	**18.6**	mdy_Ethi	2.8	31.6	**50.0**	meu_Latn	5.6	52.0	**61.2**
mfe_Latn	9.0	**78.6**	77.2	mgh_Latn	5.2	23.6	**55.0**	mgr_Latn	4.0	57.6	**64.6**
mhr_Cyrl	6.6	48.0	**55.0**	min_Latn	9.4	29.0	**54.6**	miq_Latn	4.4	47.4	**48.2**
mkd_Cyrl	76.6	74.8	**81.8**	mlg_Latn	29.0	**66.0**	64.8	mlt_Latn	5.8	50.4	**74.6**
mos_Latn	4.2	42.8	**46.2**	mps_Latn	3.2	21.6	**26.0**	mri_Latn	4.2	48.4	**72.4**
mrw_Latn	6.0	52.2	**61.8**	msa_Latn	40.0	40.6	**44.6**	mwm_Latn	2.6	35.8	**52.8**
mxv_Latn	3.0	8.8	**17.0**	mya_Mymr	20.2	29.4	**33.4**	myv_Cyrl	4.6	35.0	**62.8**
mzh_Latn	4.6	36.2	**49.4**	nan_Latn	3.2	13.6	**29.0**	naq_Latn	3.0	25.0	**39.2**
nav_Latn	2.4	11.2	**18.2**	nbl_Latn	9.2	53.8	**62.6**	nch_Latn	4.4	21.4	**46.6**
ncj_Latn	4.6	25.2	**49.6**	ndc_Latn	5.2	40.0	**55.4**	nde_Latn	13.0	53.8	**62.0**
ndo_Latn	5.2	48.2	**63.8**	nds_Latn	9.6	43.0	**69.2**	nep_Deva	35.6	58.6	**72.6**
ngu_Latn	4.6	27.6	**53.6**	nia_Latn	4.6	29.4	**44.0**	nld_Latn	78.0	71.8	**83.8**
nmf_Latn	4.6	**36.6**	35.6	nnb_Latn	3.6	42.0	**46.4**	nno_Latn	58.4	72.6	**80.0**
nob_Latn	82.6	79.2	**86.0**	nor_Latn	81.2	**86.2**	85.6	npi_Deva	50.6	76.6	**82.2**
nse_Latn	5.2	54.8	**68.0**	nso_Latn	6.0	57.0	**67.6**	nya_Latn	4.0	60.2	**64.6**

Table 17: Top-10 accuracy of baselines and FURINA on **SR-B** (Part I).

Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA
nyn_Latn	4.4	51.8	**60.4**	nyy_Latn	3.0	25.6	**40.6**	nzi_Latn	3.2	47.2	**56.4**
ori_Orya	42.6	56.0	**75.2**	ory_Orya	31.4	53.4	**63.8**	oss_Cyrl	4.2	54.8	**63.8**
ote_Latn	3.6	18.0	**40.0**	pag_Latn	8.0	61.2	**76.8**	pam_Latn	8.2	49.8	**77.0**
pan_Guru	43.2	48.8	**58.0**	pap_Latn	12.4	72.4	**79.8**	pau_Latn	4.4	29.8	**46.4**
pcm_Latn	13.6	66.8	**73.2**	pdt_Latn	9.2	68.6	**77.0**	pes_Arab	69.4	**80.6**	69.0
pis_Latn	6.4	57.2	**74.6**	pls_Latn	5.0	34.4	**56.2**	plt_Latn	26.6	59.8	**66.8**
poh_Latn	3.4	15.2	**33.4**	pol_Latn	**79.2**	63.8	78.6	pon_Latn	5.6	21.6	**36.2**
por_Latn	81.6	76.6	**84.4**	prk_Latn	3.6	49.8	**58.8**	prs_Arab	79.4	**88.8**	72.8
pxm_Latn	3.2	24.0	**40.8**	qub_Latn	4.6	**43.4**	40.6	quc_Latn	3.6	24.8	**46.2**
qug_Latn	4.8	50.8	**66.8**	quh_Latn	4.6	56.2	**62.2**	quw_Latn	6.2	49.2	**58.6**
quy_Latn	4.6	**61.4**	53.6	quz_Latn	4.8	**68.0**	65.4	qvi_Latn	4.4	46.8	**70.6**
rap_Latn	3.2	25.6	**41.4**	rar_Latn	3.2	26.6	**36.8**	rmy_Latn	6.8	34.6	**61.8**
ron_Latn	72.2	66.6	**77.4**	rop_Latn	4.6	46.0	**62.6**	rug_Latn	3.6	49.0	**64.4**
run_Latn	5.4	54.6	**65.0**	rus_Cyrl	75.8	71.2	**77.2**	sag_Latn	6.0	52.4	**61.6**
sah_Cyrl	6.2	45.8	**60.2**	san_Deva	13.8	27.2	**38.4**	san_Latn	4.6	9.8	**17.2**
sba_Latn	2.8	37.6	**58.2**	seh_Latn	6.4	74.6	**82.0**	sin_Sinh	44.8	45.0	**53.4**
slk_Latn	**75.2**	63.6	74.6	slv_Latn	63.6	51.8	**67.6**	sme_Latn	6.8	47.8	**55.2**
smo_Latn	4.4	36.0	**61.8**	sna_Latn	7.0	43.0	**58.8**	snd_Arab	52.2	66.6	**71.4**
som_Latn	22.2	33.0	**52.8**	sop_Latn	5.2	31.2	**54.0**	sot_Latn	6.0	52.2	**72.6**
spa_Latn	81.2	80.0	**84.4**	sqi_Latn	58.2	63.4	**70.8**	srm_Latn	4.0	32.4	**45.4**
srn_Latn	6.8	79.8	**81.4**	srp_Cyrl	83.0	81.2	**85.6**	srp_Latn	**85.0**	81.2	84.4
ssw_Latn	4.8	47.0	**58.4**	sun_Latn	22.4	43.0	**63.8**	suz_Deva	3.6	34.2	**44.8**
swe_Latn	79.8	78.0	**84.0**	swh_Latn	47.8	66.4	**74.6**	sxn_Latn	4.8	25.8	**49.6**
tam_Taml	42.8	52.0	**54.8**	tat_Cyrl	8.2	67.2	**71.4**	tbz_Latn	2.6	28.0	**29.2**
tca_Latn	2.4	15.4	**38.4**	tdt_Latn	6.2	62.2	**68.4**	tel_Telu	44.4	42.6	**47.2**
teo_Latn	5.8	26.0	**27.8**	tgk_Cyrl	4.6	71.2	**77.8**	tgl_Latn	61.0	78.6	**84.4**
tha_Thai	30.0	**45.4**	41.8	tih_Latn	5.2	51.6	**67.6**	tir_Ethi	7.4	43.4	**54.2**
tlh_Latn	7.8	72.4	**73.2**	tob_Latn	2.2	16.8	**31.8**	toh_Latn	4.0	47.2	**64.4**
toi_Latn	4.2	47.4	**58.0**	toj_Latn	4.2	15.6	**30.8**	ton_Latn	4.2	22.4	**44.0**
top_Latn	3.4	8.0	**17.0**	tpi_Latn	5.8	58.0	**68.4**	tpm_Latn	3.6	39.6	**40.6**
tsn_Latn	5.4	41.8	**62.0**	tso_Latn	5.6	50.8	**65.0**	tsz_Latn	5.6	27.0	**39.8**
tuc_Latn	2.6	31.4	**38.2**	tui_Latn	3.6	38.0	**47.2**	tuk_Cyrl	13.6	65.0	**70.2**
tuk_Latn	9.6	66.2	**71.8**	tum_Latn	5.2	**66.2**	64.2	tur_Latn	**74.4**	63.2	70.6
twi_Latn	3.8	**50.0**	48.0	tyv_Cyrl	6.8	46.6	**63.6**	tzh_Latn	6.0	25.8	**48.8**
tzo_Latn	3.8	16.6	**34.0**	udm_Cyrl	6.0	55.2	**59.2**	uig_Arab	45.8	56.2	**71.4**
uig_Latn	9.8	62.8	**72.6**	ukr_Cyrl	66.0	57.0	**71.4**	urd_Arab	47.6	65.0	**67.6**
uzb_Cyrl	6.2	78.8	**82.4**	uzb_Latn	54.8	67.6	**84.0**	uzn_Cyrl	5.4	**87.0**	84.8
ven_Latn	4.8	47.2	**47.6**	vie_Latn	**72.8**	57.8	61.4	wal_Latn	4.2	51.4	**54.8**
war_Latn	9.8	43.4	**78.2**	wbm_Latn	3.8	46.4	**49.4**	wol_Latn	4.6	35.8	**49.0**
xav_Latn	2.2	5.0	**10.4**	xho_Latn	10.4	40.8	**62.2**	yan_Latn	4.2	31.8	**38.8**
yao_Latn	4.4	55.2	**56.0**	yap_Latn	4.0	24.0	**42.0**	yom_Latn	4.8	42.2	**57.2**
yor_Latn	3.4	37.4	**51.6**	yua_Latn	3.8	18.2	**32.2**	yue_Hani	17.2	24.0	**55.8**
zai_Latn	6.2	38.0	**47.6**	zho_Hani	40.4	**44.4**	35.0	zlm_Latn	83.4	**87.0**	82.8
zom_Latn	3.6	50.2	**59.2**	zsm_Latn	**90.2**	83.0	85.0	zul_Latn	11.0	49.0	**56.6**

Table 18: Top-10 accuracy of baselines and FURINA on **SR-B** (Part II).

Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA
afr_Latn	71.9	81.1	**84.7**	amh_Ethi	35.1	**44.6**	44.0	ara_Arab	59.2	**64.2**	46.6
arz_Arab	32.5	**63.5**	40.3	ast_Latn	59.8	87.4	**88.2**	aze_Latn	62.6	79.9	**81.9**
bel_Cyrl	70.0	**81.4**	79.0	ben_Beng	54.1	69.4	**71.3**	bos_Latn	78.5	92.4	**94.1**
bre_Latn	10.3	19.9	**21.7**	bul_Cyrl	84.4	**86.7**	**86.7**	cat_Latn	72.8	78.7	**84.6**
cbk_Latn	33.2	49.4	**59.8**	ceb_Latn	15.2	41.3	**42.5**	ces_Latn	71.1	75.1	**78.2**
cmn_Hani	79.5	**85.6**	39.7	csb_Latn	21.3	40.3	**68.4**	cym_Latn	45.7	**55.7**	53.9
dan_Latn	91.9	91.5	**94.7**	deu_Latn	95.9	95.0	**96.5**	dtp_Latn	5.6	**21.1**	19.8
ell_Grek	76.2	**80.2**	73.5	epo_Latn	64.9	74.3	**82.8**	est_Latn	63.9	69.1	**76.2**
eus_Latn	45.9	52.7	**58.8**	fao_Latn	45.0	82.4	**88.5**	fin_Latn	**81.9**	72.3	72.7
fra_Latn	85.7	**86.0**	84.8	fry_Latn	60.1	75.1	**84.4**	gla_Latn	21.0	41.9	**46.0**
gle_Latn	32.0	50.8	**51.8**	glg_Latn	72.6	77.5	**83.9**	gsw_Latn	36.8	69.2	**73.5**
heb_Hebr	**76.3**	76.0	49.1	hin_Deva	73.8	**85.6**	76.0	hrv_Latn	79.6	**89.8**	88.9
hsb_Latn	21.5	53.6	**65.2**	hun_Latn	**76.1**	69.2	69.9	hye_Armn	64.6	**83.2**	66.2
ido_Latn	25.7	57.6	**76.6**	ile_Latn	34.6	75.6	**82.9**	ina_Latn	62.7	91.4	**93.4**
ind_Latn	84.3	**88.8**	86.4	isl_Latn	78.7	84.0	**87.2**	ita_Latn	81.3	86.4	**90.0**
jpn_Jpan	**74.4**	72.6	50.7	kab_Latn	3.7	16.4	**20.0**	kat_Geor	61.1	**67.7**	50.0
kaz_Cyrl	60.3	**72.3**	64.9	khm_Khmr	41.1	**52.4**	44.9	kor_Hang	73.4	**78.0**	56.2
kur_Latn	24.1	54.1	**60.2**	lat_Latn	33.6	42.8	**46.0**	lfn_Latn	32.5	59.3	**62.0**
lit_Latn	73.4	65.6	**74.2**	lvs_Latn	73.4	76.9	**81.7**	mal_Mlym	80.1	**83.8**	73.9
mar_Deva	63.5	**77.9**	67.8	mhr_Cyrl	6.5	**34.9**	30.0	mkd_Cyrl	70.5	**81.4**	77.5
mon_Cyrl	60.9	**77.0**	71.4	nds_Latn	28.8	77.1	**84.8**	nld_Latn	90.3	**91.8**	91.6
nno_Latn	70.7	87.8	**90.8**	nob_Latn	93.5	95.7	**97.0**	oci_Latn	22.9	46.9	**61.7**
pam_Latn	4.8	11.0	**14.3**	pes_Arab	83.3	**87.6**	73.7	pms_Latn	16.6	54.5	**67.8**
pol_Latn	82.6	82.4	**82.8**	por_Latn	91.0	90.1	**91.8**	ron_Latn	86.0	82.8	**88.1**
rus_Cyrl	89.6	**91.5**	84.9	slk_Latn	73.2	75.9	**81.8**	slv_Latn	72.1	77.0	**81.2**
spa_Latn	85.5	88.9	**89.5**	sqi_Latn	72.2	84.7	**88.6**	srp_Latn	78.1	90.0	**90.2**
swe_Latn	90.4	89.7	**92.2**	swh_Latn	30.3	44.1	**44.6**	tam_Taml	46.9	**66.4**	57.0
tat_Cyrl	10.3	**70.3**	61.3	tel_Telu	58.5	**67.9**	67.5	tgl_Latn	47.6	**77.1**	76.5
tha_Thai	56.8	**78.1**	35.6	tuk_Latn	16.3	63.5	**65.0**	tur_Latn	77.9	**78.4**	75.4
uig_Arab	38.8	**62.6**	50.7	ukr_Cyrl	77.1	**83.7**	80.0	urd_Arab	54.4	**80.9**	70.2
uzb_Cyrl	25.2	**64.5**	61.2	vie_Latn	85.4	**87.0**	73.7	war_Latn	8.0	26.2	**37.7**
wuu_Hani	56.1	**79.7**	51.6	xho_Latn	28.9	56.3	**60.6**	yid_Hebr	37.3	**74.4**	66.2
yue_Hani	50.3	**76.3**	51.7	zsm_Latn	81.4	**91.8**	88.8				

Table 19: Top-10 accuracy of baselines and FURINA on **SR-T**.

Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA
ace_Latn	13.4	65.6	**70.5**	ach_Latn	10.9	39.6	**55.9**	acr_Latn	8.8	51.7	**67.7**
afr_Latn	65.7	65.5	**67.0**	agw_Latn	13.9	**63.0**	60.6	ahk_Latn	**9.3**	8.9	9.1
aka_Latn	9.1	47.8	**55.7**	aln_Latn	53.8	59.3	**62.3**	als_Latn	57.8	**60.9**	**60.9**
alt_Cyrl	25.4	46.9	**50.4**	alz_Latn	11.8	32.4	**49.3**	amh_Ethi	9.3	**14.1**	10.1
aoj_Latn	12.2	45.2	**56.4**	arn_Latn	9.1	48.5	**50.8**	ary_Arab	14.5	39.4	**43.4**
arz_Arab	21.9	39.3	**45.6**	asm_Beng	47.3	60.9	**69.6**	ayr_Latn	7.7	60.7	**72.3**
azb_Arab	16.1	60.5	**67.1**	aze_Latn	64.6	68.1	**73.5**	bak_Cyrl	22.6	64.0	**71.6**
bam_Latn	7.7	58.0	**60.6**	ban_Latn	18.9	50.9	**56.0**	bar_Latn	34.1	51.5	**53.5**
bba_Latn	8.6	46.1	**52.6**	bci_Latn	8.4	28.2	**41.6**	bcl_Latn	31.5	52.7	**63.2**
bel_Cyrl	**62.0**	60.2	59.4	bem_Latn	15.8	49.2	**60.0**	ben_Beng	63.4	58.7	**66.9**
bhw_Latn	14.9	48.0	**50.7**	bim_Latn	9.1	49.8	**70.2**	bis_Latn	14.8	64.8	**77.5**
bqc_Latn	9.1	**40.1**	36.9	bre_Latn	30.3	38.6	**44.1**	btx_Latn	24.6	56.0	**59.7**
bul_Cyrl	69.2	66.8	**69.9**	bum_Latn	14.0	48.5	**53.3**	bzj_Latn	13.3	71.0	**73.3**
cab_Latn	8.0	27.7	**34.6**	cac_Latn	10.5	47.6	**61.4**	cak_Latn	10.7	60.3	**67.0**
caq_Latn	8.3	51.8	**53.3**	cat_Latn	**65.6**	53.9	59.8	cbk_Latn	51.8	58.6	**75.7**
cce_Latn	9.7	51.3	**63.6**	ceb_Latn	26.2	52.6	**60.6**	ces_Latn	**67.7**	57.9	55.7
cfm_Latn	9.1	69.2	**73.4**	che_Cyrl	11.4	24.0	**26.5**	chv_Cyrl	13.4	**64.7**	64.4
cmn_Hani	71.9	69.4	**74.2**	cnh_Latn	9.7	67.6	**72.8**	crh_Cyrl	14.7	62.7	**72.3**
crs_Latn	16.5	68.3	**73.5**	csy_Latn	11.8	65.7	**69.8**	ctd_Latn	9.4	64.0	**69.0**
ctu_Latn	13.0	53.9	**62.8**	cuk_Latn	14.2	40.2	**48.9**	cym_Latn	52.9	44.7	**53.5**
dan_Latn	**62.1**	60.3	62.0	deu_Latn	53.9	51.9	**61.0**	djk_Latn	14.7	**61.0**	60.1
dln_Latn	11.0	58.0	**67.2**	dtp_Latn	10.8	48.2	**69.9**	dyu_Latn	5.1	55.0	**59.3**
dzo_Tibt	4.9	**66.1**	65.1	efi_Latn	13.7	59.5	**64.9**	ell_Grek	46.6	66.5	**72.6**
eng_Latn	74.6	76.1	**78.8**	enm_Latn	57.5	**75.2**	74.1	epo_Latn	63.0	57.6	**64.3**
est_Latn	**67.1**	53.7	65.7	eus_Latn	22.7	27.0	**28.5**	ewe_Latn	7.3	45.8	**58.6**
fao_Latn	33.6	**66.5**	62.2	fas_Arab	68.7	73.6	**78.5**	fij_Latn	13.0	49.7	**58.9**
fil_Latn	53.7	54.4	**69.3**	fin_Latn	**60.0**	49.7	59.6	fon_Latn	6.2	50.0	**58.8**
fra_Latn	**74.8**	65.8	74.4	fry_Latn	40.1	42.0	**55.4**	gaa_Latn	5.0	**52.6**	51.4
gil_Latn	8.4	48.9	**59.6**	giz_Latn	9.0	53.6	**59.5**	gkn_Latn	9.7	45.8	**62.6**
gkp_Latn	6.0	41.7	**43.0**	gla_Latn	36.2	48.3	**54.0**	gle_Latn	40.1	40.0	**44.6**
glv_Latn	11.7	**48.4**	43.4	gom_Latn	13.0	**34.4**	32.5	gor_Latn	18.5	50.9	**63.5**
guc_Latn	8.7	36.4	**47.5**	gug_Latn	15.3	48.1	**56.6**	guj_Gujr	62.9	70.8	**74.7**
gur_Latn	7.4	44.3	**45.1**	guw_Latn	12.0	56.1	**63.9**	gya_Latn	5.0	50.8	**53.2**
gym_Latn	10.9	47.0	**58.7**	hat_Latn	14.5	59.1	**71.3**	hau_Latn	44.3	54.9	**64.6**
haw_Latn	9.0	**52.5**	47.9	heb_Hebr	17.9	35.3	**46.2**	hif_Latn	19.2	49.4	**53.7**
hil_Latn	33.8	**71.1**	70.9	hin_Deva	66.7	65.9	**71.5**	hmo_Latn	15.3	**65.6**	64.7
hne_Deva	41.0	**74.4**	72.8	hnj_Latn	15.2	69.9	**72.6**	hra_Latn	13.3	59.9	**65.6**
hrv_Latn	61.0	65.4	**73.1**	hui_Latn	9.3	55.4	**66.9**	hun_Latn	**75.5**	59.4	67.6
hus_Latn	10.7	47.9	**51.3**	hye_Armn	72.1	71.2	**77.2**	iba_Latn	40.7	62.4	**68.1**
ibo_Latn	8.0	65.3	**68.1**	ifa_Latn	12.5	52.2	**60.3**	ifb_Latn	8.9	**57.0**	50.4
ikk_Latn	9.5	56.5	**64.3**	ilo_Latn	20.0	58.7	**74.0**	ind_Latn	75.6	74.2	**76.6**
isl_Latn	60.3	50.8	**60.6**	ita_Latn	71.2	60.5	**75.9**	ium_Latn	7.4	61.5	**63.4**
ixl_Latn	12.6	42.2	**47.7**	izz_Latn	12.3	48.7	**58.9**	jam_Latn	18.0	56.2	**63.9**
jav_Latn	48.7	49.4	**61.9**	jpn_Jpan	**71.0**	61.5	68.3	kaa_Cyrl	16.7	**66.4**	65.9
kab_Latn	9.1	31.2	**38.4**	kac_Latn	11.3	56.5	**62.8**	kal_Latn	10.3	34.4	**40.2**
kan_Knda	69.9	65.6	**73.5**	kat_Geor	**66.6**	60.4	65.1	kaz_Cyrl	63.4	63.7	**65.3**
kbp_Latn	4.9	**42.1**	41.9	kek_Latn	7.7	49.7	**52.4**	khm_Khmr	63.6	68.6	**71.4**
kia_Latn	13.4	55.1	**66.2**	kik_Latn	6.4	46.2	**58.2**	kin_Latn	17.0	58.3	**64.5**
kir_Cyrl	61.4	65.7	**69.5**	kjb_Latn	8.8	53.5	**63.1**	kjh_Cyrl	21.6	**60.3**	58.0
kmm_Latn	9.1	60.9	**71.7**	kmr_Cyrl	9.5	45.2	**60.9**	knv_Latn	8.6	59.6	**62.7**
kor_Hang	**72.7**	62.5	72.4	kpg_Latn	10.6	**67.8**	66.6	krc_Cyrl	24.8	61.9	**70.8**
kri_Latn	10.8	65.7	**70.0**	ksd_Latn	12.7	60.1	**62.4**	kss_Latn	4.9	27.5	**34.2**
ksw_Mymr	4.9	**64.3**	58.6	kua_Latn	17.5	48.9	**56.1**	lam_Latn	12.8	35.4	**50.2**
lao_Laoo	73.5	74.1	**75.8**	lat_Latn	**65.9**	48.7	53.6	lav_Latn	69.9	65.0	**70.7**
ldi_Latn	13.7	25.5	**39.6**	leh_Latn	14.3	49.6	**66.4**	lhu_Latn	6.3	28.7	**29.6**
lin_Latn	12.7	54.7	**70.5**	lit_Latn	**65.1**	60.4	**65.1**	loz_Latn	13.8	51.5	**67.5**
ltz_Latn	27.2	53.5	**69.5**	lug_Latn	13.7	54.3	**63.7**	luo_Latn	10.6	45.6	**55.2**
lus_Latn	9.1	60.5	**67.2**	lzh_Hani	62.9	63.2	**70.8**	mad_Latn	24.6	63.6	**72.8**
mah_Latn	10.6	43.5	**47.9**	mai_Deva	30.5	62.7	**72.7**	mal_Mlym	10.5	5.7	**10.6**
mam_Latn	9.2	34.8	**40.9**	mar_Deva	60.7	67.1	**73.9**	mau_Latn	6.5	6.0	**8.1**
mbb_Latn	8.7	56.9	**64.9**	mck_Latn	18.2	52.0	**61.4**	mcn_Latn	10.7	41.5	**43.8**
mco_Latn	8.2	**26.5**	22.7	mdy_Ethi	4.9	56.2	**68.5**	meu_Latn	15.6	57.6	**66.5**
mfe_Latn	15.6	63.2	**77.3**	mgh_Latn	9.4	35.9	**50.3**	mgr_Latn	15.8	50.9	**66.3**
mhr_Cyrl	10.5	44.2	**49.1**	min_Latn	23.9	52.8	**63.2**	miq_Latn	5.2	56.2	**66.2**
mkd_Cyrl	74.4	76.0	**77.7**	mlg_Latn	38.3	51.5	**60.9**	mlt_Latn	14.7	53.6	**64.1**
mos_Latn	10.7	41.5	**57.3**	mps_Latn	11.6	62.5	**67.1**	mri_Latn	8.5	44.3	**63.3**
mrw_Latn	16.7	54.7	**56.2**	msa_Latn	43.5	51.6	**61.6**	mwm_Latn	6.7	61.6	**67.9**
mxv_Latn	11.7	20.4	**32.9**	mya_Mymr	50.0	61.0	**70.3**	myv_Cyrl	14.2	49.7	**52.5**
mzh_Latn	12.6	**51.9**	50.7	nan_Latn	6.4	29.3	**36.8**	naq_Latn	7.7	44.7	**52.4**
nav_Latn	6.9	**27.4**	26.0	nbl_Latn	20.2	54.3	**59.0**	nch_Latn	6.4	44.4	**55.6**
ncj_Latn	7.4	39.4	**52.4**	ndc_Latn	18.5	48.1	**56.0**	nde_Latn	20.2	54.3	**59.0**
ndo_Latn	16.1	44.8	**56.2**	nds_Latn	15.4	45.5	**55.3**	nep_Deva	65.9	67.9	**78.6**
ngu_Latn	10.9	52.4	**56.4**	nld_Latn	66.4	64.0	**68.1**	nmf_Latn	11.9	48.7	**51.9**
nnb_Latn	10.9	49.3	**63.7**	nno_Latn	59.4	**69.0**	66.0	nob_Latn	**67.9**	56.4	63.7
nor_Latn	67.1	54.8	**67.7**	npi_Deva	65.2	64.4	**81.2**	nse_Latn	15.7	46.3	**61.1**
nso_Latn	15.8	57.5	**71.9**	nya_Latn	16.0	65.2	**66.0**	nyn_Latn	15.6	38.7	**51.6**
nyy_Latn	8.1	37.0	**47.4**	nzi_Latn	6.5	38.7	**52.2**	ori_Orya	63.0	73.2	**73.4**
ory_Orya	61.8	71.9	**73.4**	oss_Cyrl	9.4	47.9	**60.3**	ote_Latn	5.5	41.3	**47.6**
pag_Latn	22.0	**61.1**	59.9	pam_Latn	25.8	48.5	**50.2**	pan_Guru	64.8	70.1	**74.7**

Table 20: F1 scores of baselines and FURINA on **Taxi1500** (Part I).

Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA
pap_Latn	36.3	60.9	**76.3**	pau_Latn	15.6	42.2	**49.7**	pcm_Latn	31.8	**70.6**	60.9
pdt_Latn	18.1	57.6	**66.3**	pes_Arab	72.6	**74.3**	73.3	pis_Latn	12.5	68.1	**69.1**
pls_Latn	16.2	54.9	**60.6**	plt_Latn	32.3	49.8	**59.4**	poh_Latn	12.7	**55.9**	55.7
pol_Latn	68.8	60.9	**76.7**	pon_Latn	7.9	62.7	**64.8**	por_Latn	73.4	62.6	**74.7**
prk_Latn	11.2	63.3	**67.2**	prs_Arab	74.4	69.8	**78.5**	pxm_Latn	11.5	44.4	**48.7**
qub_Latn	10.1	64.3	**70.6**	quc_Latn	15.3	54.3	**61.7**	qug_Latn	12.0	67.1	**73.6**
quh_Latn	12.1	70.1	**75.2**	quw_Latn	11.2	58.9	**65.9**	quy_Latn	11.1	71.4	**74.4**
quz_Latn	12.5	71.7	**76.5**	qvi_Latn	7.6	66.7	**71.3**	rap_Latn	5.4	61.3	**64.8**
rar_Latn	9.0	53.0	**64.1**	rmy_Latn	16.0	51.6	**57.7**	ron_Latn	67.0	58.6	**75.8**
rop_Latn	13.6	59.7	**70.1**	rug_Latn	6.2	54.0	**69.7**	run_Latn	17.7	50.9	**59.9**
rus_Cyrl	68.9	67.7	**75.5**	sag_Latn	11.9	50.7	**58.2**	sah_Cyrl	14.8	**68.0**	67.1
sba_Latn	7.9	**53.1**	52.5	seh_Latn	13.4	51.4	**61.5**	sin_Sinh	65.8	59.3	**73.0**
slk_Latn	**72.6**	65.2	62.8	slv_Latn	66.6	62.6	**75.6**	sme_Latn	12.3	**50.9**	44.6
smo_Latn	12.8	57.8	**68.6**	sna_Latn	14.4	38.9	**60.3**	snd_Arab	66.4	65.8	**72.8**
som_Latn	41.7	39.0	**46.0**	sop_Latn	12.7	39.0	**48.2**	sot_Latn	15.3	48.0	**70.7**
spa_Latn	74.0	64.7	**75.9**	sqi_Latn	74.4	71.5	**79.5**	srm_Latn	14.1	57.6	**68.9**
srn_Latn	15.9	69.7	**71.6**	srp_Latn	67.8	67.7	**73.4**	ssw_Latn	14.9	48.0	**59.6**
sun_Latn	52.9	49.8	**56.8**	suz_Deva	16.4	**63.8**	61.6	swe_Latn	74.6	64.9	**75.7**
swh_Latn	61.3	59.1	**67.5**	sxn_Latn	13.1	51.5	**56.7**	tam_Taml	62.9	66.2	**72.3**
tat_Cyrl	27.8	66.4	**67.7**	tbz_Latn	6.9	53.1	**53.2**	tca_Latn	9.4	**51.6**	51.3
tdt_Latn	15.9	68.9	**70.7**	tel_Telu	68.7	66.6	**69.4**	teo_Latn	14.2	29.4	**35.3**
tgk_Cyrl	9.8	66.6	**70.8**	tgl_Latn	53.7	54.4	**69.3**	tha_Thai	68.8	61.9	**69.5**
tih_Latn	12.8	57.4	**72.9**	tir_Ethi	19.5	54.0	**71.7**	tlh_Latn	35.0	**70.3**	65.7
tob_Latn	7.5	**59.5**	54.4	toh_Latn	15.4	39.5	**57.1**	toi_Latn	17.6	48.0	**60.5**
toj_Latn	14.5	39.1	**49.1**	ton_Latn	9.3	61.2	**63.0**	top_Latn	10.7	**21.0**	20.9
tpi_Latn	12.9	65.6	**70.4**	tpm_Latn	12.1	60.4	**62.0**	tsn_Latn	11.4	53.4	**57.4**
tsz_Latn	10.5	46.4	**61.0**	tuc_Latn	8.7	59.7	**68.2**	tui_Latn	8.6	48.5	**57.4**
tuk_Latn	21.1	54.1	**63.7**	tum_Latn	13.3	47.7	**70.4**	tur_Latn	66.1	67.4	**70.4**
twi_Latn	8.9	46.7	**54.0**	tyv_Cyrl	17.2	58.7	**63.8**	tzh_Latn	11.4	47.1	**58.7**
tzo_Latn	7.7	41.8	**50.8**	udm_Cyrl	12.6	**66.5**	61.6	ukr_Cyrl	**67.8**	63.1	67.5
urd_Arab	53.6	63.3	**72.8**	uzb_Latn	53.3	53.0	**73.2**	uzn_Cyrl	11.3	66.7	**71.7**
ven_Latn	10.9	45.6	**50.7**	vie_Latn	68.8	**72.7**	72.1	wal_Latn	17.4	43.0	**60.3**
war_Latn	21.9	48.0	**60.3**	wbm_Latn	10.8	62.4	**68.9**	wol_Latn	15.2	37.8	**43.9**
xav_Latn	10.3	**42.9**	39.5	xho_Latn	20.7	52.2	**58.4**	yan_Latn	11.1	53.1	**61.4**
yao_Latn	13.5	45.7	**68.5**	yap_Latn	10.6	48.1	**52.6**	yom_Latn	14.4	39.8	**48.6**
yor_Latn	14.6	51.6	**60.2**	yua_Latn	12.4	36.5	**51.3**	yue_Hani	60.1	**70.1**	64.8
zai_Latn	14.2	**50.1**	44.3	zho_Hani	**71.4**	70.1	70.4	zlm_Latn	73.9	73.5	**76.4**
zom_Latn	11.4	57.8	**73.7**	zsm_Latn	72.9	71.9	**73.6**	zul_Latn	25.9	58.6	**67.8**

Table 21: F1 scores of baselines and FURINA on **Taxi1500** (Part II).

Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA
ace_Latn	33.7	**43.2**	**43.2**	afr_Latn	75.7	74.9	**76.1**	als_Latn	61.8	**80.2**	76.0
amh_Ethi	**41.8**	41.7	40.0	ara_Arab	45.4	55.9	**66.0**	arg_Latn	73.7	75.8	**79.9**
arz_Arab	48.0	51.4	**59.0**	asm_Beng	53.3	**66.7**	65.5	ast_Latn	80.2	83.1	**85.3**
aym_Latn	36.0	44.7	**46.1**	aze_Latn	63.6	63.6	**67.7**	bak_Cyrl	36.6	**59.0**	58.6
bar_Latn	57.5	72.4	**75.7**	bel_Cyrl	73.2	73.9	**75.1**	ben_Beng	65.5	73.9	**74.9**
bih_Deva	50.0	**58.9**	57.1	bod_Tibt	0.0	**34.6**	33.5	bos_Latn	**74.5**	72.0	74.2
bre_Latn	59.5	62.7	**65.1**	bul_Cyrl	77.2	**77.7**	77.1	cat_Latn	81.8	83.7	**84.0**
cbk_Latn	52.9	**54.3**	51.4	ceb_Latn	**54.9**	48.8	51.7	ces_Latn	**77.7**	77.5	77.5
che_Cyrl	15.3	**60.9**	56.3	chv_Cyrl	58.7	76.9	**79.3**	ckb_Arab	33.7	**74.9**	**74.9**
cos_Latn	56.5	54.6	**58.0**	crh_Latn	40.7	52.1	**56.5**	csb_Latn	54.1	56.5	**57.6**
cym_Latn	58.4	**62.5**	61.1	dan_Latn	81.1	80.8	**81.9**	deu_Latn	74.7	74.2	**76.5**
diq_Latn	43.7	50.9	**52.2**	div_Thaa	0.0	**50.2**	47.5	ell_Grek	**73.7**	72.5	73.4
eml_Latn	33.5	42.1	**42.7**	eng_Latn	82.5	83.3	**83.6**	epo_Latn	64.5	**71.3**	68.6
est_Latn	72.2	72.3	**74.4**	eus_Latn	**59.2**	57.3	59.0	ext_Latn	39.1	45.3	**47.6**
fao_Latn	60.2	69.3	**72.7**	fas_Arab	51.0	42.9	**55.8**	fin_Latn	**75.6**	73.7	75.0
fra_Latn	77.3	75.5	**77.5**	frr_Latn	46.8	**57.0**	56.0	fry_Latn	74.0	**77.6**	75.9
fur_Latn	42.1	**56.5**	56.0	gla_Latn	50.6	62.0	**65.0**	gle_Latn	69.3	**73.2**	72.6
glg_Latn	80.2	78.2	**81.2**	grn_Latn	39.1	51.3	**52.0**	guj_Gujr	**60.8**	59.2	56.9
hbs_Latn	61.6	60.5	**69.8**	heb_Hebr	**51.4**	46.6	49.7	hin_Deva	68.5	69.5	**71.1**
hrv_Latn	77.0	76.1	**77.7**	hsb_Latn	64.0	70.9	**72.9**	hun_Latn	76.1	74.4	**76.7**
hye_Armn	52.7	**55.6**	53.9	ibo_Latn	36.4	55.3	**57.1**	ido_Latn	59.8	79.3	**81.7**
ilo_Latn	55.2	73.3	**78.3**	ina_Latn	53.2	56.4	**57.8**	ind_Latn	47.8	**57.5**	50.9
isl_Latn	68.8	71.3	**74.2**	ita_Latn	76.9	78.3	**79.0**	jav_Latn	**58.7**	55.8	53.3
jbo_Latn	19.2	25.8	**33.6**	jpn_Jpan	**19.3**	15.5	15.4	kan_Knda	**57.1**	56.0	52.0
kat_Geor	65.7	67.1	**69.1**	kaz_Cyrl	42.7	48.9	**52.4**	khm_Khmr	39.8	38.9	**40.4**
kin_Latn	58.3	63.8	**67.8**	kir_Cyrl	45.0	**45.3**	43.4	kor_Hang	49.5	50.7	**53.7**
ksh_Latn	42.4	56.7	**56.9**	kur_Latn	62.2	**64.3**	61.9	lat_Latn	69.1	74.5	**80.1**
lav_Latn	73.8	71.5	**73.9**	lij_Latn	38.7	**45.0**	41.9	lim_Latn	62.6	68.7	**71.7**
lin_Latn	37.1	50.3	**50.9**	lit_Latn	71.9	72.3	**73.2**	lmo_Latn	67.3	67.1	**72.3**
ltz_Latn	49.0	**68.4**	68.3	lzh_Hani	**15.7**	11.0	10.8	mal_Mlym	**62.8**	61.5	61.8
mar_Deva	60.7	59.8	**63.3**	mhr_Cyrl	43.4	60.9	**62.2**	min_Latn	42.3	**43.1**	38.7
mkd_Cyrl	75.8	72.9	**77.0**	mlg_Latn	54.6	**59.2**	56.5	mlt_Latn	42.4	70.1	**78.7**
mon_Cyrl	68.7	68.0	**71.0**	mri_Latn	16.0	**49.3**	49.1	msa_Latn	60.2	65.7	**69.4**
mwl_Latn	44.7	**47.6**	45.6	mya_Mymr	50.4	**53.8**	47.7	mzn_Arab	39.7	40.9	**46.4**
nan_Latn	42.3	**84.6**	79.0	nap_Latn	50.9	**54.3**	52.6	nds_Latn	62.5	74.7	**77.0**
nep_Deva	**63.5**	60.1	60.7	nld_Latn	79.8	80.5	**81.0**	nno_Latn	77.1	77.0	**77.5**
nor_Latn	76.7	75.9	**77.6**	oci_Latn	63.9	67.3	**75.2**	ori_Orya	33.0	31.4	**34.4**
oss_Cyrl	31.8	**55.3**	48.6	pan_Guru	49.3	**56.7**	45.8	pms_Latn	72.1	**78.1**	77.1
pnb_Arab	57.8	**66.9**	66.1	pol_Latn	77.4	77.5	**78.4**	por_Latn	78.1	78.7	**79.8**
pus_Arab	33.8	38.3	**40.9**	que_Latn	56.2	63.0	**69.3**	roh_Latn	51.9	57.6	**63.1**
ron_Latn	75.0	70.6	**77.5**	rus_Cyrl	64.5	67.7	**70.2**	sah_Cyrl	45.8	**73.7**	73.1
san_Deva	41.9	32.9	**42.1**	scn_Latn	54.4	**64.3**	60.4	sco_Latn	80.6	**88.1**	83.7
sgs_Latn	44.2	**64.2**	63.9	sin_Sinh	52.2	53.7	**59.7**	slk_Latn	76.3	77.9	**79.3**
slv_Latn	78.8	79.2	**80.6**	snd_Arab	39.1	40.1	**42.0**	som_Latn	56.0	**58.4**	55.4
spa_Latn	73.4	71.0	**75.0**	sqi_Latn	74.9	**76.2**	**76.2**	srp_Cyrl	59.6	66.4	**72.9**
sun_Latn	43.7	**55.9**	54.1	swa_Latn	60.3	**68.2**	66.9	swe_Latn	**71.6**	65.6	70.1
szl_Latn	57.9	65.3	**70.7**	tam_Taml	55.1	**57.5**	54.5	tat_Cyrl	39.6	**66.4**	66.1
tel_Telu	**49.4**	43.0	49.3	tgk_Cyrl	26.3	63.8	**68.5**	tgl_Latn	69.6	**76.0**	73.9
tha_Thai	3.8	**5.8**	4.1	tuk_Latn	45.3	57.9	**60.5**	tur_Latn	74.8	74.4	**76.6**
uig_Arab	45.5	49.7	**50.0**	ukr_Cyrl	**76.8**	72.7	73.5	urd_Arab	56.3	73.0	**75.6**
uzb_Latn	70.7	75.3	**75.7**	vec_Latn	57.5	65.6	**66.4**	vep_Latn	57.6	68.5	**74.2**
vie_Latn	66.9	69.0	**70.5**	vls_Latn	63.2	74.6	**76.0**	vol_Latn	**60.0**	58.7	59.0
war_Latn	59.6	**63.3**	62.8	wuu_Hani	28.9	30.1	**35.4**	xmf_Geor	50.6	64.6	**65.1**
yid_Hebr	46.2	53.2	**61.9**	yor_Latn	40.7	**61.6**	60.4	yue_Hani	**23.4**	23.3	20.3
zea_Latn	**68.1**	67.1	67.6	zho_Hani	**24.3**	**24.3**	21.0				

Table 22: F1 scores of baselines and FURINA on **NER**.

Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA	Language-script	XLM-R	Glot500-m	FURINA
afr_Latn	**89.3**	87.7	86.7	ajp_Arab	63.0	**73.1**	70.1	aln_Latn	54.1	49.7	**54.2**
amh_Ethi	63.9	**65.9**	63.6	ara_Arab	**67.9**	65.7	65.3	bam_Latn	25.1	39.5	**49.7**
bel_Cyrl	**86.0**	85.7	85.2	ben_Beng	82.0	**83.9**	82.2	bre_Latn	61.3	60.2	**67.8**
bul_Cyrl	**88.6**	88.3	87.3	cat_Latn	86.6	**86.9**	86.7	ceb_Latn	50.1	66.3	**68.8**
ces_Latn	**84.4**	**84.4**	83.4	cym_Latn	**65.8**	64.7	65.3	dan_Latn	90.3	90.1	**90.7**
deu_Latn	**88.4**	87.9	87.3	ell_Grek	**88.0**	83.9	82.1	eng_Latn	**96.3**	96.1	96.1
est_Latn	**85.9**	82.5	83.7	eus_Latn	**71.2**	61.1	63.6	fao_Latn	77.6	89.1	**89.4**
fas_Arab	70.3	71.3	**72.1**	fin_Latn	**85.1**	80.3	82.4	fra_Latn	85.9	86.4	**87.1**
gla_Latn	58.4	60.2	**60.6**	gle_Latn	**66.1**	64.8	65.5	glg_Latn	82.7	83.7	**84.2**
glv_Latn	27.2	52.7	**54.0**	grc_Grek	64.7	**72.6**	70.8	grn_Latn	10.5	20.1	**27.4**
gsw_Latn	49.1	81.0	**82.4**	hbo_Hebr	40.3	**50.0**	49.5	heb_Hebr	67.5	**68.3**	67.7
hin_Deva	73.2	71.2	**75.7**	hrv_Latn	85.2	**85.4**	83.6	hsb_Latn	72.1	**84.0**	83.7
hun_Latn	**82.3**	81.1	81.3	hye_Armn	**84.7**	83.9	84.4	hyw_Armn	79.0	81.7	**82.7**
ind_Latn	**83.7**	83.3	83.2	isl_Latn	**84.4**	82.8	83.7	ita_Latn	87.4	**89.2**	89.1
jav_Latn	73.4	73.7	**74.3**	jpn_Jpan	14.8	**34.9**	23.7	kaz_Cyrl	**77.2**	75.2	76.4
kmr_Latn	73.5	75.4	**77.7**	kor_Hang	**53.6**	52.5	52.6	lat_Latn	**75.6**	70.6	71.4
lav_Latn	**85.8**	82.6	83.9	lij_Latn	47.0	77.3	**77.4**	lit_Latn	**84.2**	80.2	81.8
lzh_Hani	14.5	**19.4**	7.4	mal_Mlym	**86.3**	84.6	84.9	mar_Deva	82.5	83.3	**84.7**
mlt_Latn	21.5	80.1	**81.5**	myv_Cyrl	39.2	64.3	**68.3**	nap_Latn	58.8	66.7	**88.9**
nds_Latn	57.3	76.5	**77.8**	nld_Latn	**88.6**	88.3	88.3	nor_Latn	**88.3**	87.5	87.4
pcm_Latn	46.7	57.9	**58.2**	pol_Latn	**83.1**	83.0	81.3	por_Latn	88.3	88.5	**88.8**
quc_Latn	28.7	62.0	**64.3**	ron_Latn	**83.6**	80.2	79.7	rus_Cyrl	**89.0**	88.8	88.0
sah_Cyrl	22.3	**77.6**	77.5	san_Deva	19.1	**24.8**	21.9	sin_Sinh	**58.5**	55.9	55.5
slk_Latn	84.1	**84.6**	83.5	slv_Latn	**78.1**	75.8	75.3	sme_Latn	29.8	73.9	**74.6**
spa_Latn	88.2	**88.9**	88.3	sqi_Latn	**78.5**	77.4	77.0	srp_Latn	**85.8**	84.8	83.0
swe_Latn	**93.4**	92.2	92.4	tam_Taml	**75.6**	75.0	74.4	tat_Cyrl	45.6	69.5	**69.8**
tel_Telu	**85.7**	82.2	82.4	tgl_Latn	73.3	75.0	**77.1**	tha_Thai	44.3	**56.5**	49.9
tur_Latn	**73.0**	70.4	72.4	uig_Arab	68.3	68.1	**68.9**	ukr_Cyrl	**85.5**	84.6	83.5
urd_Arab	59.6	65.8	**70.2**	vie_Latn	**70.4**	66.8	66.1	wol_Latn	25.6	57.1	**61.0**
xav_Latn	6.2	**17.6**	17.0	yor_Latn	22.7	63.2	**64.2**	yue_Hani	27.7	**42.7**	23.9
zho_Hani	24.6	**44.5**	23.4								

Table 23: F1 scores of baselines and FURINA on **POS**.