UDapter: Language Adaptation for Truly Universal Dependency Parsing

Ahmet Üstün **Arianna Bisazza** **Gosse Bouma** **Gertjan van Noord**
University of Groningen
{a.ustun, a.bisazza, g.bouma, g.j.m.van.noord}@rug.nl

Abstract

Recent advances in multilingual dependency parsing have brought the idea of a truly universal parser closer to reality. However, cross-language interference and restrained model capacity remain major obstacles. To address this, we propose a novel multilingual task adaptation approach based on *contextual parameter generation* and *adapter modules*. This approach enables to learn adapters via language embeddings while sharing model parameters across languages. It also allows for an easy but effective integration of existing *linguistic typology features* into the parsing network. The resulting parser, UDapter, outperforms strong monolingual and multilingual baselines on the majority of *both* high-resource and low-resource (zero-shot) languages, showing the success of the proposed adaptation approach. Our in-depth analyses show that soft parameter sharing via typological features is key to this success.[1]

1 Introduction

Monolingual training of a dependency parser has been successful when relatively large treebanks are available (Kiperwasser and Goldberg, 2016; Dozat and Manning, 2017). However, for many languages, treebanks are either too small or unavailable. Therefore, multilingual models leveraging Universal Dependency annotations (Nivre et al., 2018) have drawn serious attention (Zhang and Barzilay, 2015; Ammar et al., 2016; de Lhoneux et al., 2018; Kondratyuk and Straka, 2019). Multilingual approaches learn generalizations across languages and share information between them, making it possible to parse a target language without supervision in that language. Moreover, multilingual models can be faster to train and easier to maintain than a large set of monolingual models.

However, scaling a multilingual model over a high number of languages can lead to sub-optimal results, especially if the training languages are typologically diverse. Often, multilingual neural models have been found to outperform their monolingual counterparts on low- and zero-resource languages due to positive transfer effects, but underperform for high-resource languages (Johnson et al., 2017; Arivazhagan et al., 2019; Conneau et al., 2020), a problem also known as "the curse of multilinguality". Generally speaking, a multilingual model without language-specific supervision is likely to suffer from over-generalization and perform poorly on high-resource languages due to limited capacity compared to the monolingual baselines, as verified by our experiments on parsing.

In this paper, we strike a good balance between maximum sharing and language-specific capacity in multilingual dependency parsing. Inspired by recently introduced parameter sharing techniques (Platanios et al., 2018; Houlsby et al., 2019), we propose a new multilingual parser, UDapter, that learns to modify its language-specific parameters including the adapter modules, as a function of language embeddings. This allows the model to share parameters across languages, ensuring generalization and transfer ability, but also enables language-specific parameterization in a single multilingual model. Furthermore, we propose not to learn language embeddings from scratch, but to leverage a mix of linguistically curated and predicted typological features as obtained from the URIEL language typology database (Littell et al., 2017) which supports 3718 languages including all languages represented in UD. While the importance of typological features for cross-lingual parsing is known for both non-neural (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015) and neural approaches (Ammar et al., 2016; Scholivet et al., 2019), we are the first to use them

[1] Our code for UDapter is publicly available at https://github.com/ahmetustun/udapter

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2302–2315,
November 16–20, 2020.

effectively as direct input to a neural parser, without manual selection, over a large number of languages in the context of zero-shot parsing where gold POS labels are not given at test time. In our model, typological features are crucial, leading to a substantial LAS increase on zero-shot languages and no loss on high-resource languages when compared to the language embeddings learned from scratch.

We train and test our model on the 13 syntactically diverse high-resource languages that were used by Kulmizev et al. (2019), and also evaluate it on 30 genuinely low-resource languages. Results show that UDapter significantly outperforms state-of-the-art monolingual (Straka, 2018) and multilingual (Kondratyuk and Straka, 2019) parsers on most high-resource languages and achieves overall promising improvements on zero-shot languages.

Contributions We conduct several experiments on a large set of languages and perform thorough analyses of our model. Accordingly, we make the following contributions: 1) We apply the idea of adapter tuning (Rebuffi et al., 2018; Houlsby et al., 2019) to the task of universal dependency parsing. 2) We combine adapters with the idea of contextual parameter generation (Platanios et al., 2018), leading to a novel language adaptation approach with state-of-the art UD parsing results. 3) We provide a simple but effective method for conditioning the language adaptation on existing typological language features, which we show is crucial for zero-shot performance.

2 Previous Work

This section presents the background of our approach.

Multilingual Neural Networks Early models in multilingual neural machine translation (NMT) designed dedicated architectures (Dong et al., 2015; Firat et al., 2016) whilst subsequent models, from Johnson et al. (2017) onward, added a simple language identifier to the models with the same architecture as their monolingual counterparts. More recently, multilingual NMT models have focused on maximizing transfer accuracy for low-resource language pairs, while preserving high-resource language accuracy (Platanios et al., 2018; Neubig and Hu, 2018; Aharoni et al., 2019; Arivazhagan et al., 2019), known as the (positive) transfer - (negative) interference trade-off. Another line of work builds massively multilingual pre-trained language models to produce contextual representation to be used in downstream tasks (Devlin et al., 2019; Conneau et al., 2020). As the leading model, multilingual BERT (mBERT)[2] (Devlin et al., 2019) which is a deep self-attention network, was trained without language-specific signals on the 104 languages with the largest Wikipedias. It uses a shared vocabulary of 110K WordPieces (Wu et al., 2016), and has been shown to facilitate cross-lingual transfer in several applications (Pires et al., 2019; Wu and Dredze, 2019). Concurrently to our work, Pfeiffer et al. (2020) have proposed to combine language and task adapters, small bottleneck layers (Rebuffi et al., 2018; Houlsby et al., 2019), to address the capacity issue which limits multilingual pre-trained models for cross-lingual transfer.

Cross-Lingual Dependency Parsing The availability of consistent dependency treebanks in many languages (McDonald et al., 2013; Nivre et al., 2018) has provided an opportunity for the study of cross-lingual parsing. Early studies trained a delexicalized parser (Zeman and Resnik, 2008; McDonald et al., 2013) on one or more source languages by using either gold or predicted POS labels (Tiedemann, 2015) and applied it to target languages. Building on this, later work used additional features such as typological language properties (Naseem et al., 2012), syntactic embeddings (Duong et al., 2015), and cross-lingual word clusters (Täckström et al., 2012). Among lexicalized approaches, Vilares et al. (2016) learns a bilingual parser on a corpora obtained by merging harmonized treebanks. Ammar et al. (2016) trains a multilingual parser using multilingual word embeddings, token-level language information, language typology features and fine-grained POS tags. More recently, based on mBERT (Devlin et al., 2019), zero-shot transfer in dependency parsing was investigated (Wu and Dredze, 2019; Tran and Bisazza, 2019). Finally Kondratyuk and Straka (2019) trained a multilingual parser on the concatenation of all available UD treebanks.

Language Embeddings and Typology Conditioning a multilingual model on the input language is studied in NMT (Ha et al., 2016; Johnson et al., 2017), syntactic parsing (Ammar et al., 2016) and language modeling (Östling and Tiedemann, 2017). The goal is to embed language information in real-

[2] https://github.com/google-research/bert/blob/master/multilingual.md

valued vectors in order to enrich internal representations with input language for multilingual models. In dependency parsing, several previous studies (Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Ammar et al., 2016; Scholivet et al., 2019) have suggested that typological features are useful for the selective sharing of transfer information. Results, however, are mixed and often limited to a handful of manually selected features (Fisch et al., 2019; Ponti et al., 2019). As the most similar work to ours, Ammar et al. (2016) uses typological features to learn language embeddings as part of training, by augmenting each input token and parsing action representation. Unfortunately though, this technique is found to underperform the simple use of randomly initialized language embeddings ('language IDs'). Authors also reported that language embeddings hurt the performance of the parser in zero-shot experiments (Ammar et al., 2016, *footnote 30*). Our work instead demonstrates that typological features can be very effective if used with the right adaptation strategy in both supervised and zero-shot settings. Finally, Lin et al. (2019) use typological features, along with properties of the training data, to choose optimal transfer languages for various tasks, including UD parsing, in a hard manner. By contrast, we focus on a *soft* parameter sharing approach to maximize generalizations within a single universal model.

3 Proposed Model

In this section, we present our truly universal dependency parser, UDapter. UDapter consists of a biaffine attention layer stacked on top of the pre-trained Transformer encoder (mBERT). This is similar to (Wu and Dredze, 2019; Kondratyuk and Straka, 2019), except that our mBERT layers are interleaved with special adapter layers inspired by Houlsby et al. (2019). While mBERT weights are frozen, biaffine attention and adapter layer weights are generated by a contextual parameter generator (Platanios et al., 2018) that takes a language embedding as input and is updated while training on the treebanks.

Note that the proposed adaptation approach is not restricted to dependency parsing and is in principle applicable to a range of multilingual NLP tasks. We will now describe the components of our model.

3.1 Biaffine Attention Parser

The top layer of UDapter is a graph-based biaffine attention parser proposed by Dozat and Manning (2017). In this model, an encoder generates an internal representation $\mathbf{r}_i$ for each word; the decoder takes $\mathbf{r}_i$ and passes it through separate feedforward layers (MLP), and finally uses deep biaffine attention to score arcs connecting a head and a tail:

$$\mathbf{h}_i^{(\text{head})} = \texttt{MLP}^{(\text{head})}(\mathbf{r}_i) \quad (1)$$

$$\mathbf{h}_i^{(\text{tail})} = \texttt{MLP}^{(\text{tail})}(\mathbf{r}_i) \quad (2)$$

$$\mathbf{s}^{(\text{arc})} = \texttt{Biaffine}(\mathbf{H}^{(\text{head})}, \mathbf{H}^{(\text{tail})}) \quad (3)$$

Similarly, label scores are calculated by using a biaffine classifier over two separate feedforward layers. Finally, the Chu-Liu/Edmonds algorithm (Chu, 1965; Edmonds, 1967) is used to find the highest scoring valid dependency tree.

3.2 Transformer Encoder with Adapters

To obtain contextualized word representations, UDapter uses mBERT. For a token i in sentence S, BERT builds an input representation $\mathbf{w}_i$ composed by summing a WordPiece embedding x_i (Wu et al., 2016) and a position embedding f_i. Each $\mathbf{w}_i \in S$ is then passed to a stacked self-attention layers (SA) to generate the final encoder representation $\mathbf{r}_i$:

$$\mathbf{w}_i = \mathrm{x}_i + \mathrm{f}_i \quad (4)$$

$$\mathbf{r}_i = \texttt{SA}\left(\mathbf{w}_i\,; \Theta^{(\text{ad})}\right) \quad (5)$$

where $\Theta^{(\text{ad})}$ denotes the adapter modules. During training, instead of fine-tuning the whole encoder network together with the task-specific top layer, we use adapter modules (Rebuffi et al., 2018; Stickland and Murray, 2019; Houlsby et al., 2019), or simply *adapters*, to capture both task-specific and language-specific information. Adapters are small modules added between layers of a pre-trained network. In adapter tuning, the weights of the original network are kept frozen, whilst the adapters are trained for a downstream task. Tuning with adapters was mainly suggested for parameter efficiency but they also act as an information module for the task or the language to be adapted (Pfeiffer et al., 2020). In this way, the original network serves as a memory for the language(s). In UDapter, following Houlsby et al. (2019), two bottleneck adapters with two feedforward projections and a GELU nonlinearity (Hendrycks and Gimpel, 2016) are inserted into each transformer layer as shown in

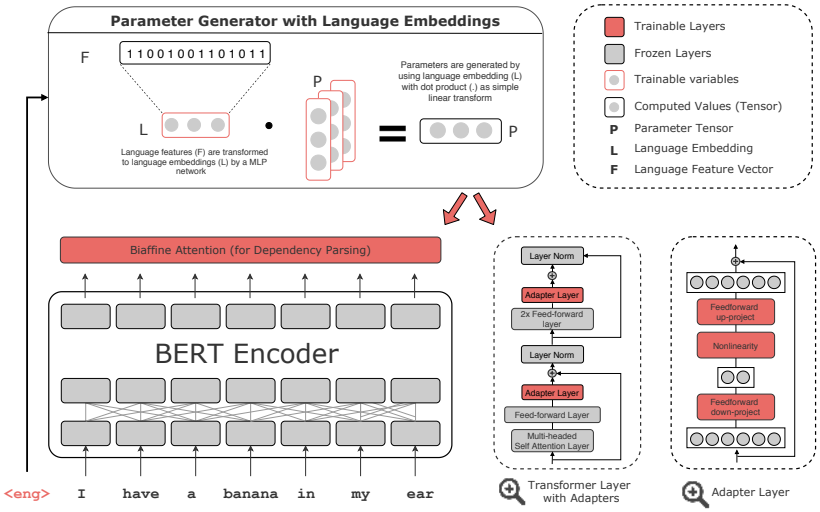


Figure 1: UDapter architecture with contextual parameter generator (CPG) and adapter layers. CPG takes languages embeddings projected from typological features as input and generates parameters of adapter layers and biaffine attention.

Figure 1. We apply adapter tuning for two reasons: 1) Each adapter module consists of only few parameters and allows to use contextual parameter generation (CPG; see § 3.3) with a reasonable number of trainable parameters.[3] 2) Adapters enable task-specific as well as language-specific adaptation via CPG since it keeps backbone multilingual representations as memory for all languages in pre-training, which is important for multilingual transfer.

3.3 Contextual Parameter Generator

To control the amount of sharing across languages, we generate trainable parameters of the model using a contextual parameter generator (CPG) function inspired by Platanios et al. (2018). CPG enables UDapter to retain high multilingual quality without losing performance on a single language, during multi-language training. We define CPG as a function of language embeddings. Since we only train adapters and the biaffine attention (i.e. adapter tuning), the parameter generator is formalized as $\{\theta^{(ad)}, \theta^{(bf)}\} \triangleq g^{(m)}(\mathbf{l}_e)$ where $g^{(m)}$ denotes the parameter generator with language embedding $\mathbf{l}_e$, and $\theta^{(ad)}$ and $\theta^{(bf)}$ denote the parameters of adapters and biaffine attention respectively. We implement CPG as a simple linear transform of a language embedding, similar to Platanios et al. (2018), so that weights of adapters in the encoder and biaffine attention are generated by the dot product of language embeddings:

$$g^{(m)}(\mathbf{l}_e) = (\mathbf{W}^{(\text{ad})}, \mathbf{W}^{(\text{bf})}) \cdot \mathbf{l}_e \quad (6)$$

where $\mathbf{l}_e \in \mathbb{R}^{\text{M}}$, $\mathbf{W}^{(\text{ad})} \in \mathbb{R}^{\mathbf{P}^{(\text{ad})} \times \text{M}}$, $\mathbf{W}^{(\text{bf})} \in \mathbb{R}^{\mathbf{P}^{(\text{bf})} \times \text{M}}$, M is the language embedding size, $\mathbf{P}^{(\text{ad})}$ and $\mathbf{P}^{(\text{bf})}$ are the number of parameters for adapters and biaffine attention respectively.[4] An important advantage of CPG is the easy integration of existing task or language features.

3.4 Typology-Based Language Embeddings

Soft sharing via CPG enables our model to modify its parsing decisions depending on a language embedding. While this allows UDapter to perform well on the languages in training, even if they are typologically diverse, information sharing is still a problem for languages not seen during training (zero-shot learning) as a language embedding is not available. Inspired by Naseem et al. (2012) and Ammar et al. (2016), we address this problem by defining language embeddings as a function of a large set of language typological features, including syntactic and phonological features. We use a multi-layer perceptron $\texttt{MLP}^{(\text{lang})}$ with two feed-forward layers and a $\texttt{ReLU}$ nonlinear activation to compute a language embedding $\mathbf{l}_e$:

$$\mathbf{l}_e = \texttt{MLP}^{(\text{lang})}(\mathbf{l}_t) \quad (7)$$

where $\mathbf{l}_t$ is a typological feature vector for a language consisting of **all** 103 syntactic, 28 phonological and 158 phonetic inventory features from the URIEL language typology database (Littell et al., 2017). URIEL is a collection of binary features

[3] Due to CPG, the number of adapter parameters is multiplied by language embedding size, resulting in a larger model compared to the baseline (more details in Appendix A.1).

[4] Platanios et al. (2018) also suggest to apply *parameter grouping*. We have not tried that yet, but one may learn separate low-rank projections of language embeddings for the adapter parameters group and the biaffine parameters group.

	ar	en	eu	fi	he	hi	it	ja	ko	ru	sv	tr	zh	HR-AVG	LR-AVG
Previous work:															
uuparser-bert [1]	81.8	87.6	79.8	83.9	85.9	90.8	91.7	92.1	84.2	91.0	86.9	64.9	83.4	84.9	-
udpipe [2]	82.9	87.0	82.9	87.5	86.9	91.8	91.5	**93.7**	84.2	92.3	86.6	67.6	80.5	85.8	-
udify [3]	82.9	88.5	81.0	82.1	88.1	91.5	**93.7**	92.1	74.3	**93.1**	89.1	67.4	**83.8**	85.2	34.1
Monolingually trained (one model per language):															
mono-udify	83.5	89.4	81.3	87.3	87.9	91.1	93.1	92.5	84.2	91.9	88.0	66.0	82.4	86.0	-
Multilingually trained (one model for all languages):															
multi-udify	80.1	88.5	76.4	85.1	84.4	89.3	92.0	90.0	78.0	89.0	86.2	62.9	77.8	83.0	35.3
adapter-only	82.8	88.3	80.2	86.9	86.2	90.6	93.1	91.6	81.3	90.8	88.4	66.0	79.4	85.0	32.9
udapter	**84.4**	**89.7**	**83.3**	**89.0**	**88.8**	**92.0**	93.5	92.8	**85.9**	92.2	**90.3**	**69.6**	83.2	**87.3**	**36.5**

Table 1: Labelled attachment scores (LAS) on high-resource languages for baselines and UDapter. Last two columns show average LAS of 13 high-resource (HR-AVG) and 30 low-resource (LR-AVG) languages respectively. Previous work results are reported from (Kulmizev et al., 2019) [1] and (Kondratyuk and Straka, 2019) [2,3].

	be	br*	bxr*	cy	fo*	gsw*	hsb*	kk	koi*	krl*	mdf*	mr	olo*	pcm*	sa*	tl	yo*	yue*	AVG
multi-udify	**80.1**	**60.5**	26.1	53.6	68.6	43.6	53.2	**61.9**	20.8	**49.2**	24.8	**46.4**	42.1	36.1	19.4	62.7	41.2	30.5	45.2
udapter-proxy	69.9	-	-	-	64.1	23.7	44.4	45.1	-	45.6	-	29.6	41.1	-	15.1	-	-	24.5	-
udapter	79.3	58.5	**28.9**	**54.4**	**69.2**	**45.5**	**54.2**	60.7	**23.1**	48.4	**26.6**	44.4	**43.3**	**36.7**	**22.2**	**69.5**	**42.7**	**32.8**	**46.2**

Table 2: Labelled attachment scores (LAS) on a subset of 30 low-resource languages. Languages with '*' are not included in mBERT training corpus. (Results for all low-resource languages, together with the chosen proxy, are given in Appendix A.2.)

extracted from multiple typological and phylogenetic databases such as WALS (World Atlas of Language Structures) (Dryer and Haspelmath, 2013), PHOIBLE (Moran and McCloy, 2019), Ethnologue (Lewis et al., 2015) and Glottolog (Hammarström et al., 2020). As many feature values are not available for each language, we use the values predicted by Littell et al. (2017) using a k-nearest neighbors approach based on average of genetic, geographical and feature distances between languages.

4 Experiments

Data and Training Details For our training languages, we follow Kulmizev et al. (2019), who selected from UD 2.3 (Nivre et al., 2018) 13 treebanks "from different language families, with different morphological complexity, scripts, character set sizes, training sizes, domains, and with good annotation quality" (see codes in Table 1).[5] During training, a language identifier is added to each sentence, and gold word segmentation is provided. We test our models on the training languages (*high-resource* set), and on 30 languages that have no or very little training data (*low-resource* set) in a zero-shot setup, i.e, without any training data.[6] The detailed treebank list is provided in Appendix A.3. For evaluation, the official CoNLL 2018 Shared Task script[7] is used to obtain LAS scores on the test set of each treebank.

For the encoder, we use *BERT-multilingual-cased* together with its WordPiece tokenizer. Since dependency annotations are between words, we pass the BERT output corresponding to the first wordpiece per word to the biaffine parser. We apply the same hyper-parameter settings as Kondratyuk and Straka (2019). Additionally, we use 256 and 32 for adapter size and language embedding size respectively. In our approach, pre-trained BERT weights are frozen, and only adapters and biaffine attention are trained, thus we use the same learning rate for the whole network by applying an inverse square root learning rate decay with linear warm-up (Howard and Ruder, 2018). Appendix A.1 gives the hyper-parameter details.

Baselines We compare UDapter to the current state of the art in UD parsing: [1] UUparser+BERT (Kulmizev et al., 2019), a graph-based BLSTM

[5]To reduce training time we cap the very large Russian Syntagrus treebank (48K sentences) to a random 15K sample.

[6]For this reason, the terms 'zero-shot' and 'low-resource' are used interchangeably in this paper.

[7]https://universaldependencies.org/conll18/evaluation.html

parser (de Lhoneux et al., 2017; Smith et al., 2018) using mBERT embeddings as additional features. [2] UDpipe (Straka, 2018), a monolingually trained multi-task parser that uses pretrained word embeddings and character representations. [3] UDify (Kondratyuk and Straka, 2019), the mBERT-based multi-task UD parser on which our UDapter is based, but originally trained on *all* language treebanks from UD. UDPipe scores are taken from Kondratyuk and Straka (2019).

To enable a direct comparison, we also re-train UDify on our set of 13 high-resource languages both monolingually (one treebank at a time; *mono-udify*) and multilingually (on the concatenation of languages; *multi-udify*). Finally, we evaluate two variants of our model: 1) Adapter-only has only task-specific adapter modules and no language-specific adaptation, i.e. no contextual parameter generator; and 2) UDapter-proxy is trained without typology features: a separate language embedding is learnt from scratch for each in-training language, and for low-resource languages we use one from the same language family, if available, as proxy representation.

Importantly, all baselines are either trained for a single language, or multilingually without any language-specific adaptation. By comparing UDapter to these parsers, we highlight its unique character that enables language specific parameterization by typological features within a multilingual framework for both supervised and zero-shot learning setup.

4.1 Results

Overall, UDapter outperforms the monolingual and multilingual baselines on both high-resource and zero-shot languages. Below, we elaborate on the detailed results.

High-resource Languages Labelled Attachement Scores (LAS) on the *high-resource* set are given in Table 1. UDapter consistently outperforms both our monolingual and multilingual baselines in all languages, and beats the previous work, setting a new state of the art, in 9 out of 13 languages. Statistical significance testing[8] applied between UDapter and *multi/mono-udify* confirms that UDapter's performance is significantly better than the baselines in 11 out of 13 languages (all except *en* and *it*).

[8]We used *paired bootstrap resampling* to check whether the difference between two models is significant ($p < 0.05$) by using Udapi (Popel et al., 2017).

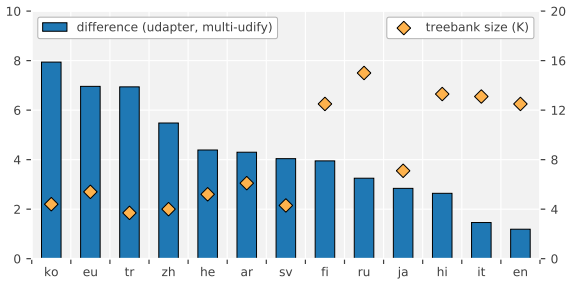


Figure 2: Difference in LAS between UDapter and multi-udify in the high-resource setting. Diamonds indicate the amount of sentences in the corresponding treebank.

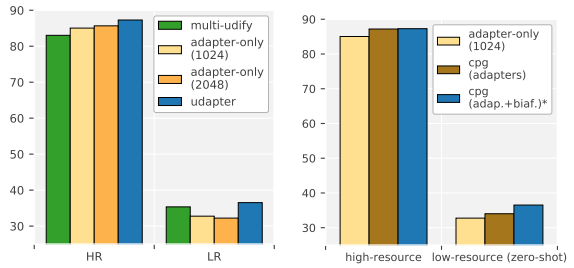
Among directly comparable baselines, *multi-udify* gives the worst performance in the typologically diverse high-resource setting. This multilingual model is clearly worse than its monolingually trained counterparts *mono-udify*: 83.0 *vs* 86.0. This result resounds with previous findings in multilingual NMT (Arivazhagan et al., 2019) and highlights the importance of language adaptation even when using high-quality sentence representations like those produced by mBERT.

To understand the relevance of adapters, we also evaluate a model which has almost the same architecture as multi-udify except for the adapter modules and the tuning choice (frozen mBERT weights). Interestingly, this adapter-only model considerably outperforms multi-udify (85.0 *vs* 83.0), indicating that adapter modules are also effective in multilingual scenarios.

Finally, UDapter achieves the overall best results, with consistent gains over both *multi-udify* and *adapter-only*, showing the importance of linguistically informed adaptation even for in-training languages.

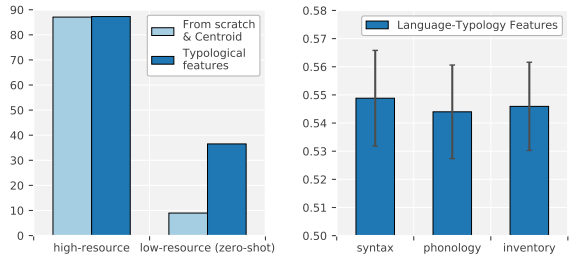
Low-Resource Languages Average LAS on the 30 low-resource languages are shown in column *lr-avg* of Table 1. Overall, UDapter slightly outperforms the multi-udify baseline (36.5 *vs* 36.3), which shows the benefits of our approach on both in-training and zero-shot languages. For a closer look, Table 2 provides individual results for the 18 representative languages in our low-resource set. Here we find a mixed picture: UDapter outperforms multi-udify on 13 out of 18 languages[9]. Achieving improvements in the zero-shot parsing

[9]LAS scores for all 30 languages are given in Appendix A.2. By significance testing, UDapter is *significantly* better than multi-udify on 16/30 low-resource languages, which is shown in Table 4



(a) (b)

Figure 3: Impact of different UDapter components on parsing performance (LAS): (a) adapters and adapter layer size, (b) application of contextual parameter generation to different portions of the network. In (b) the model named 'cpg (adap.+biaf.)' coincides with the full UDapter.



(a) (b)

Figure 4: (a) Impact of language typology features on parsing performance (LAS). (b) Average of normalized feature weights obtained from linear projection layer of the language embedding network.

setup is very difficult, thus we believe this result is an important step towards overcoming the problem of positive/negative transfer trade-off.

Indeed, UDapter-proxy results show that choosing a *proxy* language embedding from the same language family underperforms UDapter, apart from not being available for many languages. This indicates the importance of typological features in our approach (see § 5.2 for further analysis).

5 Analysis

In this section, we further analyse UDapter to understand its impact on different languages, and the importance of its various components.

5.1 Which languages improve most?

Figure 2 presents the LAS gain of UDapter over the multi-udify baseline for each high-resource language along with the respective treebank training size. To summarize, the gains are higher for languages with less training data. This suggests that in UDapter, useful knowledge is shared among in-training languages, which benefits low resource languages without hurting high resource ones.

For zero-shot languages, the difference between the two models is small compared to high-resource languages (+1.2 LAS). While it is harder to find a trend here, we notice that UDapter is typically beneficial for the languages *not* present in the mBERT training corpus: it outperforms multi-udify in 13 out of 22 (non-mBERT) languages. This suggests that typological feature-based adaptation leads to improved sentence representations when the pretrained encoder has not been exposed to a language.

5.2 How much gain from typology?

UDapter learns language embeddings from syntactic, phonological and phonetic inventory features. A natural alternative to this choice is to learn language embeddings from scratch. For a comparison, we train a model where, for each in-training language, a separate language embedding (of the same size: 32) is initialized randomly and learned end-to-end. For the zero-shot languages we use the average, or centroid, of all in-training language embeddings. As shown in Figure 4a, on the high-resource set, the models with and without typological features achieve very similar average LAS (87.3 and 87.1 respectively). On zero-shot languages, however, the use of centroid embedding performs very poorly: 9.0 *vs* 36.5 average LAS score over 30 languages. As already discussed in § 4.1 (Table 2), using a proxy language embedding belonging to the same family as the test language, when available, also clearly underperforms UDapter.

These results confirm our expectation that a model can learn reliable language embeddings for in-training languages, however typological signals are required to obtain a robust parsing quality on zero-shot languages.

5.3 How does UDapter represent languages?

We start by analyzing the projection weights assigned to different typological features by the first layer of the language embedding network (see eq. 7). Figure 4b shows the averages of normalized syntactic, phonological and phonetic inventory feature weights. Although dependency parsing is a syntactic task, the network does not only utilize syntactic features, as also observed by Lin et al. (2019), but exploits all available typological features to learn its representations.

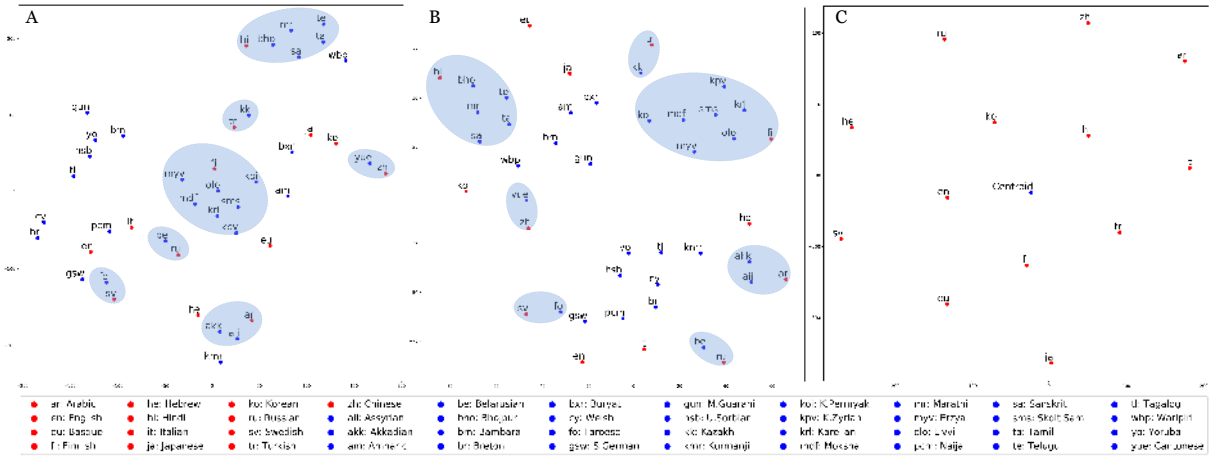


Figure 5: Vector spaces for (A) language-typology feature vectors taken from URIEL, (B) language embeddings learned from typological features by UDapter, and (C) language embeddings learned without typological features. High- and low-resource languages are indicated by red and blue dots respectively. Highlighted clusters in A and B denote sets of genetically related languages.

Next, we plot the language representations learned in UDapter by using t-SNE (van der Maaten and Hinton, 2008), which is similar to the analysis carried out by Ponti et al. (2019, figure 8) using the language vectors learned by Malaviya et al. (2017). Figure 5 illustrates 2D vector spaces generated for the typological feature vectors $\mathbf{l}_t$ (**A**) and the language embeddings $\mathbf{l}_e$ learned by UDapter with or without typological features (**B** and **C** respectively). The benefits of using typological features can be understood by comparing **A** and **B**: During training, UDapter learns to project URIEL features to language embeddings in a way that is optimal for in-training language parsing quality. This leads to a different placement of the high-resource languages (red points) in the space, where many linguistic similarities are preserved (e.g. Hebrew and Arabic; European languages except Basque) but others are overruled (Japanese drifting away from Korean). Looking at the low-resource languages (blue points) we find that typologically similar languages tend to have similar embeddings to the closest high-resource language in both **A** and **B**. In fact, most groupings of genetically related languages, such as the Indian languages (*hi*-cluster) or the Uralic ones (*fi*-cluster) are largely preserved across these two spaces.

Comparing **B** and **C** where language embeddings are learned from scratch, the absence of typological features leads to a seemingly random space with no linguistic similarities (e.g. Arabic far away from Hebrew, Korean closer to English than to Japanese, etc.) and, therefore, no principled way to represent additional languages.

Taken together with the parsing results of § 4.1, these plots suggest that UDapter embeddings strike a good balance between a linguistically motivated representation space and one solely optimized for in-training language accuracy.

5.4 Is CPG really essential?

In section 4.1 we observed that adapter tuning alone (that is, without CPG) improved the multilingual baseline in the high-resource languages, but worsened it considerably in the zero-shot setup. By contrast, the addition of CPG with typological features led to the best results over all languages. But could we have obtained similar results by simply increasing the adapter size? For instance, in multilingual MT, increasing overall model capacity of an already very large and deep architecture can be a powerful alternative to more sophisticated parameter sharing approaches (Arivazhagan et al., 2019). To answer this question we train another adapter-only model with doubled size (2048 instead of the 1024 used in the main experiments).

As seen in 3a, increase in model size brings a slight gain to the high-resource languages, but actually leads to a small loss in the zero-shot setup. This shows that adapters enlarge the per-language capacity for in-training languages, but at the same time they hurt generalization and zero-shot transfer. By contrast, UDapter including CPG which increases the model size by language embeddings (see Appendix A.1 for details), outperforms both adapter-only models, confirming once more the

importance of this component.

For our last analysis (Fig. 3b), we study soft parameter sharing via CPG on different portions of the network, namely: only on the adapter modules 'cpg (adapters)' *versus* on both adapters and biaffine attention 'cpg (adap.+biaf.)' corresponding to the full UDapter. Results show that most of the gain in the high-resource languages is obtained by only applying CPG on the multilingual encoder. On the other hand, for the low-resource languages, typological feature based parameter sharing is most important in the biaffine attention layer. We leave further investigation of this result to future work.

6 Conclusion

We have presented UDapter, a multilingual dependency parsing model that learns to adapt language-specific parameters on the basis of adapter modules (Rebuffi et al., 2018; Houlsby et al., 2019) and the contextual parameter generation (CPG) method (Platanios et al., 2018) which is in principle applicable to a range of multilingual NLP tasks. While adapters provide a more general task-level adaptation, CPG enables language-specific adaptation, defined as a function of language embeddings projected from linguistically curated typological features. In this way, the model retains high per-language performance in the training data and achieves better zero-shot transfer.

UDapter, trained on a concatenation of typologically diverse languages (Kulmizev et al., 2019), outperforms strong monolingual *and* multilingual baselines on the majority of *both* high-resource and low-resource (zero-shot) languages, which reflects its strong balance between per-language capacity and maximum sharing. Finally, the analyses we performed on the underlying characteristics of our model show that typological features are crucial for zero-shot languages.

Acknowledgements

Arianna Bisazza was partly funded by the Netherlands Organization for Scientific Research (NWO) under project number 639.021.646. We would like to thank the Center for Information Technology of the University of Groningen for providing access to the Peregrine HPC cluster and the anonymous reviewers for their helpful comments.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. 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 3874–3884.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. *CoRR*, abs/1907.05019.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

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.

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.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. A neural network model for low-resource Universal Dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 339–348.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875.

Adam Fisch, Jiang Guo, and Regina Barzilay. 2019. Working hard or hardly working: Challenges of integrating typology into neural dependency parsers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5714–5720.

Thanh-Le Ha, Jan Niehues, and Alex Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *ArXiv preprint*.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2020. *Glottolog 4.3*. Max Planck Institute for the Science of Human History, Jena.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *International Conference on Learning Representations*.

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.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795.

Artur Kulmizev, Miryam de Lhoneux, Johannes Gontrum, Elena Fano, and Joakim Nivre. 2019. Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2755–2768.

M Paul Lewis, Gary F Simons, and CD Fennig. 2015. Ethnologue: Languages of the world [eighteenth. *Dallas, Texas: SIL International*.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4992–4997.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017. From raw text to Universal Dependencies - look, no tags! In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 207–217.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing transfer languages for cross-lingual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3125–3135.

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.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Chaitanya Malaviya, Graham Neubig, and Patrick Littell. 2017. Learning language representations for typology prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2529–2535.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency annotation for multilingual parsing.

In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97.

Steven Moran and Daniel McCloy, editors. 2019. *PHOIBLE 2.0*. Max Planck Institute for the Science of Human History, Jena.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 629–637.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Katya Aplonova, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Victoria Basmov, John Bauer, Sandra Bellato, Kepa Bengoetxea, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Ọlájídé Ishola, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Kamil Kopacewicz, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Lucia Lam, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phôêng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, 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, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Keiko Sophie Mori, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lôêng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roșca, Olga Rudina, Jack Rueter, Shoval Sadde, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jing Xian Wang, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.3. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 644–649.

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*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001.

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.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2019. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *Computational Linguistics*, 45(3):559–601.

Martin Popel, Zdeněk Žabokrtský, and Martin Vojtek. 2017. Udapi: Universal API for Universal Dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 96–101.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127.

Manon Scholivet, Franck Dary, Alexis Nasr, Benoit Favre, and Carlos Ramisch. 2019. Typological features for multilingual delexicalised dependency parsing. 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 3919–3930.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal Dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 477–487.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with Universal Dependencies and predicted PoS labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349.

Ke Tran and Arianna Bisazza. 2019. Zero-shot dependency parsing with pre-trained multilingual sentence representations. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 281–288.

David Vilares, Carlos Gómez-Rodríguez, and Miguel A. Alonso. 2016. One model, two languages: training bilingual parsers with harmonized treebanks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 425–431.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844.

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

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1857–1867.

Hyper-Parameter	Value
Dependency tag dimension	256
Dependency arc dimension	768
Optimizer	Adam
$\beta_1,\ \beta_2$	0.9, 0.99
Weight decay	0.01
Label smoothing	0.03
Dropout	0.5
BERT dropout	0.2
Mask probability	0.2
Batch size	32
Epochs	80
Base learning rate	$1e^{-3}$
BERT learning rate	$5e^{-5}$
LR warm up ratio	$1/80$
Adapter size	256
Language embedding size	32

Table 3: Hyper-parameter setting

A Appendix

A.1 Experimental Details

Implementation UDapter's implementation is based on UDify (Kondratyuk and Straka, 2019). We use the same hyper-parameters setting optimized in UDify without applying a new hyper-parameter search. Together with the additional *adapter size* and *language embedding size* that are picked manually by parsing accuracy, hyper-parameters are given in Table 3. Note that, to give a fair chance to the adapter-only baseline (see §4), we used 1024 as adapter size unlike that of the final UDapter (256). For fair comparison, mono-udify and multi-udify are re-trained on the concatenation of 13 high-resource languages for only dependency parsing. Besides, we did not use a layer attention for both our model and the baselines.

Training Time and Model size Comparing to UDify, UDapter has a similar training time. An epoch over the full training set takes approximately 27 and 30 minutes in UDify and UDapter respectively on a Tesla V100 GPU. In terms of number of *trainable* parameters, UDify has 191M total number of parameters whereas UDapter uses 550M parameters in total, 302M for adapters (32x9.4M) and 248M for biaffine attention (32x7.8M), since the parameter generator network (CPG) multiplies the tensors with language embedding size (32). Note that for multilingual training, UDapter's parameter cost depends only on language embedding size regardless of number of languages, therefore it highly scalable with an increasing number of languages for larger experiments. Finally, monolingual UDify models are trained separately so the total number of parameters for 13 languages is 2.5B (13x191M).

	orig.udify	multi-udify	udapter	udap.-proxy
aii*	9.1	8.4	**14.3**	8.2 (ar)
akk*	4.4	4.5	**8.2**	9.1 (ar)
am*	2.6	2.8	**5.9**	1.1 (ar)
be	**81.8**	80.1	79.3	69.9 (ru)
bho*(†)	35.9	37.2	**37.3**	35.9 (hi)
bm*	7.9	**8.9**	8.1	3.1 (CTR)
br*	39.0	**60.5**	58.5	14.3 (CTR)
bxr*	26.7	26.1	**28.9**	9.1 (CTR)
cy	42.7	53.6	**54.4**	9.8 (CTR)
fo*	59.0	68.6	**69.2**	64.1 (sv)
gsw*	39.7	43.6	**45.5**	23.7 (en)
gun*(†)	6.0	**8.5**	8.4	2.1 (CTR)
hsb*	**62.7**	53.2	54.2	44.4 (ru)
kk	**63.6**	61.9	60.7	45.1 (tr)
kmr*(†)	**20.2**	11.2	12.1	4.7 (CTR)
koi*	22.6	20.8	**23.1**	6.5 (CTR)
kpv*(†)	**12.9**	12.4	12.5	4.7 (CTR)
krl*	41.7	**49.2**	48.4	45.6 (fi)
mdf*	19.4	24.7	**26.6**	8.7 (CTR)
mr	**67.0**	46.4	44.4	29.6 (hi)
myv*(†)	16.6	19.1	**19.2**	6.3 (CTR)
olo*	33.9	42.1	**43.3**	41.1 (fi)
pcm*(†)	31.5	36.1	**36.7**	5.6 (CTR)
sa*	19.4	19.4	**22.2**	15.1 (hi)
ta (†)	**71.4**	46.0	46.1	12.3 (CTR)
te (†)	**83.4**	71.2	71.1	23.1 (CTR)
tl	41.4	62.7	**69.5**	14.1 (CTR)
wbp*	6.7	9.6	**12.1**	4.8 (CTR)
yo	22.0	41.2	**42.7**	10.5 (CTR)
yue*	31.0	30.5	**32.8**	24.5 (zh)
avg	34.1	35.3	**36.5**	20.4

Table 4: LAS results of UDapter and UDify models (Kondratyuk and Straka, 2019) for all low-resource languages. '*' shows languages not present in mBERT training data. Additionally, (†) indicates languages where no significant difference between UDapter and *multi-udify* by significance testing. For udapter-proxy, chosen proxy language is given between brackets. CTR means centroid language embedding.

A.2 Zero-Shot Results

Table 4 shows LAS scores on all 30 low-resouce languages for UDapter, original UDify (Kondratyuk and Straka, 2019), and re-trained 'multi-udify'. Languages with '*' are not included in mBERT training data. Note that original UDify is trained on all available UD treebanks from 75 languages. For the zero-shot languages, we obtained original UDify scores by running the pre-trained model.

A.3 Language Details

Details of training and zero-shot languages such as language code, data size (number of sentences), and family are given in Table 5 and Table 6.

Language	Code	Treebank	Family	Word Order	Train	Test
Arabic	ar	PADT	Afro-Asiatic, Semitic	VSO	6.1k	680
Basque	eu	BDT	Basque	SOV	5.4k	1799
Chinese	zh	GSD	Sino-Tibetan	SVO	4.0k	500
English	en	EWT	IE, Germanic	SVO	12.5k	2077
Finnish	fi	TDT	Uralic, Finnic	SVO	12.2k	1555
Hebrew	he	HTB	Afro-Asiatic, Semitic	SVO	5.2k	491
Hindi	hi	HDTB	IE, Indic	SOV	13.3k	1684
Italian	it	ISDT	IE, Romance	SVO	13.1k	482
Japanese	ja	GSD	Japanese	SOV	7.1k	551
Korean	ko	GSD	Korean	SOV	4.4k	989
Russian	ru	SynTagRus	IE, Slavic	SVO	15k*	6491
Swedish	sv	Talbanken	IE, Germanic	SVO	4.3k	1219
Turkish	tr	IMST	Turkic, Southwestern	SOV	3.7k	975

Table 5: Training languages that are from UD 2.3 (Nivre et al., 2018) with the details including treebank name, family, word order and data size of training and test sets.

Language	Code	Treebank(s)	Family	Test
Akkadian	akk	PISANDUB	Afro-Asiatic, Semitic	1074
Amharic	am	ATT	Afro-Asiatic, Semitic	101
Assyrian	aii	AS	Afro-Asiatic, Semitic	57
Bambara	bm	CRB	Mande	1026
Belarusian	be	HSE	IE, Slavic	253
Bhojpuri	bho	BHTB	IE, Indic	254
Breton	br	KEB	IE, Celtic	888
Buryat	bxr	BDT	Mongolic	908
Cantonese	yue	HK	Sino-Tibetan	1004
Erzya	myv	JR	Uralic, Mordvin	1550
Faroese	fo	OFT	IE, Germanic	1207
Karelian	krl	KKPP	Uralic, Finnic	228
Kazakh	kk	KTB	Turkic, Northwestern	1047
Komi Permyak	koi	UH	Uralic, Permic	49
Komi Zyrian	kpv	LATTICE, IKDP	Uralic, Permic	210
Kurmanji	kmr	MG	IE, Iranian	734
Livvi	olo	KKPP	Uralic, Finnic	106
Marathi	mr	UFAL	IE, Indic	47
Mbya Guarani	gun	THOMAS, DOOLEY	Tupian	98
Moksha	mdf	JR	Uralic, Mordvin	21
Naija	pcm	NSC	Creole	948
Sanskrit	sa	UFAL	IE, Indic	230
Swiss G.	gsw	UZH	IE, Germanic	100
Tagalog	tl	TRG	Austronesian, Central Philippine	55
Tamil	ta	TTB	Dravidian, Southern	120
Telugu	te	MTG	Dravidian, South Central	146
Upper Sorbian	hsb	UFAL	IE, Slavic	623
Warlpiri	wbp	UFAL	Pama-Nyungan	54
Welsh	cy	CCG	IE, Celtic	956
Yoruba	yo	YTB	Niger-Congo, Defoid	100

Table 6: Zero-shot languages are selected from UD 2.5 to increase the number of languages in the experiments. Language details include treebank name, family and test size for zero-shot experiments.