ScriptMix: Mixing Scripts for Low-resource Language Parsing

Jaeseong Lee, Dohyeon Lee, Seung-won Hwang*
Computer Science and Engineering, Seoul National University
{tbvj5914,waylight,seungwonh}@snu.ac.kr

Abstract

Despite the success of multilingual pretrained language models (mPLMs) for tasks such as dependency parsing (DEP) or part-of-speech (POS) tagging, their coverage of 100s of languages is still limited, as most of the 6500+ languages remains "unseen". To adapt mPLMs for including such unseen langs, existing work has considered transliteration and vocabulary augmentation. Meanwhile, the consideration of combining the two has been surprisingly lacking. To understand why, we identify both complementary strengths of the two, and the hurdles to realizing it. Based on this observation, we propose *ScriptMix*, combining two strengths, and overcoming the hurdle. Specifically, ScriptMix a) is trained with dual-script corpus to combine strengths, but b) with separate modules to avoid gradient conflict. In combining modules properly, we also point out the limitation of the conventional method AdapterFusion, and propose AdapterFusion+ to overcome it. We empirically show ScriptMix is effective– ScriptMix improves the POS accuracy by up to 14%, and improves the DEP LAS score by up to 5.6%. Our code is publicly available.

1 Introduction

Recently, pretrained language models (PLMs) have become the de facto standard for solving various natural language processing tasks, such as dependency parsing and part-of-speech tagging. To support a wide range of languages, multilingual PLMs (mPLMs) have been designed with a shared architecture that incorporates multiple languages, yielding promising results. For example, mBERT (Devlin et al., 2019) trains with Wikipedia articles in 104 languages to share a common feature space.

However, considering the availability of 6500+ languages, the majority of languages are still *unseen* by mBERT. Building another mPLM to include new languages incurs massive retraining costs. Even if such cost is afforded, performance is reportedly low, due to resource imbalance and limited model capacity (Wu and Dredze, 2020; Conneau et al., 2020).

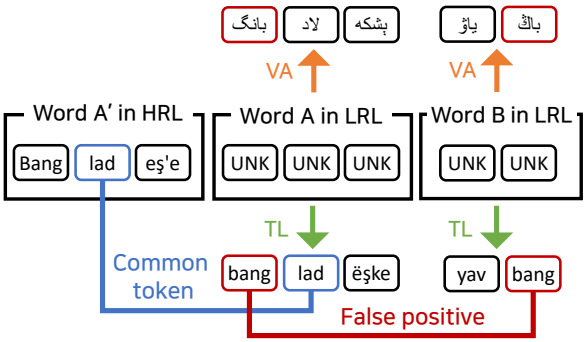

Figure 1: An illustrative example of Vocabulary Augmentation (VA), and Transliteration (TL). TL can share a common token (blue) with high-resource language (HRL), while VA is free from false positives (red).

A better alternative is to adapt mPLM by "specializing"[1] it for an unseen language using the following two approaches, as illustrated in Figure 1. When representing the word A from an unseen language, sharing the same semantics with A' from a high-resource language (HRL), **Vocabulary Augmentation (VA)** (Chau et al., 2020) augments low-resource language (LRL) tokens, by treating UNK as original LRL script such as 'بانگ' in the figure. **Transliteration (TL)** (Muller et al., 2021) replaces UNK with the script of an HRL, e.g., romanize A, which can generate a common token with A', shown as a blue word, 'lad'.

We first observe complementary aspects of TL and VA (Figure 1). TL, by sharing a common token, transfers the semantics from HRLs better, than VA using the original script that cannot overlap with HRL. However, this comes with the risk of a "false positive" transfer. To illustrate, consider two red

*Corresponding author

[1]We follow Chau and Smith (2021) to define specialization as a special case of an adaptation, which prepares a model exclusively for the target language.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 6430–6444

tokens 'بانگ' and 'باڭ', sharing the same sound with different semantics. TL maps them to the same token 'bang', even if the semantics originally differ. Meanwhile, VA does not suffer from false positives by preserving original scripts, but cannot increase token overlaps.

Inspired by the complementary strengths of VA and TL, we propose a novel method, *ScriptMix*, that combines the advantages of both techniques. ScriptMix, as the name suggests, mixes both scripts to obtain token overlap with HRLs by TL, while resolving potential false positives by source script from VA. We implement ScriptMix upon an adapter-based architecture (Pfeiffer et al., 2020), a state-of-the-art architecture for specialization at this moment, which devises language module (LA) and task module (TA) as in Figure 2a).

The first component of ScriptMix is *dual-script corpus*, which mixes the original corpus from VA and the transliterated corpus from TL ($\{S_s\}, \{S_t\}$ in Figure 2b). However, we argue this component by itself does not fully realize complementarity, as we later attribute to gradient conflict (Figure 3).

We devise our second component to overcome this hurdle: *language module separation*. To avoid gradient conflict from different scripts, we train separate adapters for each script (LA_s and LA_t in Figure 2c) . Then we linearly combine the outputs from these conflict-free adapters with a dynamically calculated weight vector. To calculate such a vector, the most natural way has been AdapterFusion (Pfeiffer et al., 2021a).

Despite the benefits of AdapterFusion, we have found it to have a harmful inductive bias that compromises its performance. To overcome this issue, we propose our third component, *AdapterFusion+*, which effectively eliminates the harmful inductive bias and improves overall performance.

We evaluate ScriptMix on dependency parsing and part-of-speech tagging– the human-annotated tasks with more low-resource languages than other NLP tasks. Combining all three components, ScriptMix could extract complementary knowledge from both scripts. To demonstrate the generalizability of ScriptMix, we also apply it for fusing UniPELT (Mao et al., 2022), where existing fusion techniques fail to lead to performance gains. ScriptMix improves the POS accuracy by up to 14%, and improves the DEP LAS score by up to 5.6%. Our contribution can be summarized as follows:

- We observe the potential of complementarity between VA and TL, from which, we propose the first solution to achieving complementarity.
- We identify gradient conflict as a hurdle, and overcome it with the proposed *language module separation*.
- We point out the limitation of the conventional method to combine adapter outputs, and propose AdapterFusion+ to overcome it.
- ScriptMix is generalizable to different types of modules, and works both on few-shot training and the setting with a larger train dataset.
- Our code and dataset are publicly available.[2]

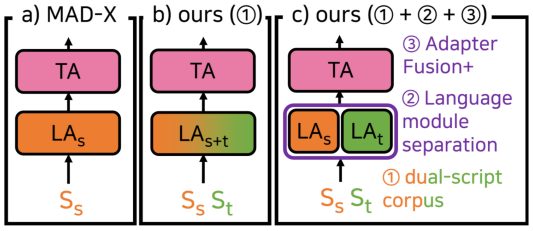

Figure 2: Comparison between MAD-X (§ 3.1) and ours (ScriptMix). We propose ① dual-script corpus (§ 3.3), ② language module separation (§ 3.4) and ③ AdapterFusion+ (§ 3.5) to achieve complementarity of source script s from VA and transliterated script t from TL.

2 Related Work

Expanding mPLM New languages can be added during pretraining multilingual models, though LRL performance is degraded due to resource imbalances (Wu and Dredze, 2020).

Specializing mPLM Specializing mPLM is a more effective alternative to overcome imbalances. From the data perspective, TL (Muller et al., 2021) and VA (Wang et al., 2020a; Chau et al., 2020; Chau and Smith, 2021), compared and contrasted in Figure 1, are proposed as specialization methods.

From the architecture perspective, one may add additional modules with few parameters for specialization. Out of the multilingual domain, modules such as Adapters (Houlsby et al., 2019), or UniPELT (Mao et al., 2022) have been proposed.

Among them, adapters have been more broadly applied for cross-lingual transfer. MAD-X (Pfeiffer et al., 2020) trains language adapters (LAs) and task adapters (TAs). Then MAD-X stacks them to utilize knowledge from both unlabeled and labeled

[2]https://anonymous.4open.science/r/scriptmix-13D7/

datasets, achieving a state-of-the-art. We use this as our baseline architecture, and also generalize to UniPELT.

Our Distinction Our distinction is observing the complementarity of VA and TL, for better specialization to an unseen language. VA preserves the original script, and TL increases token overlaps, with the risk of false positives (Figure 1). Our contribution is achieving the complementarity between VA and TL, through realizing and resolving the hurdles for the goal.

The easiest way to integrate knowledge from two different corpora would be a polyglot corpus (Devlin et al., 2019; Conneau et al., 2020). We devise a dual-script corpus inspired by this approach, but we argue that training one module with a polyglot corpus is suboptimal. To overcome, we propose to train two different adapters and add a component to fuse the knowledge.

To fuse two adapter outputs, AdapterFusion (Pfeiffer et al., 2021a) devises an attention-based fusion mechanism. However, we observe that AdapterFusion is suboptimal in our scenario, and propose to reduce inductive bias from it.

Existing work on reducing inductive bias includes MLP-Mixer (Tolstikhin et al., 2021)– it reduces inductive bias from convolutional neural networks by replacing them with MLPs, which was effective for our purpose of bias reduction from AdapterFusion.

3 Proposed Method: ScriptMix

3.1 Preliminaries: MAD-X

Given language l, for each layer in an mPLM, MAD-X adds two adapters as in Figure 2a); language adapter (LA_l) and task adapter (TA_l).

Let h be the output of a layer from an mPLM. MAD-X prepares $\text{LA}_l = W_{lu} \circ \phi \circ W_{ld}$, where $W_{ld} \in \mathbb{R}^{d_h \times d_b}$, $W_{lu} \in \mathbb{R}^{d_b \times d_h}$, and ϕ is ReLU layer, and prepares $\text{TA}_l = W_{\tau u} \circ \phi \circ W_{\tau d}$ similarly. MAD-X first transforms the output as $\text{LA}_l(h)$, and updates the parameters of LA_l, with unlabeled data of l. Then, MAD-X alters the output as $\text{TA}_l \circ \text{LA}_l(h)$, and trains parameters of TA_l using labeled data.

3.2 Motivation: TL vs VA

A key motivation of ScriptMix comes from comparing TL and VA. To compare TL and VA, we build simple baselines upon MAD-X.

transliterated character	can be generated from
h	ھ, ح, ع
n	ڭ, ن
g	ڭ, گ
k	ك, ع

Table 1: Collapse of characters from Uyghur to Latin transliteration, which may cause false positives.

MAD-X-VA/TL Let V and E_V denote the vocabulary and the corresponding embedding layer of the given mPLM. Let us denote the original script used in VA as the source language s and the transliterated script t. Given the original corpus $C_s = \{\tilde{S}_{s,i}\}$ and transliterated corpus $C_t = \{\tilde{S}_{t,i}\}$, we first train new wordpieces V_n to deal with these corpora, and add them to V, obtaining $V' = V \cup V_n$. Next, we introduce new embeddings E_{V_n}, and pretrain them with C_s or C_t, while freezing other parameters. These embeddings are concatenated to the original E_V, to build $E_{V'}$.

Finally, upon the modified mPLM with $E_{V'}$, MAD-X-VA trains LA_s, TA_s with C_s, and MAD-X-TL trains LA_t, TA_t with C_t.

TL vs VA Recall from Figure 1, that TL suffers from "false positive" transfer, while VA does not, though VA is not as effective in shared semantics as TL sharing common tokens.

We run a preliminary experiment to identify such pros and cons, with Uyghur as an example. In the Uyghur corpus, we examine which distinct characters are mapped to the same one by TL (Table 1), which may cause false positives. We train parsers following the settings we describe in Section 4.1. We evaluated all sentences, and checked whether the head and label prediction for each token with or without collapsed characters was correct. We find TL is superior to VA for tokens without such collapsed characters, but inferior to VA for tokens with these characters.

Based on this observation, we propose to mix both scripts. We hypothesize mixing both scripts would provide synergetic strength, while alleviating weaknesses.

3.3 Dual-Script Corpus, the Gradient Conflict

First, to leverage both scripts, we design a corpus for our goal. At every batch, we utilize

$$[\tilde{S}_{s,k}, \cdots, \tilde{S}_{s,k+\frac{b}{2}}, \tilde{S}_{t,k}, \cdots, \tilde{S}_{t,k+\frac{b}{2}}] \quad (1)$$

to train LA, where b is the batch size. With dual-script corpus, we expect to obtain a positive transfer from both TL and VA.

However, we conjecture that full complementary is yet to be realized, as a dual-script LA would introduce negative interference. Wang et al. (2020b) similarly observed conflict of gradients from different languages, which interferes with positive transfer in multilingual language modeling.

Such a negative interference by gradient conflict can be measured by a similarity score between gradients from each language. Formally, let g_1 and g_2 be the gradients from two different languages. Then we can define gradient conflict to be high, when $cos(g_1, g_2)$ is low. Wang et al. (2020b) show that gradient similarity between two batches from two languages is lower than that from a single language, which explains negative interference.

Following this convention, we plot the relative cosine similarity between gradients from training an LA with a dual-script corpus, compared with that of a single-script corpus.[3] We observe a similar behavior (color-mixed line in Figure 3): training an LA with dual-script corpus shows lower cosine similarity than using single-script. This suggests a negative interference occurs, which explains the limited performance gain.

Motivated by this observation, we propose an alternative design, of separating LAs, which avoids a gradient conflict by design. By separating adapters, our design would be free from gradient conflict, making the relative similarity constantly maximized to 1.[4]

3.4 Language Module Separation

As motivated by Figure 3, we propose to train separate LAs, each for one script, then fuse their knowledge. The figure shows that the relative gradient similarity is maximized when the LAs are separated, resolving the gradient conflict issue. First, we train LA_s with $\{\tilde{S}_{s,i}\}$ and LA_t with $\{\tilde{S}_{t,i}\}$. Since we utilize only one script in training each LA, LA is free from such gradient conflict (purple line in Figure 3). Then, we design to linearly combine the outputs of these conflict-free LAs, with an adaptively calculated weight a.

[3]To analyze gradient similarity, we collected gradients per 1024 sentences, and exponentially smoothed the gradients by 0.6 to capture the tendency.

[4]As we need to train LA_s and LA_t for MAD-X-VA and MAD-X-TL respectively, we average the gradient similarity of the two, when dealing with the single-script corpus.

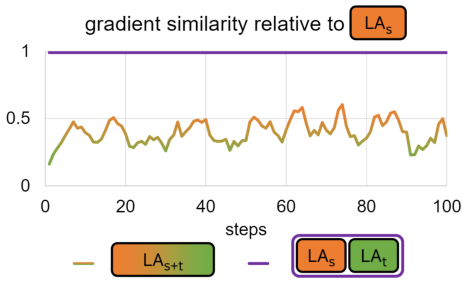

Figure 3: Relative gradient similarity of training with dual-script corpus, with (purple) and without LA separation (color-mixed), compared with training with single-script corpus. LA separation resolves gradient conflict.

With this, by design, we are free from gradient conflict between different scripts in training LA, while leveraging dual-script corpus, combining the strengths of the two.

3.5 AdapterFusion+: Reducing Inductive Bias

To evaluate the weight vector a for fusing these adapters, the most natural direction is using AdapterFusion (Pfeiffer et al., 2021a). Formally, given LA_s for the original script and LA_t for its transliteration, we may design M, the ScriptMix, which updates the output h to $M(h)$ as follows:

$$a_l = \text{softmax}(h^T F_q \otimes \text{LA}_l(h)^T F_k) \quad (2)$$

$$z_l = \text{LA}_l(h)^T F_v, \; l \in \{s, t\} \quad (3)$$

$$M(h) = \sum_{l \in \{s,t\}} a_l z_l \quad (4)$$

where $\otimes$ is the dot product, and F_q, F_k, F_v are learnable matrices.

However, we observe it is suboptimal in our case (§ 4.2.2) and attribute the reason to its inductive bias– The weight a_l gets higher as the output of LA_l is similar to the input h. It may help focus on suitable adapters, but there is a simple counterexample– an adapter with identity mapping, passing over the input as output, would get high a_l.

Therefore we propose AdapterFusion+, to reduce the inductive bias from the conventional AdapterFusion. We design to learn a good weight vector a, without relying on the inductive bias considering the similarities between adapter outputs and inputs. To reduce such inductive bias, we use MLP instead, inspired by MLP-Mixer (Tolstikhin et al., 2021). We re-design the Eq. 2 as follows:

$$(a_s, a_t) = \text{softmax}(f(\text{LA}_s(h)^T; \text{LA}_t(h)^T)) \quad (5)$$

where f is an MLP, and ; indicates the concatenation. To realize, we design f as a linear layer, in-

spired by Ma et al. (2018).[5] Such a design is even using fewer parameters than AdapterFusion; It can reduce 2-3 times of parameters. This parameter-efficient fusion method can successfully enable the mixture of knowledge from two different adapters (§ 4.2.2).

3.6 Generalizing ScriptMix for UniPELT

In addition to our proposed method, we aim to demonstrate the generalizability of our approach by applying it to UniPELT, a technique known to be more robust than adapters in low-resource settings (Mao et al., 2022). UniPELT combines several techniques, such as LoRA (Hu et al., 2022) and prefix-tuning (Li and Liang, 2021), with adapters to improve the self-attention mechanism. Unlike AdapterFusion, fusing UniPELT has not been studied yet, and we found applying conventional techniques fails to achieve gains, while our proposed fusion allows them.

First, we describe how UniPELT differs from adapters, then establish UniPELT-based ScriptMix.

3.6.1 Preliminaries: UniPELT

In addition to adapters, UniPELT (Mao et al., 2022) adds two more techniques, LoRA (Hu et al., 2022) and prefix-tuning (Li and Liang, 2021). These three are combined by gating mechanism as follows.

Formally, the traditional self-attention layer manipulates input hidden representation h_i as follows:

$$Q = W_q(h_i), K = W_k(h_i), V = W_v(h_i) \quad (6)$$

$$h_a = \text{softmax}(QK^T/c)V \quad (7)$$

where W_q, W_k, W_v are learnable matrices, and c is some constant.

UniPELT first adds LoRA (Hu et al., 2022) to W_q and W_k. The key idea of LoRA is restricting matrix updates to a low-rank matrix. Adding LoRA to W_k changes K to K' as follows:

$$W'_k = W_k + \alpha_k W_{ku} W_{kd} \quad (8)$$

$$K' = W'_k(h_i) \quad (9)$$

where $W_{kd} \in \mathbb{R}^{d_h \times d_m}$ and $W_{ku} \in \mathbb{R}^{d_m \times d_h}$. Similarly, it alters Q into Q'.

UniPELT then adds prefix-tuning (Li and Liang, 2021) to K' and V. The gist of prefix-tuning is learning an extra prefix vector for the given representation. With prefix-tuning, K' is manipulated to K'_p as follows:

$$K'_p = P_k; K' \quad (10)$$

where ; denotes the concatenation, and P_k is a learnable vector, typically modeled as an MLP. Similarly, it alters V into V_p. These Q', K'_p, V_p are used to calculate the new self-attention output. Finally, they design a gating mechanism to balance each component. For details, please refer to their original paper (Mao et al., 2022).

3.6.2 UniPELT-based MAD-X and ScriptMix

Since ScriptMix is based on MAD-X, we first build UniPELT-based MAD-X. Then we design UniPELTFusion and UniPELTFusion+, to extend ScriptMix to UniPELT with AdapterFusion and AdapterFusion+ respectively. These will be compared in § 4.2.3, showing UniPELTFusion+ can only establish complementarity.

UniPELT-based MAD-X We allow language and task components of LoRA and prefix-tuning. Eq. 8-10 is modified as follows:

$$W'_{lk} = W_k + \alpha_{lk} W_{lku} W_{lkd} \quad (11)$$

$$W'_k = W'_{lk} + \alpha_{\tau k} W_{\tau k} \quad (12)$$

$$K' = W'_k(h_i) \quad (13)$$

$$K'_{lp} = P_{lk}; K' \quad (14)$$

$$K'_p = P_{\tau k}; K'_{lp} \quad (15)$$

where l, τ indicates the components of language UniPELT, and task UniPELT, respectively.

UniPELT-based ScriptMix We need to fuse LoRAs and prefixes from different scripts. First, we fuse LoRAs from each script similarly to Eq. 3,4 –Eq. 11 is changed as follows:

$$z'_l = (W_k + \alpha_{lk} W_{lk}) F'_v, \; l \in \{s, t\} \quad (16)$$

$$W'_{lk} = \sum a'_l z'_l \quad (17)$$

Likewise, replacing the language prefix of single script changes Eq. 14 as follows:

$$z_{lp} = F_{vp}(P_{lk}; K'), \; l \in \{s, t\} \quad (18)$$

$$K'_{lp} = \sum a_{lp} z_{lp} \quad (19)$$

Finally, we design a'_l, a_{lp}. We may design them similarly to AdapterFusion (Eq. 2) as follows:

$$a'_l = \text{softmax}(h_i^T F'_q \otimes (W_k + \alpha_{lk} W_{lk})(h_i)^T F'_k)$$

$$a_{lp} = \text{softmax}(h_i^T F_{qp} \otimes (P_{lk}; K')^T F_{kp})$$

[5] We also tried a 2-layer MLP, but the gain was similar.

which we name as UniPELTFusion.[6] However, as we will discuss in § 4.2.2, it cannot achieve complementarity.

Alternatively, we design a'_l, a_{lp} similarly to AdapterFusion+ (Eq. 5) as follows:

$$(a'_s, a'_t) = \text{softmax}(f'((W_k + \alpha_{sk}W_{sk})(h_i)^T; (W_k + \alpha_{tk}W_{tk})(h_i)^T))$$
$$(a_{sp}, a_{tp}) = \text{softmax}(f_p((P_{sk}; K')^T; (P_{tk}; K')^T))$$

which we name as UniPELTFusion+.

The whole process is applied similarly to Q', V_p.

4 Experiments

We mainly evaluate our method on few-shot training, which assumes a more low-resource scenario. We also validate that ScriptMix is effective when more train data is allowed.

4.1 Experimental Settings

We use mBERT as the representative mPLM, to be consistent with previous works for TL (Muller et al., 2021) and VA (Chau et al., 2020).

Tasks and Datasets Identifying tasks supporting low-resource languages not covered by mPLMs is challenging (Ahuja et al., 2022). We evaluate part-of-speech tagging (POS) and dependency parsing (DEP), which are human-annotated datasets with more low-resource languages than other NLP tasks (Chau et al., 2020). As the labeled dataset, we utilize the treebanks from Universal Dependencies (Nivre et al., 2020), version 2.7 (Zeman et al., 2020). For the few-shot evaluation, we randomly sample 32 and 16 examples from the train dataset and dev dataset. We utilize Wikipedia articles extracted with WIKIEXTRACTOR,[7] as the unlabeled data.

Language selection Among unseen languages not covered by mBERT, we aim to exhaustively cover all languages, with transliterator to Latin script, and a sufficient amount of treebanks for evaluation. We allow languages lacking train data, by performing 8-fold cross-validation using test data only.[8] This results in five languages to probe with: ug, cu, bxr, myv, and am (Table 2).

[6]Following AdapterFusion, we calculate the weight a per token. However in prefix-tuning, the sequence length differs, thus we calculate a per sequence, using the average of representations over tokens.

[7]https://github.com/attardi/wikiextractor

[8]Since Russia Buriat has only 19 training examples, we also perform 8-fold cross-validation for it.

language (iso code)	script	# train	# dev	# test	# wiki
Old Church Slavonic (cu)	Cyrs	4214	1073	1141	1136
Uyghur (ug)	ug-Arab	1656	900	900	5255
Russia Buriat (bxr)	Cyrl	19	0	908	2763
Erzya (myv)	Cyrl	0	0	1550	7499
Amharic (am)	Ethi	0	0	1074	15002

Table 2: Unseen languages used for the experiments in this paper. We report the size of the unlabeled dataset (# wiki articles), and the train/dev/test split.

Transliterators For Uyghur, we use the transliterator used by Muller et al. (2021). We utilize WIKITRA (Batsuren et al., 2019) for the other languages.

Implementation Details We train new wordpieces with a size of 5K and select subwords that reduce unknown tokens at most, following Chau et al. (2020). We add 1K new subwords for ug, 500 and 99 for cu and its transliteration, 99 for bxr transliteration, 99 and 99 for myv and its transliteration, 3K and 99 for am and its transliteration.[9]

To train language adapters, we use similar settings to Pfeiffer et al. (2020). We perform MLM with $d_b = 384$, batch size of 64, learning rate of 1e-4, for 50K steps,[10] with sequence length of 512. We conduct MLM on TPUv2-8 or TPUv3-8, taking less than 4 hours.

To fine-tune for dependency parsing, we follow the setting from Rust et al. (2021). We use a single-layer multi-class classifier for POS tagging, and the transformer-based variant (Glavaš and Vulić, 2021) of the deep bi-affine attention dependency parser (Dozat and Manning, 2017) for dependency parsing. We fine-tune with $d_b = 48$, batch size of 32, learning rate of 5e-4, sequence length of 256, for 30 epochs. F_q, F_k, F_v in Eq. 2 or f in Eq. 5 is added in this stage and jointly trained with the task adapters, with language adapters frozen.

For experiments with UniPELT, we let d_b as 48, d_m as 8, and the length of P_k as 10 for the task module, following the default setting from Mao et al. (2022). For the language module, we scale these numbers by 8 times, following Pfeiffer et al. (2020). Settings for training are the same as the experiments with adapters.

We report the LAS score for dependency parsing, and accuracy for POS tagging. We evaluate with

[9]We chose the least number of subwords among {99, 300, 500, 1000, 3000} to make the UNK ratio lower than 1%. Since naïve TL sometimes increases UNK overlaps, we add new subwords for TL also, to increase more useful overlaps.

[10]Low-resource language adapters are trained for 50K steps (https://adapterhub.ml/explore/qu/wiki/).

POS (acc %)	32-shot						16-shot					
	ug	cu	bxr	myv	am	avg	ug	cu	bxr	myv	am	avg
MAD-X-VA	72.15	49.94	63.04	65.22	65.09	63.09	68.33	43.99	56.51	61.84	60.86	58.31
MAD-X-TL	71.65	52.61	63.11	65.74	68.07	64.23	68.04	44.58	58.05	60.39	65.25	59.26
max(MAD-X-VA/TL + UniPELT)	76.84	62.17	70.55	74.08	78.24	72.38	71.08	54.12	63.78	65.29	73.27	65.51
ScriptMix (UniPELTFusion)	77.32	65.81	69.84	69.90	80.45	72.67	70.44	57.90	61.78	58.96	74.43	64.70
ScriptMix (UniPELTFusion+)	77.40	67.91	71.61	73.70	82.27	74.58	72.61	58.49	65.64	63.89	76.57	67.44
ScriptMix (AdapterFusion)	78.58	63.62	71.09	76.31	80.17	73.95	74.89	54.47	65.31	70.50	73.90	67.81
ScriptMix (AdapterFusion+)	79.85	66.80	73.63	78.55	82.45	**76.26**	75.79	56.59	67.36	71.51	77.15	**69.68**
DEP (LAS %)	32-shot						16-shot					
	ug	cu	bxr	myv	am	avg	ug	cu	bxr	myv	am	avg
MAD-X-VA	26.81	19.70	23.17	27.63	37.24	26.91	22.71	16.47	16.83	21.74	28.54	21.26
MAD-X-TL	26.67	21.16	23.85	26.17	38.22	27.21	21.54	17.63	17.86	19.60	30.45	21.42
max(MAD-X-VA/TL + UniPELT)	28.83	20.97	26.33	32.90	38.51	29.51	23.18	18.98	20.29	25.49	30.34	23.66
ScriptMix (UniPELTFusion)	31.18	23.70	27.59	28.11	40.47	30.21	23.79	19.22	20.45	22.44	30.41	23.26
ScriptMix (UniPELTFusion+)	31.48	24.96	28.06	28.90	40.73	**30.82**	24.87	19.39	20.51	24.29	29.94	23.80
ScriptMix (AdapterFusion)	30.15	23.01	26.73	31.84	41.54	30.65	23.18	18.58	19.91	25.92	30.93	23.70
ScriptMix (AdapterFusion+)	29.58	24.21	26.28	32.27	40.06	30.48	24.46	20.25	20.14	25.90	32.21	**24.59**

Table 3: Averaged scores of ScriptMix and the baselines on POS and DEP.

original or transliterated treebanks, and choose the best model based on the validation score. We report the average score over 5 runs, except for 8-fold cross-validation experiments, where we run once per fold and take the average. Fine-tuning is conducted on RTX 3090, taking less than an hour.

4.2 Experimental Results and Analysis

4.2.1 Effectiveness of ScriptMix

To discuss, we compare ScriptMix with the following baselines– 1) MAD-X-VA: MAD-X trained with vocabulary augmented data of each language, 2) MAD-X-TL: MAD-X trained with transliterated data of each language.

ScriptMix outperforms these baselines by a large margin (last line in Table 3). For example, on POS, ScriptMix improves by more than 14% accuracy in Old Church Slavonic (cu), or Amharic (am). On DEP, ScriptMix improves 3-4% LAS scores on average. This implies that ScriptMix successfully combines the strengths of both scripts: obtaining the benefit of TL, while alleviating its downside with the knowledge from the original script.

4.2.2 Effectiveness of New Fusion Method

We empirically show that the conventional fusion method from AdapterFusion (Pfeiffer et al., 2021a) is suboptimal as we described in section 3.5, and how our solution overcomes it. For example, in the 16-shot POS experiment, we observe that UniPELTFusion, combining the knowledge with the conventional fusion approach, loses the performance on average (red in Table 3). Besides, combining the knowledge with our solution resolves it; UniPELTFusion+ improves the performance by 2.7% on average compared to UniPELTFusion.

We observe that AdapterFusion+ and UniPELTFusion+ outperform AdapterFusion and UniPELTFusion most cases (35/40). These show that the existing fusion method by AdapterFusion is suboptimal for ScriptMix, and our solution alleviates it successfully.

4.2.3 Generalizability of ScriptMix

Table 3 shows that ScriptMix can be applied to UniPELT also. On average, ScriptMix with UniPELT improves the baselines with UniPELT up to 3% for POS, and 2% for DEP (third vs fifth line in Table 3). We re-emphasize that This is not possible without our proposed UniPELTFusion+ (red in Table 3).

4.2.4 Adapter vs UniPELT for ScriptMix

We evaluate whether introducing UniPELT is needed for ScriptMix. UniPELT is expected to perform better on our few-shot scenarios, since it is known to be more robust (Mao et al., 2022). This is true for the baselines (third lines of Table 3).

However surprisingly, for ScriptMix, using adapters only is sufficient; On average, ScriptMix using adapters is on par or even better than ScriptMix with UniPELT (fifth vs last line in Table 3). This leads to a more parameter-efficient result, as we will discuss in § 4.2.6.

	DEP						POS					
	ug	cu	bxr	myv	am	avg	ug	cu	bxr	myv	am	avg
max(MAD-X-VA, MAD-X-TL)	66.63	74.33	49.78	66.15	67.59	64.90	89.65	94.40	89.01	92.49	92.60	91.63
ScriptMix (AdapterFusion)	67.98	76.94	54.32	70.07	69.88	67.84	89.78	94.65	89.07	93.04	92.72	91.85
ScriptMix (AdapterFusion+)	68.23	78.59	55.41	70.83	70.15	**68.64**	89.89	94.81	90.04	93.20	92.69	**92.13**

Table 4: Results on POS and DEP, when using full training datasets.

			w/ Adapters		w/ UniPELT	
dual-script	lang module	sep	POS	DEP	POS	DEP
			61.83	24.81	68.94	26.58
✓			64.88	27.09	67.57	25.05
✓		✓	**72.97**	**27.54**	**71.01**	**27.31**

Table 5: Ablation study of each component.

	params(M)	POS
ScriptMix (UniPELTFusion+)	28.49	71.01
ScriptMix (AdapterFusion)	21.11	70.88
ScriptMix (AdapterFusion+)	**7.64**	**72.97**

Table 6: The number of parameters used for fine-tuning.

4.2.5 Language Module Separation Is Needed

To investigate whether separating language modules is essential, we train a single language module using the dual-script corpus. We even allow the module to have 2x of parameters, since it has to model both of original corpus and the transliteration. We report the averaged score over 32-shot and 16-shot experiments in varying languages.

We observe a clear deficit of ScriptMix without language module separation (Table 5). In the UniPELT-based architecture, we even observe performance degradation when we train a single module. This indicates that the language module suffered negative interference, while our language module separation alleviated it successfully.

4.2.6 Parameter Efficiency of ScriptMix

To investigate whether the gain of ScriptMix simply comes from the newly introduced parameters for fusion, we describe the number of parameters needed for fine-tuning POS tagging, in Table 6. We also report the averaged score over 32-shot and 16-shot POS experiments.

Table 6 shows that the increase in parameters does not guarantee a performance increase. This implies that the gain of ScriptMix with AdapterFusion+ does not simply come from parameter increase. We also highlight that AdapterFusion+ is more parameter-efficient than AdapterFusion: It achieves better performance with about 3 times lower parameters.

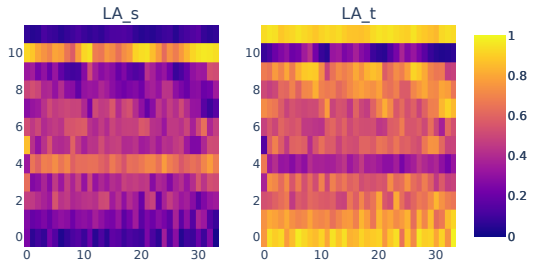

Figure 4: Heatmap of weights a of LA_s and LA_t, when inferencing with transliterated Uyghur POS examples. x and y axes refer to token and layer indices respectively.

4.2.7 Experiments on Full-Data Setting

We also argue that ScriptMix with AdapterFusion+ is effective even if we remove the extremely low-resource assumption. We conduct experiments allowing full train datasets, and report the averaged score in Table 4.

We observe that ScriptMix with AdapterFusion+ improves the performance over baselines. On DEP, we improve the LAS score by up to 5.6%. On POS, the gain is smaller, as the room for improvement smaller. Again, AdapterFusion is working inferior to AdapterFusion+.

4.3 Analysis: Complementarity Visualization

We argued that fusing the outputs of two adapters LA_s and LA_t benefits complementarily. To support this, we visualize the weight vector a_s, a_t of transliterated Uyghur as a representative. Visualization for other languages can be found in the Appendix.

Figure 5 shows that ScriptMix complementarily uses two adapters from different scripts. For example, for layer 4 and 10, LA_s contributes more, while LA_t is utilized more for layer 0 and 11. This reveals that using one adapter trained from a single script is insufficient; ScriptMix leverages complementary benefits from two.

5 Conclusion and Discussion

We studied how to specialize mPLM to unseen languages and introduced ScriptMix, the complementary approach to merge the benefits of TL and VA. We verified our effectiveness across a comprehensive array of languages. While our focus was to show the feasibility of script-mixing, future work could explore combining TL with improved VA methods, such as Pfeiffer et al. (2021b).

6 Limitation

In this work, we studied a limited set of tasks and languages to evaluate the proposed method. We believe we exhaustively evaluated our method on available low-resource human-annotated task datasets to apply ScriptMix. We leave the evaluation on more tasks if public datasets become available.

We also focused on only one mPLM, mBERT. However, we are following previous works to claim their effectiveness only on mBERT (Muller et al., 2021; Chau and Smith, 2021; Pfeiffer et al., 2021b). We leave extending the evaluation to more recent mPLMs, such as XLM-R (Conneau et al., 2020) as a future work.

Acknowledgements

This research was partially supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2024-2020-0-01789) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). This work was also partially supported by IITP grant funded by MSIT (No.2022-0-00077, AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data). We would also like to thank Google's TPU Research Cloud (TRC) program for providing Cloud TPUs.

References

Kabir Ahuja, Sandipan Dandapat, Sunayana Sitaram, and Monojit Choudhury. 2022. Beyond Static models and test sets: Benchmarking the potential of pre-trained models across tasks and languages. In *Proceedings of NLP Power! The First Workshop on Efficient Benchmarking in NLP*, pages 64–74, Dublin, Ireland. Association for Computational Linguistics.

Khuyagbaatar Batsuren, Gabor Bella, and Fausto Giunchiglia. 2019. CogNet: A Large-Scale Cognate Database. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3136–3145, Florence, Italy. Association for Computational Linguistics.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with Multilingual BERT, a Small Corpus, and a Small Treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.

Ethan C. Chau and Noah A. Smith. 2021. Specializing Multilingual Language Models: An Empirical Study. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 51–61, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

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.

Timothy Dozat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Goran Glavaš and Ivan Vulić. 2021. Is Supervised Syntactic Parsing Beneficial for Language Understanding Tasks? An Empirical Investigation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3090–3104, Online. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2790–2799. PMLR.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. 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 4582–4597, Online. Association for Computational Linguistics.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1930–1939, New York, NY, USA. Association for Computing Machinery.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264, Dublin, Ireland. 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.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021a. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2021b. UNKs Everywhere: Adapting Multilingual Language Models to New Scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10186–10203, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. 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 3118–3135, Online. Association for Computational Linguistics.

Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. 2021. MLP-Mixer: An all-MLP architecture for vision. In *Advances in Neural Information Processing Systems*.

Zihan Wang, Karthikeyan K, Stephen Mayhew, and Dan Roth. 2020a. Extending Multilingual BERT to Low-Resource Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2649–2656, Online. Association for Computational Linguistics.

Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. 2020b. On Negative Interference in Multilingual Models: Findings and A Meta-Learning Treatment. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4438–4450, Online. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2020. Are All Languages Created Equal in Multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Elia Ackermann, Noëmi Aepli, Hamid Aghaei, Željko Agić, Amir Ahmadi, Lars Ahrenberg, Chika Kennedy Ajede, Gabrielė Aleksandravičiūtė, Ika Alfina, Lene Antonsen, Katya Aplonova, Angelina Aquino, Carolina Aragon, Maria Jesus Aranzabe, Hórunn Arnardóttir, Gashaw Arutie, Jessica Naraiswari Arwidarasti, Masayuki Asahara, Luma Ateyah, Furkan Atmaca, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Keerthana Balasubramani, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, Colin Batchelor, John Bauer, Seyyit Talha Bedir, Kepa Bengoetxea, Gözde Berk, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Kristín Bjarnadóttir, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd,

Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Özlem Çetinoğlu, Fabricio Chalub, Ethan Chi, Yongseok Cho, Jinho Choi, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Mehmet Oguz Derin, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Arawinda Dinakaramani, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Sidney Facundes, Richárd Farkas, Marília Fernanda, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Fabrício Ferraz Gerardi, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Loïc Grobol, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Tunga Güngör, Nizar Habash, Hinrik Hafsteinsson, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Muhammad Yudistira Hanifmuti, Sam Hardwick, Kim Harris, Dag Haug, Johannes Heinecke, Oliver Hellwig, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Eva Huber, Jena Hwang, Takumi Ikeda, Anton Karl Ingason, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Hildur Jónsdóttir, Fredrik Jørgensen, Markus Juutinen, Sarveswaran K, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Abdullatif Köksal, Kamil Kopacewicz, Timo Korkiakangas, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Parameswari Krishnamurthy, Sookyoung Kwak, Veronika Laippala, Lucia Lam, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Maria Levina, Cheuk Ying Li, Josie Li, Keying Li, Yuan Li, KyungTae Lim, Krister Lindén, Nikola Ljubešić, Olga Loginova, Andry Luthfi, Mikko Luukko, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Hiroshi Matsuda, Yuji Matsumoto, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Karina Mischenkova, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, AmirHossein Mojiri Foroushani, Amirsaeid Moloodi, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Tomohiko Morioka, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Mariam Nakhlé, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Alireza Nourian, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adédayọ̀ Olúòkun, Mai Omura, Emeka Onwuegbuzia, Petya Osenova, Robert Östling, Lilja Øvrelid, Şaziye Betül Özateş, Arzucan Özgür, Balkız Öztürk Başaran, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Natalia Perkova, Guy Perrier, Slav Petrov, Daria Petrova, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Taraka Rama, Loganathan Ramasamy, Carlos Ramisch, Fam Rashel, Mohammad Sadegh Rasooli, Vinit Ravishankar, Livy Real, Petru Rebeja, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Eiríkur Rögnvaldsson, Mykhailo Romanenko, Rudolf Rosa, Valentin Roșca, Davide Rovati, Olga Rudina, Jack Rueter, Kristján Rúnarsson, Shoval Sadde, Pegah Safari, Benoît Sagot, Aleksi Sahala, Shadi Saleh, Alessio Salomoni, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Kevin Scannell, Salvatore Scarlata, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Muh Shohibussirri, Dmitry Sichinava, Einar Freyr Sigurðsson, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Maria Skachedubova, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Steinþór Steingrímsson, Antonio Stella, Milan Straka, Emmett Strickland, Jana Strnadová, Alane Suhr, Yogi Lesmana Sulestio, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Mary Ann C. Tan, Takaaki Tanaka, Samson Tella, Isabelle Tellier, Guillaume Thomas, Liisi Torga, Marsida Toska, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Utku Türk, Francis Tyers, Sumire Uematsu, Roman Untilov, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Aya Wakasa, Joel C. Wallenberg, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Paul Widmer, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Kayo Yamashita, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrt-

ský, Shorouq Zahra, Amir Zeldes, Hanzhi Zhu, and Anna Zhuravleva. 2020. Universal dependencies 2.7. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

	POS (acc %)	32-shot					16-shot				
		ug	cu	bxr	myv	am	ug	cu	bxr	myv	am
VA	MAD-X	75.60	50.00	61.13	65.28	64.74	70.88	42.16	53.59	62.70	58.78
	ScriptMix (AdapterFusion)	82.47	63.06	70.63	76.27	82.73	77.19	54.90	64.11	74.07	76.43
	ScriptMix (AdapterFusion+)	83.40	66.94	72.11	80.01	83.79	78.83	55.69	66.95	74.43	77.24
	MAD-X + UniPELT	80.13	61.21	67.20	73.64	77.66	77.89	54.51	62.19	69.66	74.69
	ScriptMix (UniPELTFusion)	80.13	65.56	68.00	65.13	81.10	77.54	57.25	60.34	63.23	77.35
	ScriptMix (UniPELTFusion+)	81.33	69.60	71.19	72.18	83.08	79.06	58.63	63.68	66.67	78.06
TL	MAD-X	75.33	52.18	59.94	62.72	66.62	71.11	39.80	54.30	55.55	66.22
	ScriptMix (AdapterFusion)	83.53	59.68	70.51	74.48	81.86	81.64	53.53	64.11	68.02	76.53
	ScriptMix (AdapterFusion+)	84.60	61.37	72.55	76.78	84.50	83.16	53.14	66.31	71.14	80.82
	MAD-X + UniPELT	82.40	60.73	69.31	72.48	75.23	79.77	47.84	64.46	62.51	72.24
	ScriptMix (UniPELTFusion)	82.07	65.08	70.23	68.53	79.89	78.71	51.18	63.18	59.03	73.67
	ScriptMix (UniPELTFusion+)	82.33	67.50	71.87	73.05	80.70	81.29	55.49	67.02	64.18	76.94
	DEP (LAS %)	32-shot					16-shot				
		ug	cu	bxr	myv	am	ug	cu	bxr	myv	am
VA	MAD-X	31.33	23.31	20.03	23.66	32.17	25.85	24.90	15.07	23.28	26.33
	ScriptMix (AdapterFusion)	32.73	26.69	24.46	26.73	40.48	27.37	26.27	19.83	28.57	31.43
	ScriptMix (AdapterFusion+)	32.53	29.19	23.50	26.13	38.25	28.77	27.84	19.12	29.28	32.14
	MAD-X + UniPELT	32.80	24.68	23.54	26.92	35.36	27.02	26.08	18.98	28.13	28.16
	ScriptMix (UniPELTFusion)	34.40	25.00	24.62	23.25	37.84	27.95	26.47	20.61	22.66	29.29
	ScriptMix (UniPELTFusion+)	34.67	27.98	26.38	25.91	38.60	27.84	27.45	20.68	26.19	30.20
TL	MAD-X	29.27	25.65	21.31	23.37	37.18	21.87	25.88	16.99	15.37	34.08
	ScriptMix (AdapterFusion)	34.00	25.32	22.27	25.56	39.92	26.08	27.65	18.48	16.68	35.20
	ScriptMix (AdapterFusion+)	36.07	27.50	22.11	26.08	41.24	26.90	29.61	17.56	18.20	35.41
	MAD-X + UniPELT	33.73	24.35	24.46	25.90	35.82	29.01	26.47	19.62	17.55	33.78
	ScriptMix (UniPELTFusion)	35.80	24.84	25.26	26.16	35.82	27.72	26.67	18.19	14.58	33.78
	ScriptMix (UniPELTFusion+)	35.20	25.73	25.30	26.95	35.41	28.30	25.49	20.11	15.95	33.06

Table 7: Validation results of comparisons on few-shot setting

A Appendix

A.1 Results on Validation Split

We reveal the validation results of comparisons in Table 7, 8.

A.2 Standard Deviations of Experiments

We provide the standard deviations of experiments in Table 9, 10.

A.3 Visualizing the Complementarity

Figure 5 shows the visualization of weight vectors of all languages. As we described in section 4.3, ScriptMix benefits from both of two adapters, leveraging complementarity.

		DEP					POS				
		ug	cu	bxr	myv	am	ug	cu	bxr	myv	am
VA	MAD-X	67.58	73.88	42.55	55.28	69.27	91.10	94.99	86.18	90.26	92.86
	ScriptMix (AdapterFusion)	68.85	76.47	47.11	61.17	71.00	91.27	95.08	88.11	91.10	92.87
	ScriptMix (AdapterFusion+)	69.00	78.40	52.14	72.12	72.82	91.37	95.25	90.32	93.41	93.44
TL	MAD-X	67.62	74.30	40.64	56.12	66.92	91.12	95.14	85.28	90.91	92.33
	ScriptMix (AdapterFusion)	69.18	77.25	44.19	60.83	68.98	91.17	95.29	87.48	91.44	92.54
	ScriptMix (AdapterFusion+)	69.24	78.46	51.33	72.39	72.84	91.47	95.52	90.22	93.36	93.48

Table 8: Validation results of comparisons on setting with full train data

pos	32shot					16shot				
	ug	cu	bxr	myv	am	ug	cu	bxr	myv	am
MAD-X-VA	1.67	1.43	1.29	2.05	1.15	1.25	2.66	1.70	1.64	1.63
MAD-X-TL	0.61	1.41	1.90	2.19	1.25	0.66	2.36	2.26	1.07	1.78
max(MAD-X-VA/TL + UniPELT)	1.36	1.48	1.17	3.68	1.64	2.31	2.01	2.72	4.39	1.98
ScriptMix (UniPELTFusion)	1.57	1.53	2.14	3.03	1.53	1.36	0.71	3.42	5.11	2.74
ScriptMix (UniPELTFusion+)	2.50	0.82	2.11	3.35	1.30	1.38	1.21	2.65	4.45	2.31
ScriptMix (AdapterFusion)	0.88	0.78	0.83	2.19	2.04	0.75	3.82	1.69	2.49	2.05
ScriptMix (AdapterFusion+)	0.45	2.28	1.23	1.90	2.35	0.74	3.67	2.34	1.80	2.34
dep	32shot					16shot				
	ug	cu	bxr	myv	am	ug	cu	bxr	myv	am
MAD-X-VA	0.78	0.37	1.27	1.51	1.21	1.25	0.82	1.11	1.42	3.06
MAD-X-TL	0.65	0.22	1.29	1.21	1.76	0.74	0.70	1.52	1.37	1.85
max(MAD-X-VA/TL + UniPELT)	0.95	0.55	1.70	1.68	0.80	0.99	1.57	1.67	2.29	1.94
ScriptMix (UniPELTFusion)	0.79	0.91	3.12	0.94	1.73	2.20	1.45	2.16	2.65	2.86
ScriptMix (UniPELTFusion+)	0.68	0.89	2.25	1.79	2.31	0.68	1.60	1.71	2.52	3.13
ScriptMix (AdapterFusion)	0.85	1.30	1.94	2.54	2.19	0.82	1.24	1.98	3.01	2.61
ScriptMix (AdapterFusion+)	1.00	0.53	1.98	2.50	1.37	1.04	1.34	1.36	1.43	2.31

Table 9: Standard deviations of few-shot experiments.

	DEP					POS				
	ug	cu	bxr	myv	am	ug	cu	bxr	myv	am
max(MAD-X-VA, MAD-X-TL)	0.29	0.18	1.63	2.40	1.36	0.24	0.04	0.91	0.94	0.83
ScriptMix (AdapterFusion)	0.16	0.85	2.62	0.84	1.46	0.06	0.04	1.19	0.60	0.92
ScriptMix (AdapterFusion+)	0.41	0.40	1.31	1.71	1.00	0.12	0.12	1.47	0.75	0.76

Table 10: Standard deviations of experiments with full dataset.

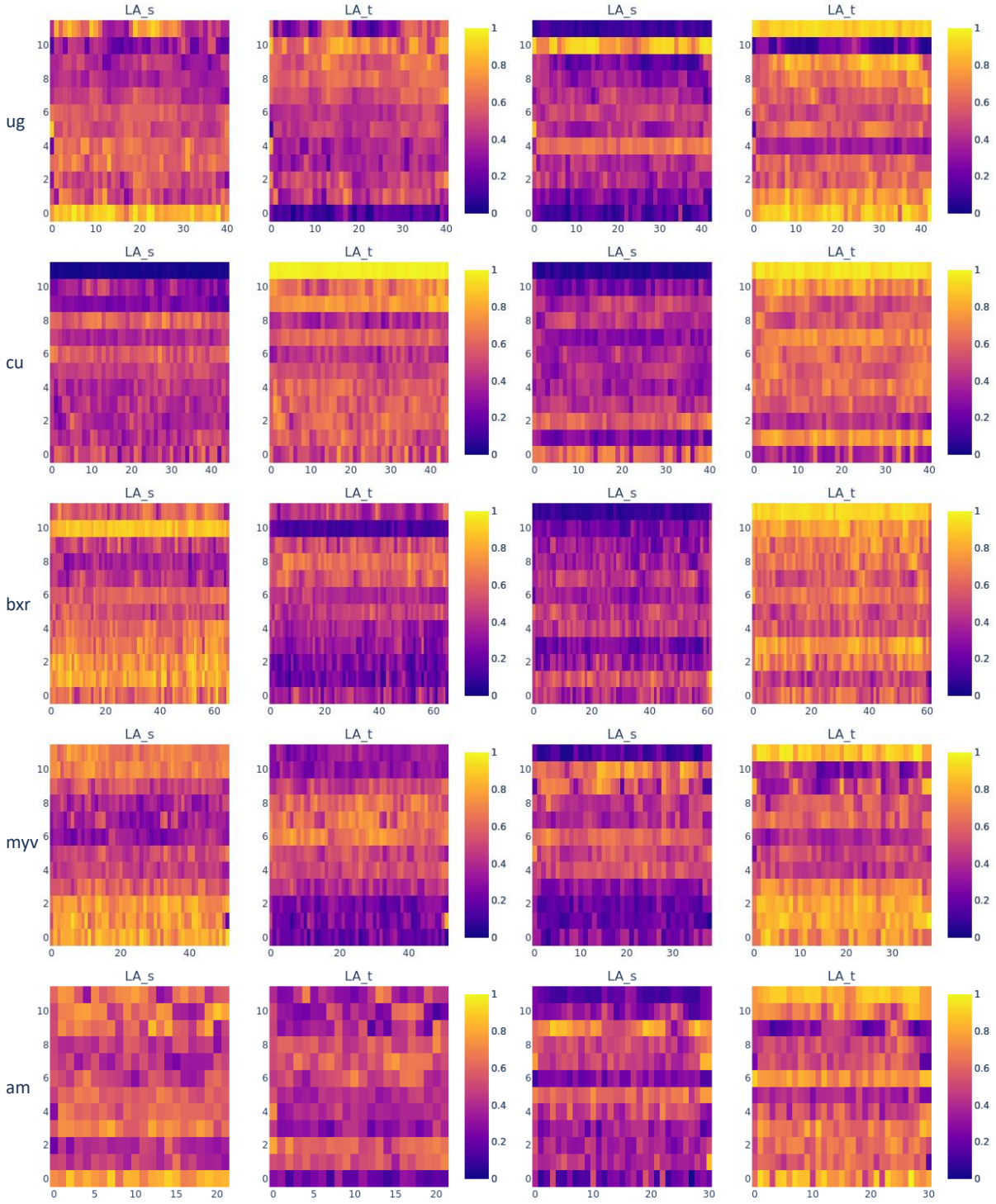

Figure 5: Heatmap of weights a of LA_s and LA_t varying different languages. Left: we inference with original POS examples. Right: We inference with transliterated POS examples.