ZGUL: Zero-shot Generalization to Unseen Languages using Multi-source Ensembling of Language Adapters

Anonymous ACL submission

Abstract

Our goal is to achieve high zero shot performance on low-resource languages (LRLs) that are unseen by the multilingual pre-trained language models like mBERT and XLM-R. A recent approach for handling LRLs is to use language adapters (LAs), but they are also unavailable for unseen languages. All existing works that study LAs for unseen languages train on only a single source language (English), and most use only the English adapter at test time. We believe that to achieve best zero-shot performance, we must make use of multiple (related) source languages/adapters at both training and test time. In response, we propose an architecture that performs both training-time and test-time ensembling of LAs. It also incorporates the typological properties of languages (encoded in existing language vectors) for further improvements. Extensive experiments and analysis over four language families demonstrate substantial improvements over standard fine tuning and other recent baselines on sequence labelling tasks.

1 Introduction

Our focus is zero-shot performance on *unseen languages* – those low-resource languages (LRLs) that are not seen by multilingual pre-trained language models (LMs) like mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) during training. A common approach is standard fine tuning (SFT): finetune the LM on task-specific training data from high-resource source languages, and apply the model zero-shot on unseen languages.

A more recent line of work to handle LRLs uses adapters (Houlsby et al., 2019; Rebuffi et al., 2017) – small modules that are inserted in every layer of transformer. In this approach, each *Language Adapter (LA)* is first pretrained monolingually using Masked Language Modeling. During task-specific fine-tuning, the source training language's LA is kept frozen and a separate trainable task adapter (TA) is stacked on top of LA. At inference time, the source language's LA is replaced by target language's LA to obtain zero-shot performance.

However this approach only works if an LA of target language has been pretrained, which is generally not the case for unseen languages. As a solution, all existing works simply train on English training data and most use just the English adapter at inference time (He et al., 2021; Pfeiffer et al., 2020c). We posit that this is less than ideal; for better performance, we should combine multiple source languages (ideally, related to target language) and their LAs, both at train and test time.

We propose ZGUL, **Z**ero-shot **G**eneralization to **U**nseen **L**anguages, which explores this hypothesis. It has three main components. First, it fuses LAs from source languages at train time by adapting AdapterFusion (Pfeiffer et al., 2021), which was originally developed for fusing multiple task adapters. This allows ZGUL to locally decide the relevance of each LA for each position in each layer. Second, ZGUL leverages syntactic and phonological properties of languages as additional information for computing (global) LA attention scores. For this, we make use of the URIEL database (Littell et al., 2017) of language features. Finally, ZGUL also implements the entropy-minimization based test-time tuning of adapter weights (Wang et al., 2021). This combination of train and test-time ensembling of pretrained LAs leads to a strong performance on our task.

We experiment with four language families - Slavic, Germanic, African and Indic on POS tagging and NER tasks. In each family, we train on 3-4 languages: English and 2-3 related languages for which task-specific training data and LAs are available. We find that training on multiple sources outperforms just training on English. Moreover, within multi-source training, ZGUL obtains substantial improvements compared to strong baselines like SFT and CPG (Üstün et al., 2020), on

languages unseen in mBERT/XLM-R, in purely zero-shot setting (and also few-shot setting). Further ablations show the importance of each component in ZGUL. We release our code and trained models[1] for further research.

We summarize our contributions as follows:

- We propose a strong method (ZGUL) to combine the existing LAs during training itself. To our knowledge, we are the first to attempt this in context of language adapters.
- ZGUL further incorporates test-time tuning of LA weights.
- ZGUL outperforms strong baselines, including SFT, for multi-source training for zero-shot transfer on LRLs unseen in mBERT/XLMR.
- ZGUL also achieves competitive results in the few-shot setting, where a small amount of unseen language training data is available.
- We find strong correlation between learned attention scores to adapters and the relatedness between high & low resource languages.

2 Related Work

Single-source Adapter Tuning: We build on Pfeiffer et al. (2020c), who introduce two phases of adapter training. 1. Pretraining LA for each language L_i: inserting an LA in each layer of transformer model $\mathcal{M}$ (denoted by $\mathcal{L}_i \circ \mathcal{M}$) and training on unlabeled data for language $\mathcal{L}_i$ using the MLM objective. 2. Training TA for a task T_j: stacking LA for source language L_{src} with TA for task T_j (denoted by $\mathcal{T}_j \circ \mathcal{L}_{src} \circ \mathcal{M}$), in which $\mathcal{T}_j$ and the task-specific prediction head are the only trainable parameters. $\mathcal{L}_{src}$ is replaced with $\mathcal{L}_{tgt}$ in inference, i.e. $\mathcal{T}_j \circ \mathcal{L}_{tgt} \circ \mathcal{M}$ is used. This paradigm uses only one LA for a given input sentence. Also, it works only if $\mathcal{L}_{tgt}$ is available. For unseen languages, only English adapter has been used at test time (He et al., 2021; Pfeiffer et al., 2020c).

Adapter Combination: Pfeiffer et al. (2021) introduce *AdapterFusion*, a technique that combines multiple pretrained TAs $\mathcal{T}_1, ...\mathcal{T}_n$ to solve a new target task T_{n+1}. It learns the attention weights of $\mathcal{T}_1, ...\mathcal{T}_n$ while being fine-tuned on the data for T_{n+1}. Vu et al. (2022) adapt this technique for fusing domains and testing on out-of-domain data. This technique has not been applied in the context of LAs so far. The recent release of 50 LAs in AdapterHub[2] enables studying this for LAs.

Recently, Wang et al. (2021) propose EMEA (Entropy Minimized Ensembling of Adapters) for efficiently combining multiple LAs at inference time. EMEA computes the entropy of the prediction at test time and differentiates it w.r.t. the LA attention scores (initialized uniformly) and updates those using Gradient Descent. In essence, EMEA adjusts the attention weights so as to give higher importance to the LA that increases the confidence score of the prediction at test time. However, training is still done using English as a single source.

Generation of LA using Shared Parameters: Üstün et al. (2020) apply Conditional Parameter Generation (CPG) (Platanios et al., 2018) for training on multiple source languages. They provide a typological language vector as input to a CPG module (called *CPGAdapter*) that generates an LA. The CPGAdapter is shared across all source languages, and trained from scratch for a given task. Since an LA is a function of input language's vector, this method can generalize directly to unseen languages. However, CPG is data intensive, since it learns the CPGAdapter parameters from scratch. We note that CPG comes under a broader category of hypernetworks that generate weights for a larger main network (Ha et al., 2016), which have also been recently explored successfully for mixing tasks (Karimi Mahabadi et al., 2021). We compare against CPG in our experiments.

Pretraining-based approaches: Other works have studied pre-training solutions to deal with LRLs. Ansell et al. (2021) propose MAD-G, which pretrains CPG on 95 languages' Wikipedias, using Masked Language Modeling (MLM). They further fine-tune it on task data (by inserting a TA) and test on LRLs in zero-shot setting. A recent work by Pfeiffer et al. (2022) incrementally pretrains on unlabeled data of new languages by inserting language-specific modules in order to incorporate new languages. Our paper takes a complementary view, and eschews pre-training completely, and instead, focuses on fine-tuning using task-specific training data of multiple source languages.

3 Model for Ensembling of Adapters

Our goal is to effectively combine a set of source language adapters both at train and test time for

[1] https://anonymous.4open.science/r/ZGUL

[2] https://adapterhub.ml/explore/text_lang/

zero-shot generalization to unseen languages. In this paper we focus on those unseen languages whose scripts are seen by the pre-trained language model. Our approach can be divided in two high level parts: ensembling at train time, and ensembling at inference time.

3.1 Ensembling during Training

During training time, we make use of attention mechanism inspired from combination of task adapters explored in Pfeiffer et al. (2021). Whereas they create an ensemble of task adapters, our focus is on combining *language* adapters. Additionally, we note that there is useful information available in typological language vectors (Littell et al., 2017), which we wish to exploit. We achieve this objective by designing two sub-components in our architecture and later combining them (for each layer), which we describe next (see Figure 1).

Token Based Attention ($Fusion$): This sub-network computes the attention weights for source LAs using the output of the previous transformer layer (which, in turn, depends on the input tokens) as its query, and using the language adapter outputs both as the key as well as the value. Mathematically, the output of transformer layer $l-1$ passed through the feed forward of layer l becomes the query Q^l, and the individual adapter outputs following this become keys (and values) K^l (and V^l), the attention weights of individual adapters are computed using dot product between $W_q^l Q^l$ and $W_k^l K^l$ followed by softmax.

$$\alpha_F^l = Softmax((W_q^l Q^l)^T (W_k^l K^l))$$

The output of fusion is given by:

$$o_F^l = \alpha_F^l (W_v^l V^l)$$

where W_q, W_k and W_v are the learnable parameters.

Language vector based Attention ($Lang2vec$): This sub-network computes the attention weights for source adapters using the input language (of the token) as the query, language vectors of the source languages as the keys, and the language adapter outputs as the value. Mathematically, given the language vector LV_{inp} as the query, and the source language vectors $LV_1, LV_2, \cdots LV_r$ as the keys, we perform Bahdanau Attention over them, to get the attention weights over the LA outputs.

Here the language vectors are obtained by passing the language features LF[3] (each entry being 0 or 1) through a 1-layer MLP (shared across layers) as:

$$LV_{inp} = MLP(LF_{inp})$$

Attention scores are given by:

$$\alpha_L^l = (LV_{inp})^T W_L^l (LV_{1:r}),$$

Where W_L^l is a learnable matrix for each layer l (not shared across 12 layers).

The output of Lang2vec module is given by:

$$o_L^l = \alpha_L^l (W_v V^l)$$

We note that in $Fusion$, attention scores are learnt differently across tokens, layers and examples in a given language. However, in $Lang2Vec$, attention scores will be same for a given language across all tokens in a layer, but they will be different across the 12 layers due to learnable matrix W_L^l.

Combining the two ensembling modules: We pass the input sentence through both the networks, and receive the outputs o_F and o_L, corresponding to the token based and language vector based attention, respectively. We concatenate these two vectors and pass through a fully connected layer. The output of this linear layer, denoted as $o_{LA}^{(l)}$, goes as input to task adapter to get the final output $o_{final}^{(l)}$.

$$o_{LA}^{(l)} = LinearLayer^{(l)}(o_F^{(l)} \oplus o_L^{(l)})$$
$$o_{final}^{(l)} = TA^{(l)}(o_{LA}^{(l)})$$

This process is repeated for each layer l in the transformer architecture. We note that the LAs are kept frozen throughout the training process, while only the TA and other parameters described in $Fusion$ and $Lang2Vec$ modules being trainable.

3.2 Ensembling during inference

Wang et al. (2021) proposed inference time Entropy Minimization (EM) algorithm to adjust the adapter weights (initializing from uniform weights). In our case, since we have learnt the weights during training itself, we seek to further leverage the EM algorithm in our framework. Since we have two different networks in our model – token based attention and language vector based attention, we

[3]We use 103-dimensional syntactic features available on https://github.com/antonisa/lang2vec

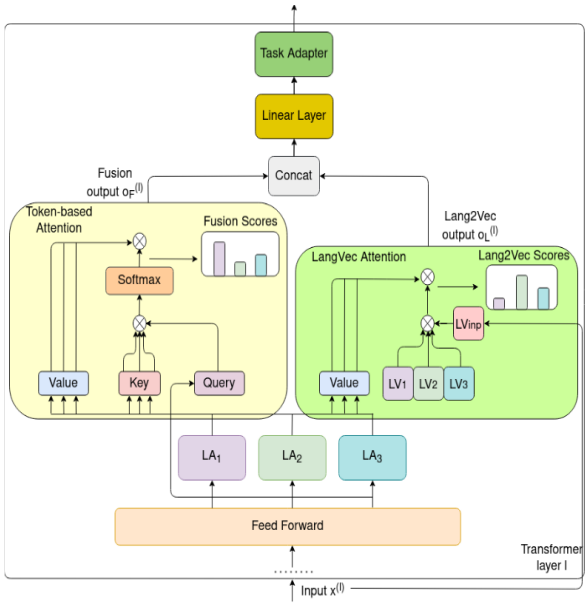


Figure 1: Fusion Network (left) and Lang2Vec Network (right) outputs are concatenated and sent to a Linear layer followed by a TA in every layer l of the transformer

propose a *two-step* EM algorithm. We calculate entropy during inference and alternately update the adapter weights for both the networks.

Unlike EMEA which ties all the attention weights for a given layer, we maintain separate tokenwise learnable attention weights in each layer. This gives greater representational flexibility to our model. Further, while EMEA initializes the attention weights uniformly, we initialize them with the ones obtained using a forward pass from ZGUL tained in a multi-source fashion. This helps our model start from an informed weight combination for the language adapters; EMEA has no such knowledge of a good starting point since it is trained on a single source. The detailed algorithm is described in Algo 11. The hyperparameters lr and T are tuned on the dev set for each target language with grid search details (ref. Appendix A).

4 Experiments

In our experiments, we set out to answer the following questions. (1) How does ZGUL perform in a zero-shot setting compared to the other baselines on unseen languages? What is the incremental contribution of ZGUL components for performance on LRLs? (2) Are LA attention weights learnt by ZGUL interpretable, i.e., whether genetically/syntactically more similar source languages get higher attention scores? (3) How does ZGUL's performance vary in the *few-shot* setting, where a few training examples of the target language are shown to the model for fine-tuning, compared to other baselines, especially SFT-M? (4) Do ZGUL's benefits carry over to larger pre-trained models such as XLM-Roberta? We next describe our datasets, evaluation metrcis, methodology followed by our experimental findings.

4.1 Datasets, Tasks and Baselines

Datasets and Tasks: We experiment with 15 languages that are unseen for mBERT, from four language families: Slavic, Germanic, African and Indic. We choose either of two sequence labeling tasks: named entity recognition (NER) or part-of-speech (POS) Tagging, based on available test sets for unseen languages. For Slavic and Germanic, we use POS tagging dataset UDPOS. For African and Indic, we use NER datasets MasakhaNER and PAN-X respectively. PAN-X and UDPOS datasets have been obtained from XTREME Repository (Hu et al., 2020) while MasakhaNER is obtained from Adelani et al. (2021).[4]. We choose training languages from each family that have pre-trained adapters available [5] We justify our language grouping using the phylogenetic trees (Cole and Siebert-Cole, 2022a,b) for Indo-European and African languages (we combined Niger-Congo and Afro-Asiatic families following Adelani et al. (2021)). The details of training and test languages, as well as corresponding task for each family are described in Table 1.

Baselines: We experiment with two sets of baselines. In the first set, the baselines make use of only English as the source language during training. The baselines here include finetuning the mBERT model (SFT), using English adapter in training and inference (Adapter-En), replacing English adapter with one related LA at inference time (Adapter-Related), and EMEA: ensembling multiple LAs at inference time via entropy minimization (Wang et al., 2021). In the second set, we compare with algorithms that make use of multiple source languages (belonging to a family), in addition to English. We compare against CPG and standard fine tuning (SFT-M), trained on all source languages.

Evaluation Metric: We report the micro-F1 score evaluated on each token using seqeval toolkit (Nakayama, 2018), which is consistent with the one used in previous research (Wang et al., 2021)

[4] https://github.com/masakhane-io/masakhane-ner

[5] We do not have a pre-trained LA for Urdu; we instead use the LA for Arabic, which shares the same script as Urdu.

Algorithm 1 EM algorithm during Inference

Input: Our model's scores α^0= (α^0_F,α^0_L), test data x, learning rate lr, update steps T
Output: Prediction $\hat{y}$

function EMEA++()
  for $t \leftarrow 0$ to $T-1$ **do**
    $\beta^t \leftarrow Softmax(\alpha^t)$ ▷ Normalize the LA scores
    $H(x,\alpha) \leftarrow Entropy(TA \circ L_{wavg}(h,\beta^t) \circ M)$ ▷ Compute Entropy
    $g^t_F = \bigtriangledown_\alpha H(x,\alpha^t_F)$ ▷ Compute Token Attention gradient
    $\alpha^{t+1}_F \leftarrow Update(\alpha^t_F, g^t_F)$ ▷ Update Token Attention weights
    $g^t_L = \bigtriangledown_\alpha H(x,\alpha^t_L)$ ▷ Compute LangVec Attention gradient
    $\alpha^{t+1}_L \leftarrow Update(\alpha^t_L, g^t_L)$ ▷ Update LangVec Attention weights
    $\alpha^{t+1} = (\alpha^{t+1}_F, \alpha^{t+1}_L)$
  end for
  $\alpha^T \leftarrow Softmax(\alpha^T)$ ▷ Final Normalize
  $\hat{y} \leftarrow Predict(TA \circ L_{wavg}(h,\alpha^T) \circ M))$ ▷ Compute Prediction
end function

Family	**Train set***	**Test set**	**Task**
Germanic	German(De), Islandic(Is)	Faroese(Fo), Gotham(Got), Swiss German(Gsw)	POS
Slavic	Russian(Ru), Czech(Cs)	Pomak(Qpm), Upper Sorbian(Hsb) Old East Slavic(Orv), Old Church Slavonic(Cu)	POS
African	Amharic(Amh) Swahili(Swa), Wolof(Wol)	Hausa(Hau), Igbo(Ibo), Kinyarwanda(Kin) Luganda(Lug), Luo(Luo), Nigerian Pidgin(Pcm)	NER
Indic	Hindi(Hi), Bengali(Bn) Urdu(Ur)	Assamese(As), Bhojpuri(Bh)	NER

Table 1: Language families and their corresponding train, test languages, and tasks. * English (En) was added to train set in each case.

and is available as a standard evaluation script in the Google Research's xtreme repo.[6]

4.2 Results: Zero Shot Transfer

Table 2 presents our experimental findings for Germanic and Slavic group of languages, and Table 3 for African and Indic families. For most unseen LRLs, we observe ZGUL's consistent gains compared to other baselines. For the task of POS, we observe a respectable gain of 2.5 points in ZGUL's average performance on Germanic, and a marginal improvement of 0.7 points for Slavic family in the average scores, over closest baselines in each case. For the task of NER, we observe a decent gain of 2.8 pts for the Indic family, and a marginal gain of 0.8 pts for the African family, over the closest baselines. We observe that for POS, our competitive baseline is CPG, while for NER it's SFT-M. This is not surprising since CPG needs a lot of training data to optimize Üstün et al. (2020) due to a large number of parameters in the model, and our NER datasets are much smaller in size compared to those for POR (ref. Appendix). We also note that baselines which train on a single source perform much worse in each case, highlighting the importance of multi-source training for our task, the gain being particular impressive for Indic family (24 pts).

Last three rows in Tables 2,3 present the findings of our ablation studies, where examine the effect different kinds of attention mechanisms in ZGUL , as well as the impact of entropy minimization during inference (ref. Section 3). We see that each component of ZGUL has a positive impact on its performance for each of the language family, as far as average scores are concerned, experimental higlighting the importance of each of the components in our architecture. For individual languages as well, we see an improvement in performance due to each component, except for a few cases (Assamese and Luo), where EM marginally hurts the performance.

Overall, we see that ZGULbeats all the baselines for each of the language families, with the gain ranging from 0.7 pts to 2.8 pts in average scores, and ZGUL or one of its variants obtain the best scores in 11 out of 15 languages, compared to

[6] https://github.com/google-research/xtreme/blob/master/third_party/run_tag.py

	Germanic				Slavic				
	Fo	Got	Gsw	**Avg**	Qpm	Hsb	Orv	Cu	**Avg**
SFT	72.4	13.2	55.1	46.9	40	64.8	56.5	28	47.3
Adapter-En	71.6	15.6	57.1	48.1	41.8	63.2	54.3	29.5	49
Adapter-Rel	72.8	15.2	57	48.3	39.3	60.2	57	32.2	49.8
EMEA	74	14.8	55.3	48	42.1	63.1	56.4	32	50.5
SFT-M	**77.3**	16.8	61.8	52.0	46.8	75.6	63.7	34.7	55.2
CPG	77	16	63.6	52.2	46.9	76.7	**64.1**	**35.6**	55.8
ZGUL	77.1	**20.8**	**65.6**	**54.5**	**50.4**	**77**	63.6	34.9	**56.5**
ZGUL w/o EM	76.8	15.8	63	51.9	49.4	76.4	63.3	33.6	55.7
ZGUL w/o L	76.9	17.8	61.3	52	48.8	76.5	63.2	**35.6**	56
ZGUL w/o F	76.9	18.7	62.4	52.7	49.5	76.5	63	35.2	56.1

Table 2: F1 of POS Tagging Results for Germanic and Slavic language families

	African							Indic		
	Hau	Ibo	Kin	Lug	Luo	Pcm	**Avg**	As	Bh	**Avg**
SFT	43.2	47.1	49.1	47.5	29	64.4	46.7	32.1	35.7	33.9
Adapter-En	41.3	51.2	46	49.8	29.2	64.5	47	33	44.8	38.9
Adapter-Rel	39.1	49.1	46.3	48.7	29.7	65.1	46.3	42.4	46.9	44.7
EMEA	42	52.9	50.2	50.6	30.7	65.5	48.7	41.6	47.6	44.6
SFT-M	51.3	55.7	**57.9**	**56.0**	36.0	65.0	53.7	70.8	61.4	66.1
CPG	49	51.1	55.1	54	34.2	**65.7**	51.5	62	63.3	62.7
ZGUL	**56.4**	**56.5**	56.2	55	37.2	**65.7**	**54.5**	71	**66.7**	**68.9**
ZGUL w/o EM	55.7	55.9	55.1	54.6	**37.5**	**65.7**	54.1	**71.4**	63.1	67.3
ZGUL w/o L	49	54.7	57.3	53.5	35.5	65.5	52.6	58.8	62.2	60.5
ZGUL w/o F	53.1	54.1	57.7	55.7	36.6	65.6	53.8	67.2	61.7	64.5

Table 3: F1 of NER Results for African and Indic language families

SFT-M and CPG, which win in 4 and 2 languages, respectively (two of these being a tie).

4.3 Interpreting Attention Scores

Language Family	Lang2Vec	Fusion
Indic	0.98	-0.19
African	0.92	0.4
Germanic	0.52	0.43
Slavic	0.96	0.92

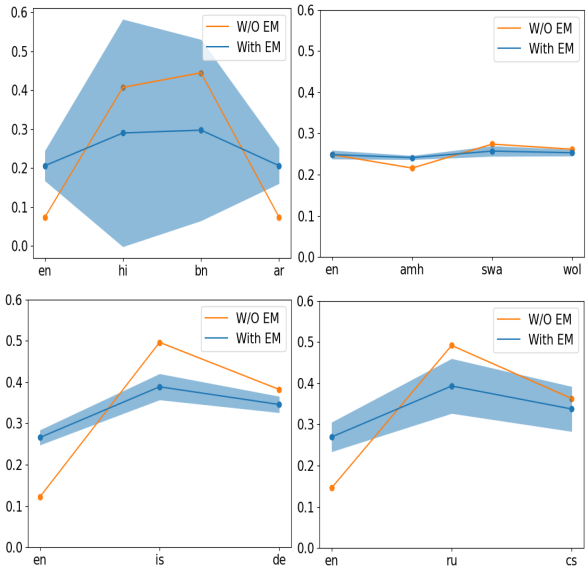
Table 4: Correlation between adapter weights given by ZGUL and actual syntactic-genetic (averaged) similarity of source-target pairs for both Lang2vec and fusion

Table 4 presents the correlation[7] between the adapter weights that ZGUL assigns to different language adapters and the actual relatedness of the languages measured as the average of similarity measured using genetic and syntactic features of the languages.[8]. Table 2b presents the pairwise similarity of languages computed using above method.

We note that our model is fed only the syntactic features and not given the genetic features while training. Despite this, we see that there is a very high correlation between language based attention weights, and language similarity as computed above. This leads us to believe that the language based attention weights computed by our model have grounding in language/linguistic similarity. On the other hand, we see the correlation between fusion based attention weights and language similarity is quite low. We hypothesize that this is because while language based attention explicitly uses the knowledge of the language, the fusion based attention weights make use of token level

[7] We have used the Pearson Product-Moment Correlation to determine the correlation.

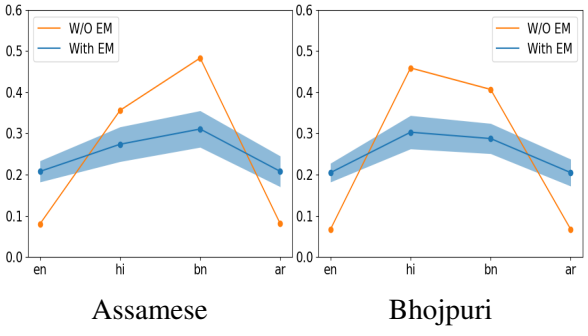
[8] refer to Section D for details



(a) L2V based adapter weights(on y axis) of each of the language families. Shown both weights, with and w/o EM. Indic (Top Left), African(Top Right), Germanic(Bottom Left), Slavic(Bottom Right)

Tgt/Src	Eng	Amh	Swa	Wol	Rus	Cze	Isl	Ger	Ara	Hin	Ben
Bho	0.21	0.18	0.08	0.11	0.29	0.26	0.23	0.26	0.15	**0.47**	0.41
Hau	**0.27**	0.24	0.22	**0.27**	0.2	0.14	0.24	0.23	0.2	0.19	0.11
Ibo	0.19	0.12	**0.41**	0.3	0.14	0.14	0.19	0.16	0.15	0.08	0.09
Kin	0.11	0.1	**0.48**	0.21	0.1	0.14	0.14	0.09	0.13	0.06	0.08
Lug	0.11	0.11	**0.57**	0.25	0.11	0.16	0.16	0.11	0.15	0.06	0.1
Luo	0.18	0.17	0.23	**0.25**	0.15	0.16	0.2	0.17	0.18	0.13	0.08
Ass	0.14	0.14	0.1	0.06	0.22	0.23	0.2	0.22	0.14	0.35	**0.56**
Qpm	0.33	0.19	0.16	0.23	**0.48**	0.44	0.28	0.3	0.19	0.28	0.22
Hsb	0.26	0.21	0.2	0.19	**0.51**	0.46	0.27	0.28	0.16	0.27	0.23
Orv	0.26	0.21	0.2	0.19	**0.51**	0.38	0.27	0.28	0.16	0.27	0.23
Chu	0.26	0.21	0.2	0.19	**0.51**	0.38	0.27	0.28	0.16	0.27	0.23
Fao	0.34	0.15	0.13	0.19	0.28	0.3	**0.64**	0.42	0.13	0.19	0.22
Got	0.26	0.14	0.16	0.19	0.26	0.27	**0.35**	0.34	0.15	0.21	0.21
Gsw	0.38	0.17	0.12	0.17	0.28	0.3	0.38	**0.51**	0.13	0.24	0.25

(b) Average of syntactic and genetic similarity of languages. See Figure 5 for more details



(c) L2V based adapter weights(on y axis) expanded language-wise for the indic family

Figure 2: Demonstrating the relation between language similarity and adapter weights

information, which may or may not capture the language specific information explicitly. To further analyze the impact of the correlation between language based weights and the language similarity, we plotted the curves 2a depicting average weights assigned to each training language. The 'with EM' curve depicts the variation for ZUGL model, while w/o EM curve depicts the variation for the model which does not use EM during inference. We see that the weight assignment for w/o EM is more extreme, which is moderated by doing inference with EM. The shaded region shows the standard deviation around the mean across language families.

In figure 2c, the language-wise division of the results are shown for the indic family. These results are consistent with the fact that Assamese is closer to Bengali than Hindi, while Bhojpuri is closer to Hindi than Bengali. This can be re-verified from Fig 2b.

4.4 Few-Shot Performance

In this experiment, we take the trained ZGUL and other multi-source models, i.e., CPG, SFT-M, and further fine-tune them for a few labeled examples from the training set of the target language. We implement this for all those languages in our set, whose training and dev sets are available (12 out of 15). We use development set for tuning hyper-parameters (details in Appendix A). We sample training bins of sizes 10, 30, 70 and 100 examples.

We observe in Figure 3 that ZGUL is able to scale quite well for all the language families, maintaining its dominance over the baselines in each case. The relative ordering of baselines, i.e. CPG outperforming SFT-M for Slavic and Germanic (POS), and SFT-M outperforming CPG for African and Indic (NER), is also maintained. As expected, we see that there is an initial quick jump in performance for all the three models with addition of first few examples for fine-tuning, after which the slope decreases in most cases (Slavic family being an exception). The plateau is not reached in the learning curve for either of the language families, showing that adding more examples of the test language would likely result in improvement of all the models, albeit at a smaller pace. We present the few-shot curves for each of the 12 languages in Appendix C. We observe that the curves are more or less parallel for most languages, with initial zero-shot gap maintained even after additional fine tuning.

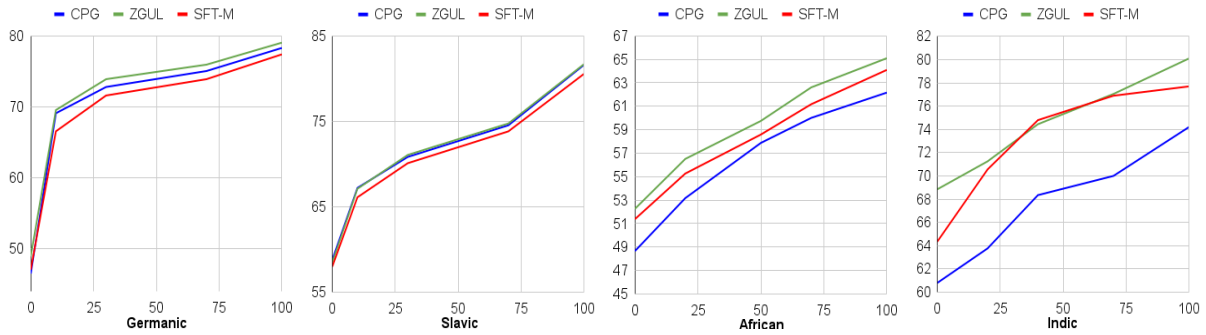


Figure 3: Few-shot F1 scores averaged over languages in a family for various few-shot bins

4.5 Robustness with XLM-Roberta

We wanted to examine whethter our findings related to ZGUL's superior performance over the existing baselines carry over to other language models. Specifically, we trained ZGUL and other multi-source baselines on XLM-R Base model available on Adapterhub (Pfeiffer et al., 2020a) for Germanic and Indic language families.We had all the adapters for languages in these two families, except for Bengali (Indic) which we eliminated during training. The test set of languages was kept the same for the original set of experiments in each case. Table 5 presents our findings. ZGUL beats both the baselines on both the language families, with a gain of 1.6 pts on the Indic family, and a gain of 0.9 pts on the Germanic family, compared to its closest competitor. This clearly confirms the finding that ZGUL's gains are not restricted to a specific choice of language model. Surprisingly, the overall performance of XLM-R based language adapters is worse compared to those trained using m-Bert (Table 2, 3), an examining the causes for this is a direction for future research.

	Indic			Germanic			
Model	As	Bh	**Avg**	Fo	Got	Gsw	**Avg**
SFT-M	**60.4**	60.5	60.5	78	15.4	54.9	49.4
CPG	55.7	60.3	58	**78.1**	18.3	**57.6**	51.3
ZGUL	59.3	**64.9**	**62.1**	77.8	**21.7**	57.1	**52.2**

Table 5: F1 of NER Results for Indic (trained on En,Hi,Ur) and POS for Germanic models (trained on En, Is, De) with XLM-R-Base Adapters

4.6 Qualitative Analysis

We also performed some qualitative analysis of where does the actual gain of ZUGL come from? To do this, for both POS and NER tasks, we examined label-wise performance of our model, vis-a-vis the baselines. For POS, we see that ZUGL does quite well on 'NOUN' which has a huge support, resulting in overall better performance for ZUGL. ZUGL does somewhat worse on labels such as 'CONJ'. Similarly, for NER, ZUGL is able to do well on labels such as 'LOC' and 'PER', resulting in overall improved performance over its competitors. We refer to the Appendix E for further details. Carefully examining the reasons behind improved performance on certain labels (while worse performance on others) is a direction for future work.

5 Conclusion and Future Work

We present ZGUL, a novel method of combining the existing pretrained language adapters for training over multiple languages. This is performed by fusing the language adapters at train time to compute one attention score, along with language vectors to compute a second attention score, which are combined for effective training. Entropy minimization is carried out at test time to further improve the attention scores over the language adapters. Our model obtains strong performance for languages unseen by pre-trained language models. Additionally, ZGUL also obtains benefits in the few shot-setting, when a small amount of training data may be available for the target language. Additional experiments confirm correlation between learned attention scores and phylogenetic relatedness between languages. Directions for future work include experimenting with additional language families, testing on tasks beyond those experimented in this work, and analyzing what happens if model is trained on languages which are quite different from those used during testing.

6 Limitations

We present the trainable parameters of ZGUL and other baselines in table 7. We are able to reduce the parameters by more than 75% compared to SFT-M. This makes our training cheaper in terms of gradient calculations and parameter updates. On the other hand, we do incur the overhead during inference since adapters need to be added in each layer. In addition,the entropy minimization step increases no. of forward passes (i.e. no. of update steps T) at the cost of performance (similar trade-off observed by (Wang et al., 2021)). Moreover, a crucial benefit with ZGUL is that it uses the pre-trained LAs which are reusable modules needing pre-training only once per language. Also, our work involves no pre-training on the unlabeled data of low resource languages (E.g. Wikipedia of those languages). So, we present a much cheaper alternative for improving low resource performance compared to other approaches requiring explicit pre-training over multiple languages. (Ansell et al., 2021; Pfeiffer et al., 2022).

References

David Ifeoluwa Adelani, Jade Abbott, Graham Neubig, Daniel D'souza, Julia Kreutzer, Constantine Lignos, Chester Palen-Michel, Happy Buzaaba, Shruti Rijhwani, Sebastian Ruder, Stephen Mayhew, Israel Abebe Azime, Shamsuddeen H. Muhammad, Chris Chinenye Emezue, Joyce Nakatumba-Nabende, Perez Ogayo, Aremu Anuoluwapo, Catherine Gitau, Derguene Mbaye, Jesujoba Alabi, Seid Muhie Yimam, Tajuddeen Rabiu Gwadabe, Ignatius Ezeani, Rubungo Andre Niyongabo, Jonathan Mukiibi, Verrah Otiende, Iroro Orife, Davis David, Samba Ngom, Tosin Adewumi, Paul Rayson, Mofetoluwa Adeyemi, Gerald Muriuki, Emmanuel Anebi, Chiamaka Chukwuneke, Nkiruka Odu, Eric Peter Wairagala, Samuel Oyerinde, Clemencia Siro, Tobius Saul Bateesa, Temilola Oloyede, Yvonne Wambui, Victor Akinode, Deborah Nabagereka, Maurice Katusiime, Ayodele Awokoya, Mouhamadane MBOUP, Dibora Gebreyohannes, Henok Tilaye, Kelechi Nwaike, Degaga Wolde, Abdoulaye Faye, Blessing Sibanda, Orevaoghene Ahia, Bonaventure F. P. Dossou, Kelechi Ogueji, Thierno Ibrahima DIOP, Abdoulaye Diallo, Adewale Akinfaderin, Tendai Marengereke, and Salomey Osei. 2021. MasakhaNER: Named entity recognition for African languages. *Transactions of the Association for Computational Linguistics*, 9:1116–1131.

Alan Ansell, Edoardo Maria Ponti, Jonas Pfeiffer, Sebastian Ruder, Goran Glavaš, Ivan Vulić, and Anna Korhonen. 2021. MAD-G: Multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Theodor Cole and Erika Siebert-Cole. 2022a. Family tree of languages – part i: Indo-european (2022).

Theodor Cole and Erika Siebert-Cole. 2022b. Family tree of languages – part iii: African, dravidian, uralic, caucasian, afro-asiatic.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard 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.

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.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.

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 *International Conference on Machine Learning*, pages 2790–2799. PMLR.

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 *International Conference on Machine Learning*, pages 4411–4421. PMLR.

Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. 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 565–576, Online. Association for Computational Linguistics.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL

and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. Association for Computational Linguistics.

Hiroki Nakayama. 2018. seqeval: A python framework for sequence labeling evaluation. *Software available from https://github. com/chakki-works/seqeval*.

Jonas Pfeiffer, Naman Goyal, Xi Victoria Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. 2022. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3479–3495. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. 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.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020): Systems Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020b. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020c. 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.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435, Brussels, Belgium. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Thuy-Trang Vu, Shahram Khadivi, Dinh Phung, and Gholamreza Haffari. 2022. Domain generalisation of NMT: Fusing adapters with leave-one-domain-out training. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 582–588.

Xinyi Wang, Yulia Tsvetkov, Sebastian Ruder, and Graham Neubig. 2021. Efficient test time adapter ensembling for low-resource language varieties. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 730–737.

A Hyperparameter and Implementation Details

We use AdapterHub [9] (Pfeiffer et al., 2020b) for all our experiments and analysis. We mention the hyperparameter grid for SFT and adapter-based (CPG and ZGUL) models in table 6. For all experiments, we report the average of 3 training runs of the models (with 3 different random seeds). We tune the Adapter Reduction Factor (RF) in the range of 2,3 and 4 but generally fine 3 or 4 the best for both CPG and ZGUL for all datasets.

Hyperparameter	SFT	Adapter Based
Learning Rate	{2e-5, 3e-5, 5e-5}	{5e-5, 1e-4}
No. of epochs	10	10
Reduction Factor	NIL	{2, 3, 4}
Batch Size	{16, 32}	{16, 32}
EM Steps	NIL	{1, 5, 10}
EM LR	NIL	{0.05, 0.1, 0.5, 1.0, 5.0}
Few-shot LR	{1e-5, 5e-5, 1e-4}	{1e-5, 5e-5, 1e-4}
Few-shot epochs	{1, 5, 10}	{1, 5, 10}
Few-shot Batch Size	{1, 4, 8}	{1, 4, 8}

Table 6: Hyperparameter grids for ours models

A.1 Trainable Parameters

We present the trainable parameters for each model in table 7

Model	# params
SFT-M	177276690
CPG w RF=4	251778898
CPG w RF=3	252959314
ZGUL w RF=4	39491053
ZGUL w RF=3	40671469

Table 7: Trainable Parameters of all models

B Class-wise F1 scores

Table 8 and Table 9 show the classwise F1 scores for each of the tasks. The scores are averaged over all languages in each task.

We have used the seqeval[10] framework for evaluating all the models, which is consistent with the previous works and used by XTREME[11]. Seqeval removes the 'B' and 'I' prefixes of the labels, hence the Table 9 has only 3 classes.

[9] https://github.com/adapter-hub
[10] https://github.com/chakki-works/seqeval
[11] https://github.com/google-research/xtreme

Class	ZGUL	CPG	SFT-M	Support
PART	**29**	26	26	831
CONJ	64	**65**	64	9000
ADJ	44	44	44	8768
ADP	**79**	77	75	9960
ADV	28	28	28	5830
VERB	**51**	**51**	50	14385
DET	40	**41**	37	3027
INTJ	10	3	**14**	211
NOUN	**59**	57	58	18957
PRON	**41**	**41**	40	7068
PROPN	**55**	53	53	4506
NUM	**58**	57	57	1508
PUNCT	**62**	61	62	7670
AUX	**42**	41	40	2800
SYM	12	**13**	12	33
X	4	4	2	228
Micro-F1	**56**	54	54	94782

Table 8: Classwise F1-scores for POS task. Averaged over all languages

Class	ZGUL	SFT-M	CPG	Support
DATE	21	**22**	22	623
LOC	**55**	53	50	1643
ORG	**53**	**53**	48	1034
PER	**67**	64	64	1483
Micro-F1	**57**	55	53	4783

Table 9: Classwise F1-scores for NER task. Averaged over all languages

C Language-wise Few Shot Performance

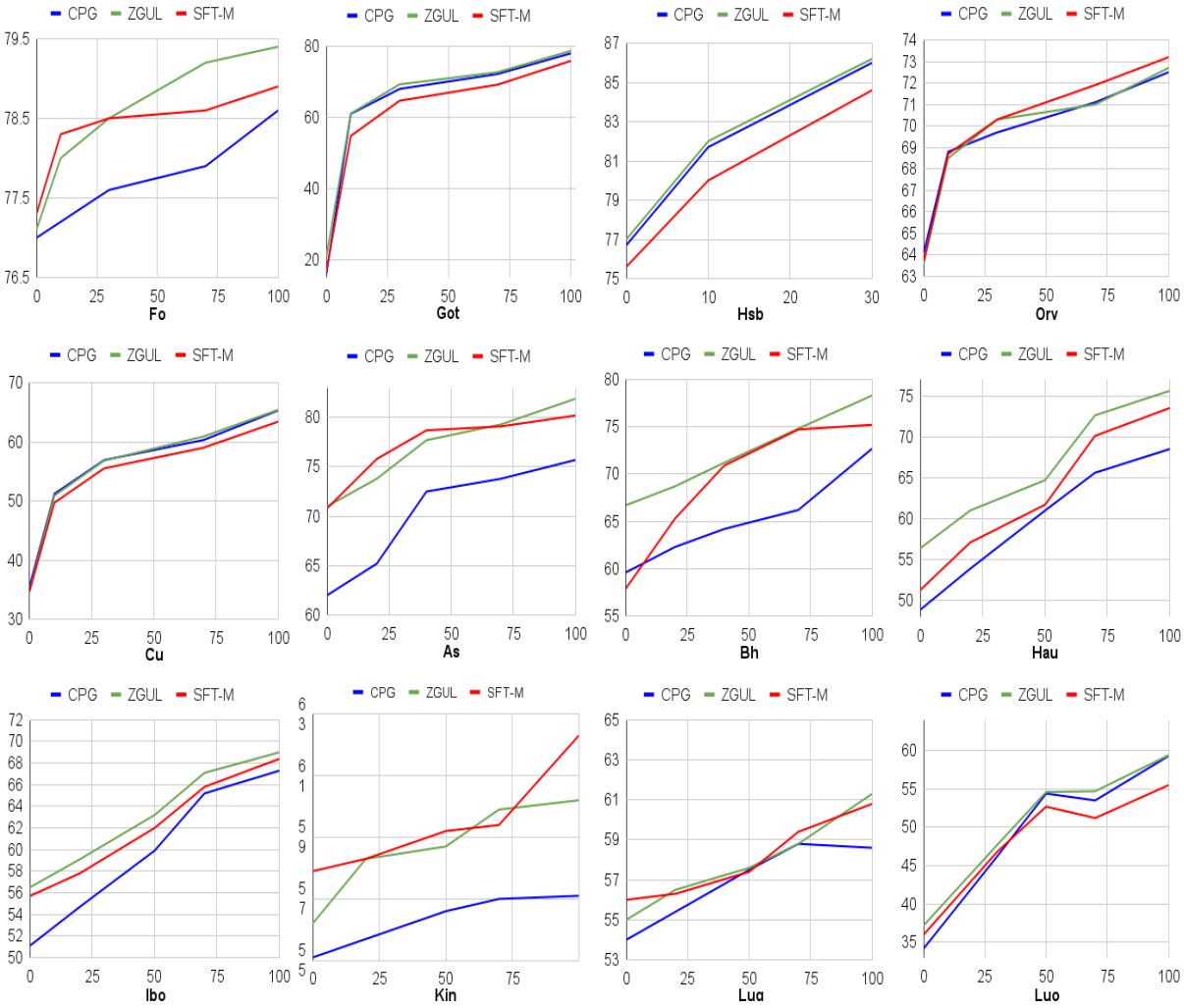


Figure 4: Caption

D Quantifying similarity between source and target languages

We use the phlogenetic and syntactic distances between source and target languages for reference.[12]

[12] https://github.com/antonisa/lang2vec

Target/Source	Eng	Amh	Swa	Wol	Rus	Cze	Isl	Ger	Ara	Hin	Ben
Bho	0.1	0	0	0	0.2	0.14	0.14	0.14	0	**0.37**	**0.37**
Hau	0	**0.12**	0	0	0	0	0	0	0	0	0
Ibo	0	0	**0.5**	0.17	0	0	0	0	0	0	0
Kin	0	0	**0.62**	0.2	0	0	0	0	0	0	0
Lug	0	0	**0.62**	0.2	0	0	0	0	0	0	0
Luo	0	0	0	0	0	0	0	0	0	0	0
Ass	0.1	0	0	0	0.2	0.14	0.14	0.14	0	0.37	**0.75**
Qpm	0.14	0	0	0	**0.43**	**0.43**	0.14	0.14	0	0.14	0.12
Hsb	0.1	0	0	0	**0.6**	0.57	0.14	0.14	0	0.12	0.12
Orv	0.1	0	0	0	**0.6**	0.43	0.14	0.14	0	0.12	0.12
Chu	0.1	0	0	0	**0.6**	0.43	0.14	0.14	0	0.12	0.12
Fao	0.3	0	0	0	0.2	0.14	**0.86**	0.43	0	0.12	0.12
Got	0.2	0	0	0	0.2	0.14	**0.29**	**0.29**	0	0.12	0.12
Gsw	0.4	0	0	0	0.2	0.14	0.43	**0.57**	0	0.12	0.12

(a) Genetic

Target/Source	Eng	Amh	Swa	Wol	Rus	Cze	Isl	Ger	Ara	Hin	Ben
Bho	0.32	0.37	0.16	0.22	0.39	0.37	0.32	0.38	0.31	**0.56**	0.45
Hau	**0.55**	0.35	0.44	**0.55**	0.41	0.28	0.49	0.46	0.41	0.38	0.22
Ibo	0.39	0.24	0.33	**0.44**	0.28	0.29	0.39	0.32	0.3	0.17	0.18
Kin	0.23	0.21	**0.34**	0.23	0.21	0.29	0.28	0.19	0.26	0.13	0.16
Lug	0.22	0.22	**0.52**	0.3	0.22	0.33	0.32	0.23	0.31	0.13	0.2
Luo	0.37	0.34	0.46	**0.51**	0.31	0.33	0.41	0.34	0.37	0.26	0.17
Ass	0.19	0.28	0.2	0.13	0.24	0.31	0.26	0.3	0.29	0.33	**0.38**
Qpm	0.52	0.38	0.32	0.46	**0.54**	0.45	0.41	0.46	0.39	0.41	0.31
Hsb	**0.43**	**0.43**	0.4	0.38	**0.43**	0.34	0.4	0.42	0.33	0.42	0.33
Orv	**0.43**	**0.43**	0.4	0.38	**0.43**	0.34	0.4	0.42	0.33	0.42	0.33
Chu	**0.43**	**0.43**	0.4	0.38	**0.43**	0.34	0.4	0.42	0.33	0.42	0.33
Fao	0.39	0.3	0.26	0.38	0.36	**0.46**	0.43	0.41	0.26	0.25	0.31
Got	0.33	0.29	0.33	0.38	0.33	0.39	**0.42**	0.4	0.3	0.29	0.3
Gsw	0.37	0.34	0.24	0.35	0.37	**0.46**	0.33	0.44	0.27	0.35	0.37

(b) Syntactic

Figure 5: Various similarity metrics between source and target languages (higher the more similar). This is used to validate the assignment of the target languages to corresponding families as well as for depicting correlation with the LA attention scores learnt by the Lang2Vec component.

D.1 Target language assignment

We justify assigning the unseen target language to a family using nearest neighour based on phylogenetic similarity (shown in fig. 5a). E.g. Hausa is genetically most similar to Amheric, so it's been assigned to the African family. On the other hand, Luo has equal genetic similarity with all source languages, so we refer to the syntactic similarity (fig. 5b), for tie-break, in which it's most similar to Wolof (also English, in this case, which is common in every family), and hence it's been assigned to the African family.

D.2 Consistency with the learnt attention scores

We also make use of these similarity metrics to validate the attention scores being learnt in the Lang2Vec modules (explained in detail in section 4.3). E.g. for Bhojpuri, highest attention score goes to Hindi LA while for Assamese, it does for Bengali. Indeed, we observe that Bhojpuri is closest to Hindi while Assamese being closest to Bengali, based upon the average of both genetic and syntactic similarities (fig. 2b). Detailed correlation between the attention scores and similarity have been discussed in section 4.3

E Examples

Model	Labels									
Sentence	Daas	Buech	laufft	besser	als	jede	vo	sine	Krimi	.
Gold Labels	DET	NOUN	VERB	ADV	CONJ	PRON	ADP	DET	NOUN	PUNCT
ZGUL	DET	NOUN	VERB	ADV	CONJ	PRON	ADP	DET	NOUN	PUNCT
CPG	DET	PROPN	VERB	ADV	ADP	DET	ADP	DET	NOUN	PUNCT

Table 10: Example from the gsw language

Model	Labels							
Sentence	весь	днь	милует	и	в	заимъ	даеть	праведныи
Gold Labels	DET	NOUN	VERB	CONJ	ADP	NOUN	VERB	ADJ
ZGUL	DET	PUNCT	VERB	PART	ADP	NOUN	VERB	ADJ
CPG	DET	PUNCT	VERB	CONJ	ADP	NOUN	VERB	ADJ

Table 11: Example from the orv language

Model	Labels											
Sentence	Jami'an	tsaron	Lebanon	sun	...	Jakadancin	Amurka	da	ke	birnin	Beirut	.
Gold Lables	O	O	LOC	O	...	O	LOC	O	O	O	LOC	O
ZGUL	O	O	LOC	O	...	O	LOC	O	O	O	LOC	O
SFT-M	O	O	LOC	O	...	PER	PER	O	O	O	LOC	O

Table 12: Example from the hau langauge

Model	Labels												
Sentence	Doho	...	pachoka	David	Maraga	chiwo	...	jii	11	matiyo	e	...	.
Gold Labels	O	...	O	PER	PER	O	...	O	O	O	O	...	O
ZGUL	O	...	O	PER	PER	O	...	O	DATE	DATE	O	...	O
SFT-M	O	...	O	PER	PER	O	...	O	O	O	O	...	O

Table 13: Example from the luo language

F Fusion based adapter weights distribution graphs

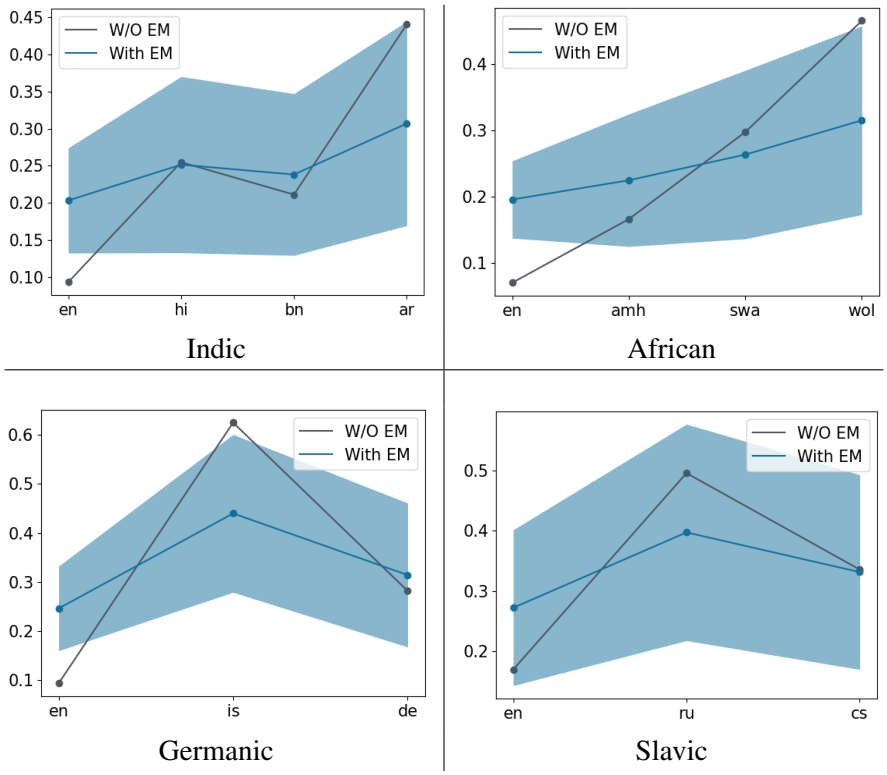


Figure 6: Fusion based adapter weights (on y axis) of each of the language families. Shown both weights with and w/o EM

G Training and Test datasets' details

Code	Language
Amh	Amharic
Ar	Arabic
As	Assamese
Be	Belorussian
Bg	Bulgarian
Bh	Bhojpuri
Bn	Bengali
Cs	Czech
Cu	Old Church Slavonic
Da	Danish
De	German
En	English
Fo	Faroese
Got	Gothic
Gsw	Swiss German
Hau	Hausa
Hi	Hindi

Code	Language
Hsb	Upper Sorbian
Ibo	Igbo
Is	Icelandic
Kin	Kinyarwanda
Lug	Ganda
Luo	Luo
Mr	Marathi
No	Norwegian
Orv	Old East Slavik
Qpm	Pomak
Ru	Russian
Swa	Swahili
Ta	Tamil
Uk	Ukrainian
Ur	Urdu
Wol	Wolof

Table 14: Languages and their codes

Family	Train set	Train set size
Indic	{En,Hi,Bn,Ur}	55026
Germanic	{En,Is,De}	222792
Slavic	{En,Ru,Cs}	179043
African	{En,Amh,Swa,Wol}	19788

Table 15: Training size

Test Language	Size
Fo	1208
Got	1031
Gsw	100
Qpm	635
Hsb	626
Orv	4204
Cu	1141

Test Language	Size
As	100
Bh	102
Hau	570
Ibo	642
Kin	611
Lug	419
Luo	189

Table 16: Testing size

Family	Train set	LA set	Test set
Indic	En,Hi,Bn,Ur	{En,Hi,Bn,Ar}	{As,Bh}
Germanic	{En,Is,De}	{En,Is,De}	{Fo,Got,Gsw}
Slavic	{En,Ru,Cs}	{En,Ru,Cs}	Qpm,Hsb,Orv,Cu
African	{En,Amh,Swa,Wol}	{En,Amh,Swa,Wol}	{Hau,Ibo,Kin,Lug,Luo,Pcm}

Table 17: Language families and their corresponding train, LA and test sets