UDapter: Typology-based Language Adapters for Multilingual Dependency Parsing and Sequence Labeling

Ahmet Üstün
University of Groningen
Center for Language and Cognition
a.ustun@rug.nl

Arianna Bisazza
University of Groningen
Center for Language and Cognition
a.bisazza@rug.nl

Gosse Bouma
University of Groningen
Center for Language and Cognition
g.bouma@rug.nl

Gertjan van Noord
University of Groningen
Center for Language and Cognition
g.j.m.van.noord@rug.nl

Recent advances in multilingual language modeling have brought the idea of a truly universal parser closer to reality. However, such models are still not immune to the "curse of multilinguality": Cross-language interference and restrained model capacity remain major obstacles. To address this, we propose a novel language adaptation approach by introducing contextual language adapters to a multilingual parser. Contextual language adapters make it possible to learn adapters via language embeddings while sharing model parameters across languages based on contextual parameter generation. Moreover, our method allows for an easy but effective integration of existing linguistic typology features into the parsing model. Because not all typological features are available for every language, we further combine typological feature prediction with parsing in a multi-task model that achieves very competitive parsing performance without the need for an external prediction system for missing features.

Action Editor: Miguel Ballesteros. Submission received: 30 November 2021; revised version received: 3 November 2022; accepted for publication: 25 February 2022.

https://doi.org/10.1162/coli_a_00443

The resulting parser, UDapter, can be used for dependency parsing as well as sequence labeling tasks such as POS tagging, morphological tagging, and NER. In dependency parsing, it 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. In sequence labeling tasks, our parser surpasses the baseline on high resource languages, and performs very competitively in a zero-shot setting. Our in-depth analyses show that adapter generation via typological features of languages is key to this success.[1]

1. Introduction

When monolingual resources are limited, training a single model on datasets of multiple languages results in state-of-the-art performance for many tasks, such as dependency parsing, part-of-speech tagging, and named entity recognition (Ammar et al. 2016; de Lhoneux et al. 2018; Rahimi, Li, and Cohn 2019). Besides being faster to train and easier to maintain than a large set of monolingual models, multilingual models have the potential to learn better generalizations by sharing parameters across languages, thereby enabling strong cross-lingual transfer and zero-shot learning.

Following recent advances in massively multilingual pre-training such as multilingual BERT (Devlin et al. 2019) and XLM-RoBERTa (Conneau et al. 2020), recent studies use a multilingual pre-trained Transformer as the backbone model and fine-tune it on the concatenation of multiple datasets from different languages (Kondratyuk and Straka 2019; Tran and Bisazza 2019; van der Goot et al. 2021). These approaches achieve further improvements for languages with little training data. However, scaling a multilingual model to a high number of languages can face the "transfer–interference trade-off" (Johnson et al. 2017; Arivazhagan et al. 2019; Conneau et al. 2020), a problem also known as "the curse of multilinguality." Due to this trade-off, multilingual models outperform monolingual baselines on low/zero-resource languages (**positive transfer**), but perform worse on high-resource languages due to a lack of language-specific capacity (**negative interference**). Moreover, multilingual transfer often gives mixed results when the model is trained on a diverse set of source languages in terms of script, morphology, and syntax (Tran and Bisazza 2019; Kondratyuk and Straka 2019).

In this work, we address this problem by striking a better balance between maximum sharing and language-specific capacity in multilingual models of dependency parsing and various sequence labeling tasks. Inspired by recent parameter sharing techniques (Platanios et al. 2018) and adapter-based tuning approaches (Houlsby et al. 2019; Stickland and Murray 2019), we propose a new multilingual parser, UDapter, that learns to modify its language-specific parameters, that is, the adapter modules, as a function of learned language representations. In other words, UDapter learns *contextual language adapters* by using the respective language embeddings in the context of the target task. Thus, our adaptation approach allows the model to share parameters across languages, ensuring generalization and transfer ability, but also enables language-specific parameterization in a single multilingual model.

Crucially, we propose not to learn language embeddings from scratch, but to leverage a mix of linguistically curated typological features as obtained from the URIEL language typology database (Littell et al. 2017), which currently supports more than

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

3,000 languages. This allows UDapter to work for languages that do not have any task-specific training data—i.e. zero-shot setting—as the model can project any vector of typological features to a language embedding for the adapter generation after fine-tuning.[2] Using typological information in this manner leads to substantial performance gains on languages without training data and no loss on high-resource languages when compared to language embeddings learned from scratch. Furthermore, because some typological features are missing from URIEL for many languages, we hypothesize that learning to predict such missing features jointly with the target task may benefit downstream performance (Ponti et al. 2019). To achieve that, we propose to mask a random set of typological features during training and predict those features together with the target word labels in a multi-task manner. During inference, the model is then capable of projecting incomplete typological feature vectors to language embeddings, thereby eliminating the need of an external typology prediction system.

While the importance of typological features for cross-lingual parsing is known for both non-neural (Naseem, Barzilay, and Globerson 2012; Täckström, McDonald, and Nivre 2013; Zhang and Barzilay 2015) and neural approaches (Ammar et al. 2016; Scholivet et al. 2019), we are the first to use them *effectively* for genuinely low-resource languages. We use these features as direct input to a neural parser, without manual selection, over a large number of languages, including the zero-shot setup, without assuming gold part-of-speech (POS) labels are given at test time. Modeling prediction of typological features jointly with the target task also fills a significant gap in previous work, as modeling typological feature predictions directly from textual data enables one to learn more composite representations of linguistic typology instead of discrete features (Ponti et al. 2019).

We train UDapter on a set of 13 syntactically diverse high-resource languages (Kulmizev et al. 2019) for dependency parsing, as our core task, and three sequence labeling tasks: POS tagging, morphological tagging, and named entity recognition (NER). Extending UDapter to different tasks allows us to investigate *how* and *when* our adaptation approach is effective. We then evaluate the resulting models on these 13 languages as well as 30 genuinely low-resource languages to assess the generalization and zero-shot learning abilities of UDapter in different tasks. Finally, to get a better understanding of what makes typology-based language adaptation successful, we present a large set of analyses including ablation tests for typological features and contextual language adapters, visualization of learned language representations, and correlation of language-specific gains with training data size and number of known typological features.

This article extends our conference paper on multilingual dependency parsing (Üstün et al. 2020) in two significant ways:

1. We generalize UDapter to sequence labeling tasks, namely: POS tagging, morphological tagging, and NER.
2. We model learning of missing typological features prediction jointly with the target task, enabling the use of UDapter without the need for an external typology prediction system at inference time.

2 Fine-tuning and training are used interchangeably in this paper to denote the task-specific fine-tuning procedure after the language model pre-training.

The article is structured as follows: We describe the background for our work in Section 2, and present the proposed model with corresponding sub-components in Section 3. Multi-task prediction of typological features is introduced in Section 4. After introducing the experimental setup in Section 5, we present the core results in Section 6. Finally we provide a comprehensive analysis of our model and results in Section 7.

2. Previous Work

This section presents the background of our approach.

2.1 Multilingual Neural Networks

Early approaches to multilingual neural machine translation (NMT) designed dedicated architectures (Dong et al. 2015; Firat, Cho, and Bengio 2016) whereas 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, Johnson, and Firat 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 prior model, multilingual BERT (mBERT)[3] (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, Schlinger, and Garrette 2019; Wu and Dredze 2019). Similar to our work, Pfeiffer et al. (2020) have proposed to combine language and task adapters, small bottleneck layers (Rebuffi, Bilen, and Vedaldi 2018; Houlsby et al. 2019), to address the capacity issue that limits multilingual pre-trained models for cross-lingual transfer.

2.2 Cross-Lingual Dependency Parsing and Sequence Labeling

The availability of uniformly annotated dependency treebanks in many languages (McDonald et al. 2013; Nivre et al. 2016, 2020; de Marneffe et al. 2021) 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, Barzilay, and Globerson 2012), syntactic embeddings (Duong et al. 2015), and cross-lingual word clusters (Täckström, McDonald, and Uszkoreit 2012). Among lexicalized approaches, Vilares, Gómez-Rodríguez, and Alonso (2016) learn a bilingual parser on training data obtained by merging harmonized treebanks. Ammar et al. (2016) train 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).

3 https://github.com/google-research/bert/blob/master/multilingual.md.

Finally, Kondratyuk and Straka (2019) trained a multilingual parser on the concatenation of all available UD treebanks.

For sequence labeling tasks, early work on multilingual learning has been applied to POS and morphological tagging (Gillick et al. 2016; Tsai et al. 2019), and NER (Mulcaire, Kasai, and Smith 2019; Rahimi, Li, and Cohn 2019). Gillick et al. (2016) showed that a compact multilingual model operating on bytes could reach similar or better performance than their monolingual counterparts in POS and NER tasks. More recently, Mulcaire, Kasai, and Smith (2019) and Wu and Dredze (2019) showed that multilingual language model pre-training based on ELMo (Peters et al. 2018) and BERT (Devlin et al. 2019) improves the performance on POS, NER, and UD tasks, including zero-shot settings.

2.3 Language Embeddings and Typology

Conditioning a multilingual model on the input language has been studied in NMT (Ha, Niehues, and Waibel 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-valued vectors in order to enrich internal representations with input language for multilingual models. In dependency parsing, several previous studies (Naseem, Barzilay, and Globerson 2012; Täckström, McDonald, and Nivre 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, Guo, and Barzilay 2019; Ponti et al. 2019). As the most similar work to ours, Ammar et al. (2016) use 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. Multilingual Adaptation with Contextual Language Adapters and Typology

We address the main limitation of existing multilingual models, the positive transfer–negative interference trade-off, by proposing a novel multilingual adaptation method for pre-trained models. Different from the previous cross-lingual methods that fine-tune all parameters of a multilingual Transformer (Wu and Dredze 2019; Kondratyuk and Straka 2019), our method creates extra capacity with adapter layers (Rebuffi, Bilen, and Vedaldi 2018; Houlsby et al. 2019) for each language separately, without restricting information sharing. Our goal is to adapt the pre-trained model in a multilingual setup enabling maximum sharing across languages as well as language-specific parameterization. The key model component to achieve this is the **adapter generator network**, which uses a contextual parameter generator (Platanios et al. 2018), that is, a hypernetwork, to generate the parameter weights of adapters per language, given a language embedding as input.

We then add another component, the **typology feature network**, to our model so it can learn to project vectors of typological features to language embeddings. Using typological features enables the model to run in the zero-shot setup for languages without training data, as their language embeddings can be computed at inference time directly from the typological features. We give an overview of our model in Figure 2 and describe different components in the following subsections.

3.1 Multilingual Encoder with Task-Specific Layers

As the base multilingual pre-trained encoder, we use multilingual BERT[4] (Devlin et al. 2019), which has been shown to facilitate cross-lingual transfer (Pires, Schlinger, and Garrette 2019; Wu and Dredze 2019). mBERT is a Transformer model (Vaswani et al. 2017) with 12 self-attention layers, each consisting of 12 attention heads. It was pre-trained on the concatenation of 104 language corpora with the original masked language modeling method, and no explicit cross-lingual signal, using a shared vocabulary of 110K wordpieces (Wu et al. 2016).

In order to use our model for dependency parsing and sequence labeling tasks, we use task-specific layers on top of mBERT. For each task separately, we insert a task-specific top layer, and fine-tune the model on the concatenation of the corpora of multiple high-resource languages by using our novel adapter generation method (see Section 3.2). After the training, for low-resource languages, we do not train the model further, but we only use typological features of those languages and evaluate our model in zero-shot setup. Figure 1 shows our experimental setup.

To benefit language-aware parameter sharing at a higher degree, we apply contextual parameter generation (CPG; Section 3.3) to generate parameter weights of task-specific layers together with language adapters. Using CPG for task-specific layers increases model performance for zero-shot languages in particular (see Section 7.6). The following paragraphs explain the details of the task-specific layer for the corresponding target tasks.

Dependency Parsing. We use the graph-based deep biaffine attention layer proposed by Dozat and Manning (2017). In this layer, intermediate embeddings for arc-head $\mathbf{h}_i^{head}$ and arc-dep $\mathbf{h}_j^{dep}$ are produced by feedforward layers with Exponential Linear Unit (ELU) non-linear activation for each word pair i, j in a sentence. Then, a biaffine attention combines $\mathbf{h}_i^{head}$ and $\mathbf{h}_j^{dep}$ to score all possible dependency arcs:

$$\mathbf{h}_{head,i} = \text{ELU}(\mathbf{W}_{head}^T \mathbf{r}_i) \tag{1}$$

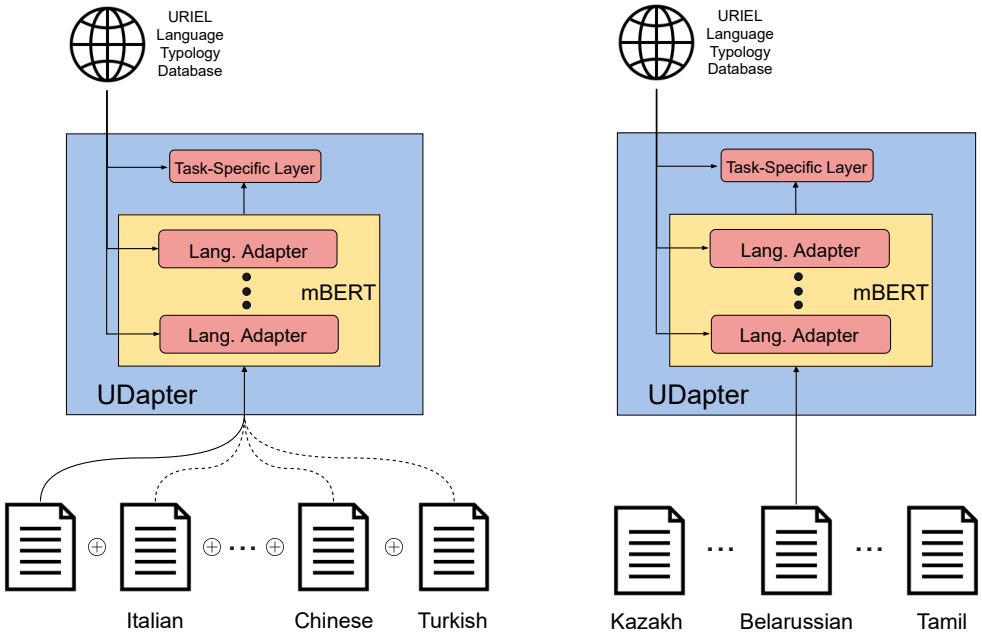
$$\mathbf{h}_{dep,j} = \text{ELU}(\mathbf{W}_{dep}^T \mathbf{r}_j) \tag{2}$$

$$\mathbf{S}_{arc} = \text{H}_{head}\ \mathbf{W}_{arc}\ \text{H}_{dep}^T \ \text{(biaffine)} \tag{3}$$

$$\mathbf{s}_{i,j}^{(arc)} = \text{softmax}(\mathbf{S}_{arc}) \tag{4}$$

where $\mathbf{r}_i$ is the output embedding of word i generated by mBERT with contextual language adapters; $\mathbf{W}_{head}, \mathbf{W}_{dep}, \mathbf{W}_{arc}$ are the weights of feedforward layers and biaffine,

4 https://github.com/google-research/bert/blob/master/multilingual.md.



(a) Multilingual Fine-tuning on HR (b) Zero-shot Evaluation on LR

Figure 1
Overview of UDapter experimental setup. We start with mBERT, and fine-tune it *multilingually* on 13 high-resource (HR) languages (1a). For low-resource (LR) languages, we evaluate UDapter in a zero-shot setup (1b). For zero-shot evaluation, the only source of information is the URIEL database (Littell et al. 2017) that provides typological features of the languages.

attention respectively; and $\mathbf{s}_{i,j}^{(arc)}$ is the probability distribution for the corresponding arc. Label scores are calculated similarly by using another 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.

POS Tagging, Morphological Tagging, and NER. We use task-specific linear layers followed by a softmax along output classes to score labels as in a standard neural sequence labeling architecture:

$$\mathbf{s}_i^{(label)} = \text{softmax}(\mathbf{W}_{task}^T \mathbf{r}_i) \tag{5}$$

where $\mathbf{W}_{task}$ are the weights of the linear layer for the target task and $\mathbf{s}_i^{(label)}$ is the probability distribution for the output labels for word i.

For morphological tagging, we experiment both with predicting morphological attributes of a word as a separate label, namely, an unfactored tag string (Inoue, Shindo, and Matsumoto 2017), and jointly predicting the value of each individual morphological attribute such as aspect, case, tense, and number (Kondratyuk et al. 2018). The first method predicts one label per word by concatenating all its morphological tags. The latter method predicts each morphological tag separately with separate softmax layers

together with the unfactored tag string, as shown below for English word `hope`. We will show in Section 6.2 that the second method helps our model to learn infrequent tags better with higher overall performance.

	`hope`	#labels
Unfactored Morph. Tag	Sing; 1; Pres	1
+ Separate Attributes	Num = **Sing**, Per = **1**, Tense = **Pres**	4

3.2 Contextual Language Adapter

Instead of fine-tuning the whole encoder network on the downstream task, which would imply updating all base model's parameters, we use adapter layers (Rebuffi, Bilen, and Vedaldi 2018; Houlsby et al. 2019), or simply **adapters**. Adapters are lightweight modules that are inserted between Transformer layers. When fine-tuning with adapters, the weights of the base model remain frozen, while the adapter weights are updated for the downstream task (Stickland and Murray 2019) and/or the target language (Pfeiffer et al. 2020). Adapters provide a parameter efficient way to perform fine-tuning, while acting as information modules for the downstream task or target language, whereas the original network serves as a memory of the generic language knowledge.

In this work, we propose *contextual language adapters* that are adapter layers in which weights of the parameters are generated by a hypernetwork (see Section 3.3) to capture language-specific information without limiting the generalization across multiple languages. Because our model is trained for one task at a time, our contextual

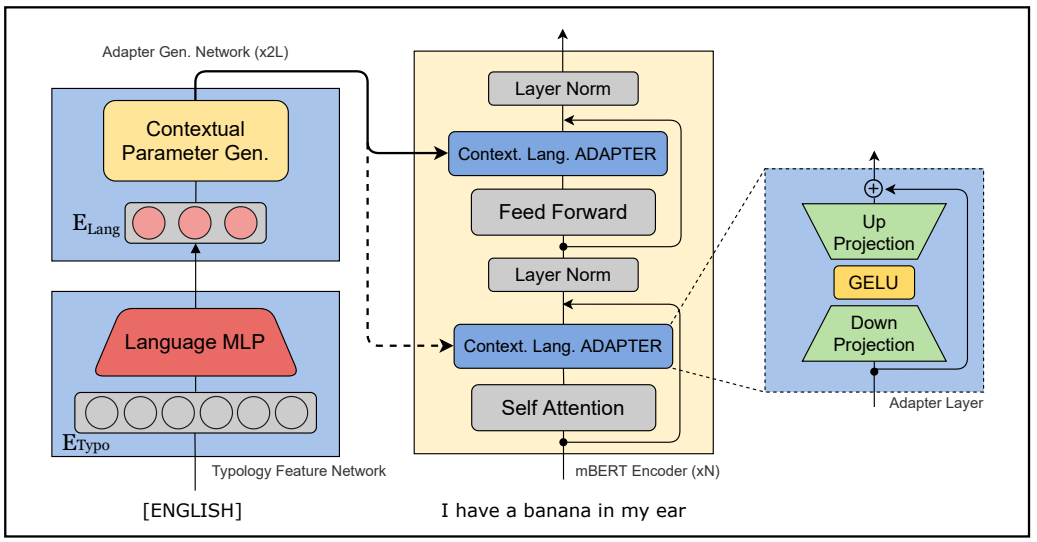


Figure 2
UDapterX architecture with adapter generation network and typology feature network. The adapter generation network includes a contextual parameter generator that takes language embeddings as inputs and generates language-specific adapter parameters for each layer of the mBERT encoder. The typology feature network consists of an multilayer perceptron (MLP) that learns to project typological features to language embeddings.

language adapters enable task-specific as well as language-specific adaptation for the base multilingual pre-trained model by infusing language-specific knowledge with the target task.

In UDapter, as adapter modules, we use a simple feedforward network with a Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel 2016) nonlinear activation function, as shown in Figure 2. More formally, a contextual language adapter layer CLA_i at layer i consists of a down projection $W_{down} \in \mathbb{R}^{h \times b}$ with bottleneck dimension b, a nonlinear function and an up projection $W_{up} \in \mathbb{R}^{b \times h}$ combined with a residual connection with the input $z_i \in \mathbb{R}^h$:

$$\text{CLA}_i(z_i) = \mathbf{W}_{up}^T \text{GELU}(\mathbf{W}_{down}^T z_i) + z_i \tag{6}$$

Bias terms are omitted for clarity. Similar to Houlsby et al. (2019), we insert an adapter layer after each multi-head attention block and feed-forward block of Transformer layers.[5]

3.3 Adapter Generator Network

In order to control the amount of sharing and generalization across languages while being able to adapt to each language separately, we use the adapter generator network. An adapter generator network consists of a contextual parameter generator (CPG) (Platanios et al. 2018) and language embeddings that are learned simultaneously. CPG is a trainable hypernetwork that generates the parameters of adapter layers as a function of language embeddings for each language. Conceptually, it enables our model to retain a high degree of multilinguality without losing performance on individual languages, for better multilingual training. CPG allows this with language-specific parameterization during multilingual training but those parameters are generated by a hypernetwork that is trained for all languages, which ensures the soft parameter sharing.

Formally, a contextual parameter generator CPG_i at layer i is defined as a function of language embedding $\mathbf{l}_n$ for a language n:

$$\theta_{CLA_i} \triangleq \text{CPG}_i\ (\mathbf{l}_n) \tag{7}$$

where θ_{CLA_i} denote the parameters of the contextual language adapter (CLA) in layer i. Similar to Platanios et al. (2018), we implement CPG as a simple linear transform of a language embedding. Thus, the weights of adapter layers are generated by the dot product between language embeddings $\mathbf{l}_n$ and the trainable parameter generator tensor $\mathbf{W}_{CPG_i}$:

$$\text{CPG}_i\ (\mathbf{l}_n) = \mathbf{W}_{CPG_i}^T \mathbf{l}_n \tag{8}$$

where $\mathbf{l}_n \in \mathbb{R}^d$, $\mathbf{W}_{CPG_i} \in \mathbb{R}^{d \times \text{P}_{CLA}}$, d is the language embedding size, P_{CLA} is the number of parameters for a contextual language adapter layer. An important advantage of CPG is that it can easily be applied to an existing model to generate additional parameters, without dramatically increasing the number of trainable parameters.

5 For simplicity, we mention only one contextual language adapter (CLA) per layer in Equation (6) and remaining definitions.

As a result of using the adapter generator network, we do not train adapter layers separately for each language, but instead we train CPG and language embeddings for all languages. As a key part of UDapter, language embeddings are learned together with the downstream task.

3.4 Typology Feature Network

The adapter generator network enables our model to learn the target task depending on language embeddings. Although this allows UDapter to perform well for languages that are included in fine-tuning, learning the task for a language that is unseen in training—zero-shot—still is a problem as its language embedding is not available. Inspired by Naseem, Barzilay, and Globerson (2012) and Ammar et al. (2016), we address this problem by defining language embeddings as a function of a large set of language typology features.

Assuming a language n can be associated with a vector of typological features, we compute its language embedding using a 2-layer feedforward network with a ReLU activation (MLP):

$$\mathbf{l}_n = \mathbf{W}^T_{lang,2}\text{ReLU}(\mathbf{W}^T_{lang,1}\mathbf{t}_n) \tag{9}$$

where $\mathbf{t}_n \in \mathbb{Z}^k_2$, $\mathbf{W}_{lang,1}$, $\mathbf{W}_{lang,2} \in \mathbb{R}^{k\times k}, \mathbb{R}^{k\times d}$, k is the typology feature vector size and d is the language embedding size. This typology feature network is trained jointly with the adapter generator network so that, during inference, our model is capable of computing a language embedding from its typological features, even for unseen—zero-shot—languages.

For typological feature vectors, we use 103 syntactic, 28 phonological, and 158 phonetic inventory features of languages from URIEL.[6] The URIEL database is a collection of binary features extracted from multiple typological and phylogenetic sources, such as WALS (World Atlas of Language Structures) (Dryer and Haspelmath 2013), PHOIBLE (Moran and McCloy 2019), Ethnologue (Lewis, Simons, and Fennig 2015), and Glottolog (Hammarström et al. 2020). Since for many languages some feature values are missing, URIEL also provides feature values that were *predicted* by a k-nearest neighbors approach based on average of genetic, geographical and feature (e.g., syntax, phonology) distances between languages. Our main model (Section 6.1, 6.2) makes use of these predicted values along with the gold ones.

As an alternative approach, in the following section, we also investigate how to predict those missing features jointly with the target task in order to inhibit possible loss caused from an external prediction system. We hypothesize that learning to predict such missing features jointly with the target task may benefit downstream performance (Ponti et al. 2019).

4. Joint Typological Feature Prediction

One of the crucial components of UDapter is the language embedding ($\mathbf{l}_n$), because contextual parameter generation is conditioned on it. Our model learns language embeddings from typological feature vectors representing a large variety of syntactic

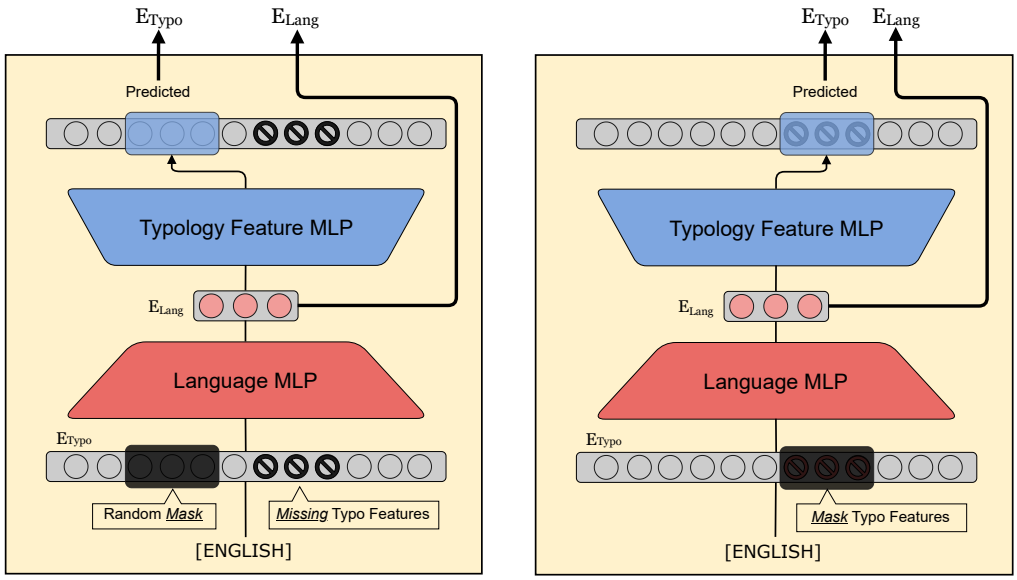
6 We use the `lang2vec` Python library (Littell et al. 2017) to obtain language typology features: `https://github.com/antonisa/lang2vec`.

and phonological attributes of languages. In practice, however, several human-curated feature values are missing from the URIEL database in many languages, especially low-resourced ones. For instance, out of 289 syntactic and phonological typology features, Kazakh (kk), Belarussian (be), and Tamil (ta) miss 275, 256, and 65 values, respectively. In our base model (Üstün et al. 2020), we use feature values that are predicted by Littell et al. (2017) using a KNN approach based on existing features. However, using externally predicted values carry potential errors that are propagated from off-the-shelf systems and limits the interaction between typological features and the target task.

To mitigate error propagation and gain downstream performance, we propose a novel prediction component and multi-task training for learning typological feature prediction jointly with the target task. This approach allows us to run our model without an external prediction system for missing features at inference time.

To implement a joint typological feature prediction component, we use a random masking strategy. Figure 3 gives an overview of this component: During training (3a), we randomly mask a portion of typological features that have annotated values in URIEL (see Section 7.4 for the impact of masking ratio), and extend the typology feature network to predict those masked features. For that, we use a one-layer feedforward network that takes a language embedding $\mathbf{l}_n$ and projects it to a new typological feature vector $\widehat{\mathbf{t}_n}$:

$$\widehat{\mathbf{t}_n} = \text{ReLU}(\mathbf{W}_{typo}^T \mathbf{l}_n) \tag{10}$$



(a) Training (b) Inference

Figure 3
Typological feature network with the *Typology Feature MLP* for jointly predicting features. The last layer takes language embeddings and projects them to typological feature vectors. During training (3a), we mask features with annotated values for learning to predict. In the inference time (3b) we mask originally missing features as our model is now capable of computing language embeddings with masked features.

where $\mathbf{W}_{typo}$ are the weights of the feedforward network. After that, we compute the loss between predicted values and original values for the masked features, and add it to the loss of the target task:

$$\mathbf{L} = L_{task} + \lambda L_{typo} \tag{11}$$

where $\mathbf{L}$ is the total loss of the model, and L_{task}, L_{typo} are the loss values for the target task and typological feature prediction, respectively. λ is the loss weight (scaling factor) to control the contribution of the typological prediction to the total loss. In this way, the model learns language embeddings not only from the feedback of the target task, but also from the feedback of typological feature prediction. At inference time (3b), we only provide the available features to the model and mask out the missing features.

5. Experimental Setup

Data and Training Details. Following Kulmizev et al. (2019), we selected 13 high-resource languages to multilingually fine-tune UDapter. These languages[7] are morpho-syntactically diverse, they belong to different language families, and they have high-quality training data of different sizes. During training, gold tokenization is provided and a language identifier is added to each sentence so that the model can distinguish languages for adapter generation. For the zero-shot experiments, we chose 30 genuinely low-resource languages from Universal Dependency (UD), and evaluate the models without further fine-tuning.[8] The detailed language list is provided in Appendix A. For dependency parsing, POS tagging, and morphological tagging, we used UD 2.3 (Nivre et al. 2018) to compare with previous work, whereas for NER we use WikiANN (Pan et al. 2017) with the standard split proposed by Rahimi, Li, and Cohn (2019).

For the mBERT model, we use `BERT-multilingual-cased` with its WordPiece tokenizer. Because UDapter predicts a label per word or word-pair, we pass the mBERT output corresponding to the first wordpiece per word to the task-specific top layers. We use the same hyperparameter values as Kondratyuk and Straka (2019). As extra hyperparameters we use 256 and 32 for adapter size and language embedding size, respectively. For dependency parsing we train UDapter for 80 epochs, and for other tasks we train each model for 30 epochs. During training, we use a learning rate of 0.001 with an inverse square root learning rate decay and a linear warm-up (Howard and Ruder 2018) for 1 epoch. Appendix B gives the hyperparameter details for UDapter and the baseline models. Finally, we evaluate each model in terms of labeled attachment score (LAS) for parsing, accuracy for POS tagging, and F1 score for morphological tagging and NER.[9]

Baseline. For all the tasks, we compare UDapter mainly with UDify (Kondratyuk and Straka 2019). UDify was designed to fine-tune all mBERT parameters for UD parsing and was originally trained on the concatenation of all UD treebanks. To enable a direct comparison, we train UDify (**multi-udify**) on the same 13 high-resource languages

7 Arabic (`ar`), English (`en`), Basque (`eu`), Finnish (`fi`), Hebrew (`he`), Hindi (`hi`), Italian (`it`), Japanese (`ja`), Korean (`ko`), Russian (`ru`), Swedish (`sv`), Turkish (`tr`) and Chinese (`zh`).

8 For this reason, the terms "zero-shot" and "low-resource" are used interchangeably in this article.

9 We use the official CoNLL 2018 Shared Task script (`https://universaldependencies.org/conll18/evaluation.html`) for UD tasks and `seqeval` (`https://github.com/chakki-works/seqeval`) for NER.

separately for each task (just like UDapter). For dependency parsing, in addition to multi-udify, we also compare UDapter with monolingually trained UDify models (**mono-udify**), and the following parsers:[10]

1. UUparser+BERT (Kulmizev et al. 2019), a graph-based BLSTM 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 pre-trained word embeddings and character representations;[11]
3. The original UDify (Kondratyuk and Straka 2019).

Furthermore, we also evaluate a variant of our model on parsing to see the impact of standard adapter modules. This variant, *adapter-only*, is a model with only task-specific adapter layers and no language specific adaptation (i.e., no adapter generation network). Finally, to understand the importance of language-typology in UDapter, we evaluate one last model variant, *udapter-proxy*, trained *without* typological features (i.e., no typology feature network). In this model variant, a separate language embedding is learned from scratch for each in-training language, and for zero-shot languages we use one from the same language family, if available, as proxy language embedding.

Note that all baselines are either trained for a single language, or multilingually without any language-specific adaptation. By comparing UDapter to these baselines, 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.

6. Main Results

In this section we present results and comparisons between UDapter and baselines for both high-resource languages and zero-shot experiments. We start with results on dependency parsing as our core task, and continue with the results for POS, morphological tagging, and NER. At the end of the section, we discuss the impact of jointly predicting typological features on parsing.

6.1 Dependency Parsing

Labeled Attachment Scores (LAS) for the *high-resource* languages are given in Table 1. UDapter consistently outperforms both the monolingual and multilingual baselines in all languages, showing the success of our proposed adaptation approach. Statistical significance testing[12] applied between UDapter and multi/mono-udify confirms that

10 There are other monolingual dependency parsers (Wang and Tu 2020; Fernández-González and Gómez-Rodríguez 2021; Yang and Tu 2021) achieving state-of-the art results in different settings. These studies focus on parsing algorithms rather than scaling a single parser to multiple languages or zero-shot parsing. Moreover, their experimental setups do not match ours, which would make it hard to provide a fair comparison.

11 UDPipe scores are taken from Kondratyuk and Straka (2019).

12 We used *paired bootstrap resampling* to check whether the difference between two models is significant ($p < 0.05$) by using Udapi (Popel, Žabokrtský, and Vojtek 2017).

Table 1
Dependency parsing results (LAS) of baselines and UDapter for high-resource languages. The last column shows average LAS of the 13 high-resource languages (HR-AVG). Results for previous work are taken from Kulmizev et al. (2019) [1] and Kondratyuk and Straka (2019) [2,3].

	ar	en	eu	fi	he	hi	it	ja	ko	ru	sv	tr	zh	HR-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
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
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
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**

Table 2
Zero-shot results on a subset of 30 low-resource languages for dependency parsing (LAS). Udapter-proxy refers to a variant of UDapter without typological features, where a proxy language embedding is used when available. The asterisk indicates that average is calculated over all low-resource languages. (Results for all low-resource languages, together with chosen proxy, are given in Appendix C.)

	be	br	bho	fo	hsb	kk	mr	olo	sa	ta	te	tl	yo	LR-AVG*
multi-udify	80.1	60.5	37.2	68.6	53.2	61.9	46.4	42.1	19.4	46.0	71.2	62.7	41.2	35.3
udapter-proxy	69.9	–	35.9	64.1	44.4	45.1	29.6	41.1	15.1	–	–	–	–	–
udapter	79.3	58.5	37.3	69.2	54.2	60.7	44.4	43.3	22.2	46.1	71.1	69.5	42.7	36.5

UDapter's performance is significantly better than the baselines in 11 out of 13 languages (all except *en* and *it*).

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. Furthermore, to understand the relevance of adapters, we also evaluate a model with standard task-specific adapters. Interestingly, this adapter-only model considerably outperforms multi-udify (85.0 vs. 83.0), indicating that adapter modules are also effective in multilingual scenarios. UDapter, however, achieving consistent gains over both multi-udify and adapter-only in this setup (87.3 vs. 86.0, 85.0), demonstrates the importance of linguistically informed adaptation for in-training languages.

For the zero-shot experiments, average LAS on the 30 low-resource languages are shown in Table 2. Overall, UDapter slightly outperforms the multi-udify baseline (36.5 vs. 35.3), 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 13 representative languages in our low-resource set. Here we find a mixed picture: UDapter outperforms multi-udify on 8 out of 13 languages.[13] Achieving improvements on high-resource

13 LAS scores for all 30 languages are given in Appendix C. By significance testing, UDapter is *significantly* better than multi-udify on 16/30 low-resource languages, which is shown in Table C1.

Table 3
Results of UDapter and UDify (Kondratyuk and Straka 2019) trained separately for POS tagging, morphological tagging, and NER on 13 high-resource languages. HR-AVG is calculated over all 13 training languages, where ja and kr are removed from morphological tagging.

	ar	en	eu	fi	he	hi	it	ja	ko	ru	sv	tr	zh	HR-AVG
POS Tagging (Accuracy)														
multi-udify	96.3	96.6	94.5	96.4	96.5	96.7	98.3	96.8	95.3	98.5	97.6	93.4	93.8	96.1
udapter	96.8	97.0	95.7	97.3	97.1	97.4	98.3	97.0	96.5	98.9	98.4	95.1	95.1	97.0
Morphological Tagging (F1)														
multi-udify	95.7	97.5	92.0	94.7	94.9	97.0	98.9	–	–	97.6	97.6	90.9	98.9	95.9
udapter	96.8	98.0	95.4	96.7	96.2	97.7	99.2	–	–	98.3	98.3	94.8	99.2	97.3
NER (F1)														
multi-udify	87.9	84.1	91.3	90.9	84.7	86.8	91.3	68.8	87.3	88.6	94.4	91.9	78.8	86.7
udapter	89.4	85.4	92.7	92.4	86.7	89.5	92.4	71.9	88.8	89.8	95.8	93.1	81.3	88.4

languages while not degrading zero-shot performance is notoriously very difficult. Thus, we believe our zero-shot results represent an important step toward overcoming the problem of positive/negative transfer trade-off.

Finally, the results of udapter-proxy show that choosing a proxy language embedding from the same language family performs clearly worse than UDapter, apart from not being available for many languages. This indicates the importance of typological features in our approach (see Section 7.3 for further analysis).

6.2 Sequence Labeling Tasks

Table 3 shows results for UDapter and multi-udify on POS tagging, morphological tagging, and NER in high-resource languages. As in dependency parsing, UDapter outperforms multi-udify on all three tasks for all languages, confirming that a language-aware adaptation via language-typology can be beneficial across sequence labeling tasks.

However, results in the zero-shot setup are not always positive. Table 4 shows model performance on 13 low-resource languages for each task.[14] In POS tagging, UDapter outperforms UDify but improvements are rather small compared to high-resource languages (58.4 vs. 58.0).

In morphological tagging, we observed a very small decrease on average but a large variation across languages. Here, jointly predicting separate morphological attributes, together with the unfactored tag string (Inoue, Shindo, and Matsumoto 2017), helps the model to learn infrequent tags. For high-resource languages this increase is rather small (+0.3 F1 on average); however, for low-resource languages, jointly predicting morphological attributes provides a higher increase of +1.3 F1 (Table 5).

In NER, unlike the other three tasks, UDapter has significantly lower performance overall and it is outperformed by multi-udify on 6 out of 10 low-resource languages. See Section 7.1 for a further comparison between tasks.

Finally, when looking at the udapter-proxy results, selecting a proxy language from the same language-family generally makes results worse. In POS and morphological

14 Results for all low-resource languages are given in Appendix C.

Table 4
Zero-shot results on subset of 30 low-resource languages for sequence labeling tasks. Udapter-proxy refers to an alternative of UDapter without typological features where a proxy language embedding is used when it is available. The asterisk indicates that average of all low-resource languages. (Results for all low-resource languages, together with chosen proxy, are given in Appendix C.)

	be	br	bho	fo	hsb	kk	mr	olo	sa	ta	te	tl	yo	LR-AVG*
POS Tagging (Accuracy)														
multi-udify	94.6	70.6	61.8	80.0	76.8	85.0	67.5	74.5	41.4	68.6	81.6	73.3	65.1	58.0
udapter-proxy	74	–	61.0	78.5	67.3	78.4	58.5	75.2	39.0	–	–	–	–	–
udapter	96.9	72.2	63.1	79.6	77.8	83.4	66.5	76.6	42.2	70.3	84.2	78.4	63.7	58.4
Morphological Tagging (F1)														
multi-udify	92.3	51.1	68.1	58.4	60.7	52.2	51.7	60.0	40.7	37.8	–	31.2	58.7	44.4
udapter-proxy	67.5	–	66.7	55.4	56.3	51.0	50.1	59.9	38.3	–	–	–	–	–
udapter	92.0	59.9	66.4	56.3	37.4	52.5	51.2	62.4	20.6	41.7	–	33.3	64.1	44.3
NER (F1)														
multi-udify	80.1	77.6	–	78.2	79.0	72.7	75.9	–	54.0	69.8	62.2	81.4	–	67.4
udapter-proxy	70.3	–	–	65.7	32.9	60.7	71.6	–	41.0	–	–	–	–	–
udapter	80.3	74.3	–	80.3	79.7	65.9	74.0	–	47.0	69.0	62.4	68.9	–	65.6

Table 5
Morphological tagging results for high-resource (HR) and low-resource languages (LR) with different tagging architectures.

	HR	LR
Unfactored Morph. Tag	97.0	43.0
+ Separate Attributes	97.3	44.3

tagging this model can compete with multi-udify and udapter for some languages; however for NER, the difference between udapter and udapter-proxy is higher.

6.3 Typological Feature Prediction

In this section, we present dependency parsing results when jointly predicting typological features during training. Table 6 shows the results for both high-resource languages and zero-shot experiments. The number of missing features is given for each language (# *missing*), and LR-AVG* shows the average LAS score over 30 languages. Here, we evaluate several variants of UDapter to measure the benefits of joint typological feature prediction:

1. No-prediction (**no-pred**): This model uses 289 typological features (syntax + phonology + inventory) without predictions for missing typological features. For missing features we use an `NA` token, the embedding of which is learned during training.
2. Typological feature prediction (**typo-pred**): UDapter with joint typological feature prediction. This model uses 289 typological features.

Table 6
Dependency parsing (LAS) results for UDapter with and without typological feature prediction. # *missing* shows the number of unavailable features in 289 typological features (syntax + phonology + inventory). Note that the base UDapter uses KNN predictions (Littell et al. 2017) for missing typological features. Results for all low-resource languages are given in Appendix D.

High-Resource Languages														
	ar	en	eu	fi	he	hi	it	ja	ko	ru	sv	tr	zh	HR-AVG
# *missing* →	5	3	21	7	41	24	54	13	17	16	51	16	3	–
no-pred	84.4	90.0	83.1	89.2	89.3	91.9	93.5	92.5	85.7	92.3	90.2	69.4	82.6	87.2
typo-pred	84.3	89.4	83.2	89.2	89.4	91.9	93.7	92.9	85.8	92.4	90.0	70.2	83.6	87.4
KNN	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
Low-Resource Languages (Zero-Shot)														
	be	br	bho	fo	hsb	kk	mr	olo	sa	ta	te	tl	yo	LR-AVG*
# *missing* →	256	41	229	256	289	275	65	289	112	61	47	31	13	–
no-pred	59.8	61.9	31.0	61.6	42.5	46.5	45.2	31.0	20.2	44.8	70.5	65.4	42.7	32.0
typo-pred	75.1	60.7	33.6	66.7	48.3	58.6	44.2	36.7	23.0	44.9	71.0	66.8	42.6	35.0
KNN	79.3	58.5	37.3	69.2	54.2	60.7	44.4	43.3	22.2	46.1	71.1	69.5	42.7	36.5

During training, a random subset of features[15] is masked and predicted during training. At test time, only missing features are masked and the model computes language embeddings without the need for external prediction.

3. KNN-based external prediction (KNN): The base UDapter model that uses values predicted by Littell et al. (2017) using a K-Nearest Neighbors method for missing features.

For the high-resource languages, as UDapter learns language embeddings from training data as well as typological features, differences are overall small. The model without missing feature prediction (no-pred) slightly underperforms the base model with KNN predictions (87.2 vs. 87.3 average LAS), whereas the one with joint prediction (typo-pred) slightly outperforms it (87.4 vs. 87.3 LAS). This shows that the relevant typological information can indeed be learned directly from task-specific training data, but the baseline KNN prediction approach remains hard to beat.

In the zero-shot experiments, where typological features are the only source to compute language embeddings, typological feature prediction significantly affects parsing results: Disabling missing feature prediction clearly hurts parsing accuracy (32.0 vs. 36.5 average LAS by the baseline KNN prediction). As expected, this effect is especially dramatic for languages with many missing features like Belarusian (be) and Kazakh (kk). Jointly predicting typological features (typo-pred) increases overall accuracy (35.0 vs. 32.0 LAS) and substantially recovers the large drop observed in languages with many missing features. However, contrary to our expectations, joint prediction falls behind the external KNN-based predictions in zero-shot parsing performance (35.00 vs. 36.5 LAS). Note that the external KNN-based feature predictor also makes use of geographical features (geo) from URIEL (Littell et al. 2017). Geographical features represent distances

15 In the best performing model, the masking ratio is 0.2. Please see Section 7.4 for the impact of the masking ratio.

from fixed points on the surface of the earth (i.e., geographical locations). When we also use these geographical features for joint typological prediction in UDapter, average LAS increases (+1.0) leading to a very competitive model that does not need external typology prediction. See Section 7.4 for further analysis of the impact of geographical features and the accuracy of feature prediction.

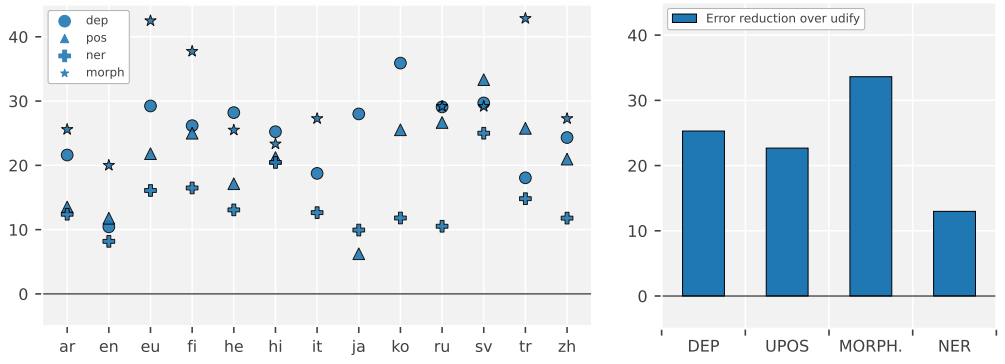
7. Analysis

In this section, we investigate when and how UDapter is most effective. We start by comparing the impact of typology-based language adapters on different tasks. After that, we focus only on dependency parsing and provide in-depth analyses of different dimensions of our model, namely: improvements on different languages, impact of modeling typology and joint feature prediction, learned language representations, and impact of contextual parameter generation.

7.1 Which Tasks Improve Most?

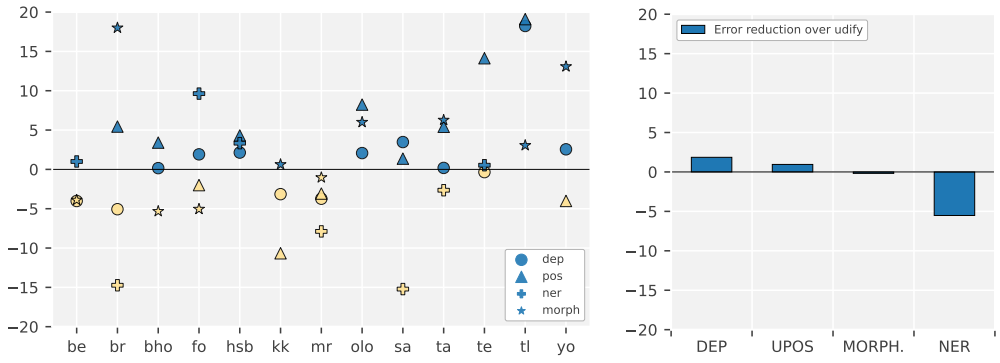
Figures 4 and 5 show the error reduction (%) achieved by UDapter over the UDify baseline (multi-udify). For high-resource languages, Figure 4 shows individual results per language (a) and average results per task (b). In this setup, UDapter reduces error rate in all the four tasks by significant ratios. Although dependency parsing shows the largest absolute increase (+4.3 LAS), morphological tagging is the task that benefits most from our approach in terms of error reduction. Especially for languages with complex morphology such as Turkish and Finnish, UDapter substantially increases morphological tagging quality. The smallest error reduction (12%) is that of NER. Overall, these results confirm that, for languages with enough training data, typology-based language-specific adaptation is robust among different tasks.

In the zero-shot setup (Figure 5), results differ per task. In dependency parsing, UDapter outperforms multi-udify with +1.2 LAS and reduce errors by 1.9%. In POS tagging, the error reduction is very small: 1.0% in favor of UDapter; however, when



(a) (b)

Figure 4
Error reduction rate (%) by UDapter over the UDify baseline (*multi-udify*) on high-resource languages. Plot (a) and (b) show error reduction rate per language and average for each task, respectively. A black horizontal line denotes zero error reduction.



(a) (b)

Figure 5
Error reduction rate (%) by UDapter over the UDify baseline (multi-udify) on low-resource languages. Plot (a) and (b) show error reduction rate per language and average for each task, respectively. A black horizontal line denotes zero error reduction. Language/task combinations that are negatively affected by UDapter are marked in yellow in (a). Note that these results refers to zero-shot experiments where no training data is available for the target language.

looking at the individual language results, 20 out of 30 languages have the same trend (either positive or negative) as dependency parsing, suggesting these two tasks share similar characteristics. In morphological tagging, UDapter performs almost the same as UDify on average, but results vary significantly among languages. We believe this may be due to the highly language-specific nature of morphological attributes, which makes it particularly hard to successfully adapt a model without target-language examples. Finally for NER, UDapter significantly underperforms multi-udify, increasing error rate by 5.5% on average in the zero-shot setup.

A possible explanation for the zero-shot results is that dependency parsing and POS tagging benefit more from typological knowledge, whereas NER mostly requires lexical coverage, which is not affected by adapters. Moreover, the low representation quality of low-resource languages in mBERT (Wu and Dredze 2020) may have a stronger negative effect on NER, which can make it more difficult to learn language-specific adapters rather than multilingual fine-tuning for those languages. However, the reason for the relatively low performance of typology-based adaptation on zero-shot NER remains an open question.

7.2 Which Languages Improve Most?

Figure 6 shows the relative gains in dependency parsing accuracy (LAS) for UDapter over multi-udify. Each bar represents one of the 13 fine-tuning languages along with the respective treebank 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 (Figure 7), the difference between the two models is smaller than for high-resource languages (+1.2 LAS, as seen in Table 1). While it is harder to find a trend here among these languages, we notice that UDapter is typically beneficial for the languages *not* present in the mBERT training corpus: It *significantly* outperforms

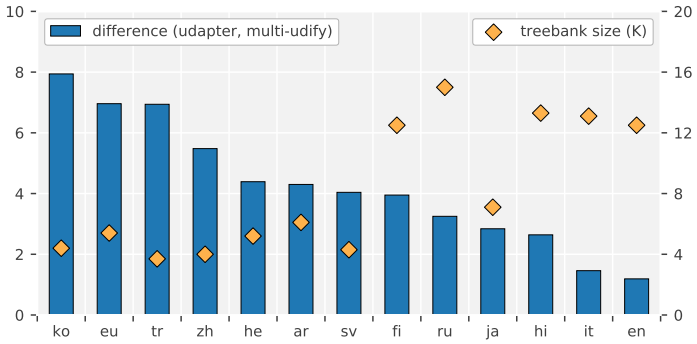


Figure 6
Difference in dependency parsing results (LAS) between UDapter and multi-udify for high-resource languages. Diamonds indicate the number of sentences in the corresponding treebank.

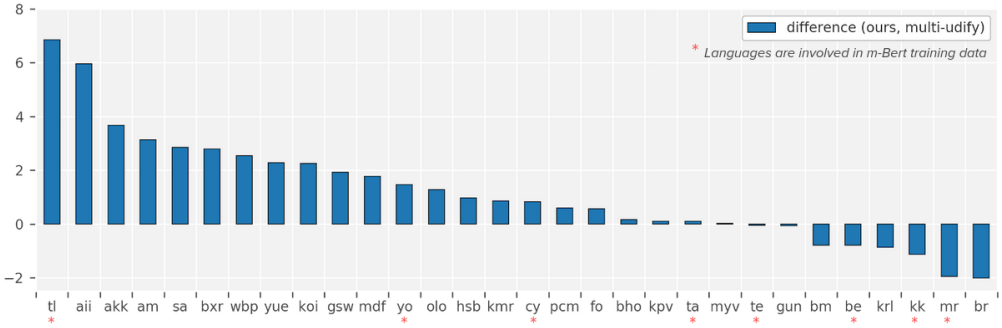


Figure 7
Difference in dependency parsing results (LAS) between UDapter and multi-udify for low-resource languages (zero-shot). '*' indicates languages that are involved in mBERT pretraining.

multi-udify in 13 out of 22 (non-mBERT) languages. This suggests that typological feature-based adaptation may lead to improved sentence representations when the pre-trained encoder has not been exposed to a language.

7.3 How Much Gain from Typology?

UDapter learns to represent languages based on their typological features. A natural alternative to this choice is to learn language embeddings from scratch. To compare UDapter with this alternative, we train another model for dependency parsing, in which a separate language embedding (of the same size: 32) is initialized randomly for each fine-tuning language and learned end-to-end. Because such embeddings are only available for the high-resource languages, in the zero-shot experiments we use the average, or **centroid**, of the 13 learned language embeddings. As shown in Figure 8a, 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 experiments, however,

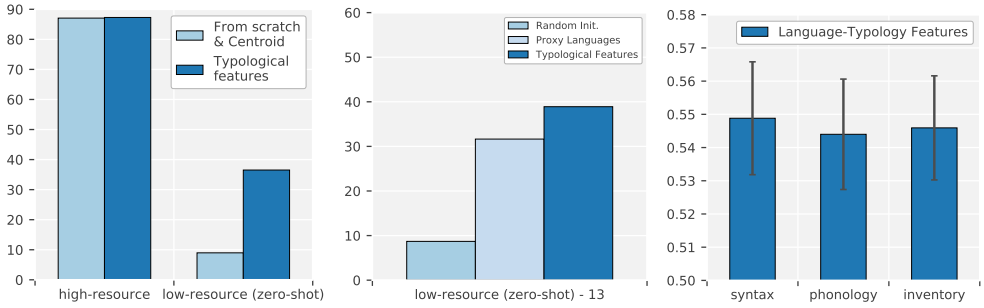


Figure 8
(a) Impact of language typology features on parsing performance (LAS). (b) Average zero-shot parsing results for 13 low-resource languages with a proxy. (c) Average normalized feature weights obtained from linear projection layer of the language embedding network.

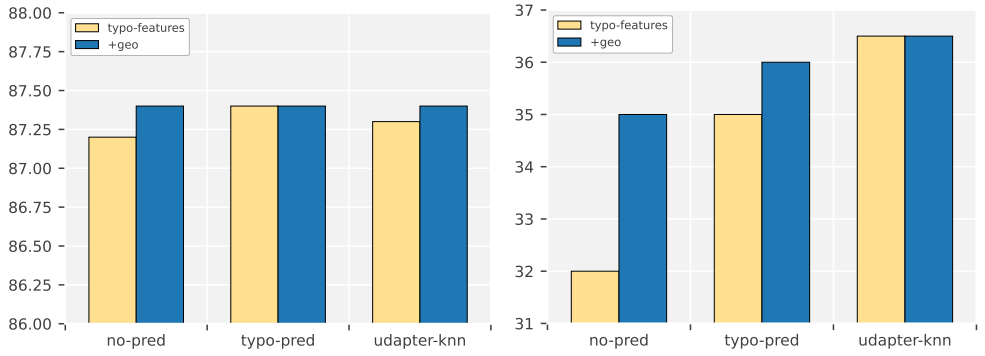
the use of centroid embedding performs very poorly: 9.0 vs. 36.5 average LAS over 30 languages.

As discussed in Section 6, a better alternative is using the embedding of a *proxy* language belonging to the same family as the evaluated low-resource language, if available. In our setup, this is possible for 13 low-resource languages (see Appendix C for the detailed list). As shown in Figure 8b, UDapter outperforms proxy embeddings even on this subset of languages.

Taken together, these results show that a model can learn reliable language embeddings directly from the data. However, this type of adaptation can only benefit in-training languages at the expense of zero-shot performance. By contrast, using typological information to condition language embeddings allows UDapter to achieve improvements on the high-resource languages while preserving (or even improving) zero-shot performance. As opposed to our model, UDify (multi-udify) achieves competing performance in zero-shot languages with a cost in parsing quality for high-resource languages due to the limited language-specific capacity.

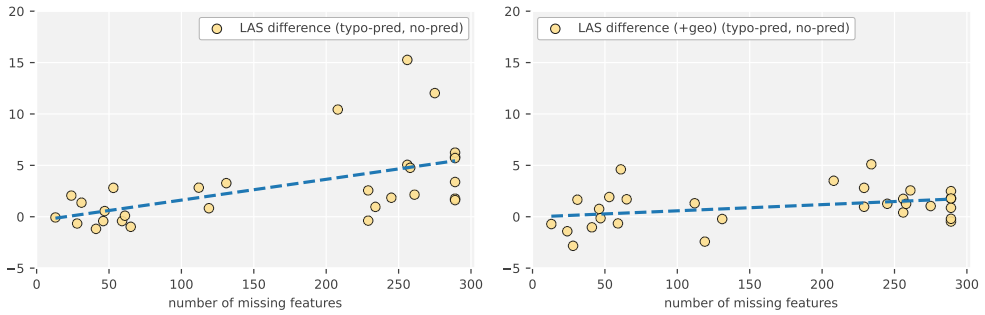
7.4 How and When to Jointly Predict Typological Features?

The main results (Section 6.3) show that predicting typology jointly with dependency parsing leads to similar parsing accuracy as our base model. Moreover, when compared to no-prediction models, joint prediction improves zero-shot parsing performance in terms of LAS. In the main experiments, we used all syntax, phonology, and inventory features, for a total of 289 binary features, some of which are not annotated for many languages. However, URIEL (Littell et al. 2017) also provides geographical features (geo) representing distances to fixed points on the surface of the earth (i.e., geographical locations). These features are instead available for all languages. Figure 9 shows the parsing results when geo features are used in the corresponding model. Although the contribution of geo features for high-resource languages is very limited (9a), for low-resource languages, geo features considerably improve parsing performance, leading to a model that is very competitive with udapter-knn with the additional advantage of not requiring external predictions. By contrast, udapter-knn does not benefit from the addition of geo features as these were already used for KNN prediction.



(a) High-resource languages (b) Low-resource languages

Figure 9
Contribution of geographical features (**geo**) to the parsing performance for high-resource (9a) and low-resource languages (9b). Note that second plots refers to zero-shot experiments.



(a) Without geographical features (b) With geographical features

Figure 10
Performance difference in dependency parsing (+/−LAS) between typology prediction model and no prediction model. Each data point represents a low-resource language with the respective number of missing features. Trend lines (in blue) show the correlation between number of missing features and LAS gains.

For a closer look at the benefits of joint typological prediction, Figure 10 presents the LAS difference between the typology prediction model and the no-prediction model for each low-resource language, with and without geographical features (10a, 10b). Trend lines (in blue) confirm that, when more features are missing, UDapter tends to outperform the no-prediction model. Comparing 10a and 10b, using geographical features decreases the benefits of the joint prediction model, especially for languages with few missing features. However, predicting typology remains beneficial.

We also evaluate different masking ratios for the typological feature prediction network, and different scaling factors (λ) for the prediction loss. While the masking ratio determines the number of features to mask and predict during training, λ determines the contribution of typology prediction to the target task in our multi-task architecture.

The results, shown in Figure 11, suggest that UDapter with joint typology prediction performs better with smaller masking ratios, whereas λ does not have a consistent effect. In sum, UDapter works best with 20% masking and 0.8 scaling factor.

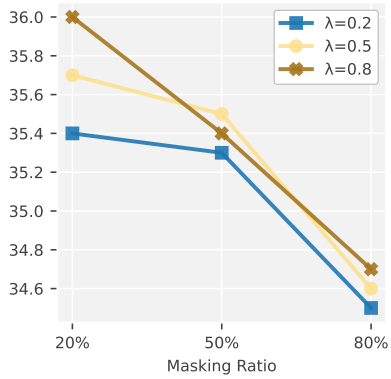


Figure 11
Effect of masking ratio and typology loss scaling factor (λ) on zero-shot parsing result.

Table 7
Accuracy of typological feature prediction over 5-fold cross validation for 6 high-resource languages (in-training) and 6 low-resource languages (zero-shot). Majority refers to the majority class baseline. Typo-pred and typo-pred**+geo** refer to our model without and with geographical features, respectively.

	ar	en	hi	it	sv	zh	be	br	fo	mr	sa	te
majority	61.1	64.5	62.8	72.3	67.7	63.4	51.0	63.5	55.1	67.8	70.7	70.1
typo-pred	90.0	91.9	93.9	93.5	85.6	86.2	83.3	82.5	86.0	79.8	80.6	80.7
typo-pred **+geo**	90.4	93.2	92.5	94.5	87.2	86.6	88.3	84.5	84.2	81.2	80.7	83.0

Finally, in addition to the impact on parsing performance, we also evaluate the *accuracy* of typological feature prediction by UDapter. Although the predicted values for missing features do not directly affect parsing performance, we expect better predictions should lead to better language embeddings for adapter generation. Table 7 shows the feature prediction accuracy of UDapter without and with geographical distances, for a subset of high- and low-resource languages.[16] This accuracy is calculated over 5-fold cross validation with 20% masking ratio.

As a reference point, we also present the results of a baseline selecting the majority class for each feature.[17] We find that UDapter strongly outperforms the majority baseline, showing the potential of extracting typology relevant information directly from (annotated) textual data (Ponti et al. 2019). Clearly, UDapter may also be strongly relying on universal correlations among features (Greenberg 1963) to predict the missing values. We leave a more in-depth investigation of these two factors and their interplay to future work.

7.5 How Does UDapter Represent Languages?

We start by analyzing the projection weights assigned to different typological features by the first layer of the language MLP (see Figure 2). Figure 8c shows the averages

16 Feature prediction accuracies for the full list of languages are given in Appendix D.
17 Although `lang2vec` provides knn-based predictions for the missing values, the KNN model itself is not available for evaluation on human-annotated features.

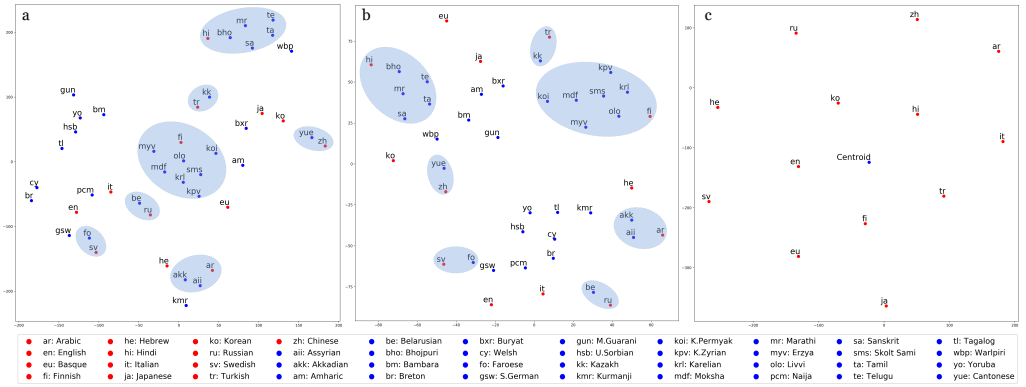


Figure 12
Vector spaces for (a) language-typology feature vectors taken from URIEL, (b) language embeddings learned from typological features by UDapter, and (c) language embeddings learned without typological features. High- and low-resource languages are indicated by red and blue dots respectively. Highlighted clusters in (a) and (b) denote sets of genetically related languages.

of normalized syntactic, phonological, and phonetic inventory feature weights when UDapter is trained for dependency parsing. 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.

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, Neubig, and Littell (2017). Figure 12 illustrates 2D vector spaces generated for the typological feature vectors $\mathbf{l}_t$ taken from URIEL (12a), as well as the language embeddings $\mathbf{l}_e$ learned by UDapter from scratch and from typological features (12b and 12c, respectively). The benefits of using typological features can be understood by comparing 12a and 12b: 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 12a and 12b. In fact, most groupings of genetically or geographically related languages, such as the Indian languages (*hi*-cluster) or the Uralic ones (*fi*-cluster) are largely preserved across these two spaces.

Comparing 12c to 12b where language embeddings are learned from scratch, the absence of typological features leads to a seemingly random space with no linguistic similarities (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 in Section 6.1, 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.

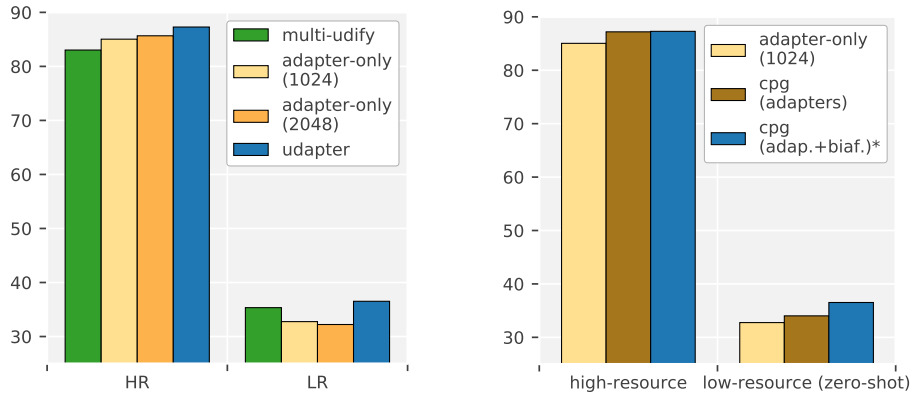


Figure 13
Impact of different UDapterX components on parsing performance (LAS): (a) adapters and adapter layer size, (b) application of contextual parameter generation to different portions of the network. In (b), the model named cpg (adap.+biaf.)* indicates the full UDapter.

7.6 Is Contextual Parameter Generation Really Essential?

In Section 6 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 (2,048 instead of the 1,024 used in the main experiments) for dependency parsing.

As seen in Figure 13a, increasing 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 standard, non language-adaptive 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 B for details), outperforms both adapter-only models, confirming once more the importance of this component.

For our last analysis (Figure 13b), we study soft parameter sharing via CPG on different portions of the network, namely: (1) Only on the adapter modules cpg (adapters) versus (2) 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.

8. Conclusion

We have presented UDapter, a multilingual model for dependency parsing and sequence labeling tasks such as POS, morphological tagging, and NER. It learns to adapt

language-specific parameters via contextual language adapters on the basis of adapter modules (Rebuffi, Bilen, and Vedaldi 2018; Houlsby et al. 2019) and the contextual parameter generation (CPG) method (Platanios et al. 2018). 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. Because not all typological features are available for every language, we further include typological feature prediction in our model in a multi-task manner that achieves very competitive parsing performance without the need of an external prediction system for missing features.

For dependency parsing, UDapter trained on a concatenation of typologically diverse languages outperforms strong monolingual *and* multilingual baselines on the majority of *both* high-resource (HR) and low-resource languages (LR)—zero-shot—which reflects its strong balance between per-language capacity and maximum sharing. Our in-depth analyses show that typological features are crucial to this success. For sequence labeling tasks, UDapter surpasses the multilingual baseline on HR languages and performs better or comparably to it on LR languages, except for a performance decrease in NER (see Section 7.1).

The main limitation in our approach remains the low representation quality for languages with zero or little data in the pre-trained encoder (multilingual pre-training). Sentences in such languages typically get tokenized very aggressively, leading to representations that are not very informative for the task-specific layers, regardless of any downstream language adaptation strategy. This could explain why our reported gains are overall more modest in LR languages than in HR ones. Considering this, promising research directions include improving the quality of the multilingual encoder representations for low-resource languages, for instance, by continued training on monolingual data (Pfeiffer et al. 2020) or learning a new vocabulary (Artetxe, Ruder, and Yogatama 2020) for the target language(s) of interest. More work also remains to be done to assess the quality of our joint typology prediction model, especially regarding the true contribution of the parsing loss to the reported prediction accuracy.

Appendix A: Language Details

Details of training and zero-shot languages such as language code, data size (number of sentences), and family are given in Table A1 and Table A2.

Appendix B: Experimental Details

Implementation. UDapter's implementation is based on UDify (Kondratyuk and Straka 2019). We use the same hyperparameters setting optimized in UDify without applying a new hyperparameter search. Together with the additional *adapter size* and *language embedding size* that are picked manually by parsing accuracy, hyperparameters are given in Table B1. To give a fair chance to the adapter-only baseline, we used 1,024 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. Additionally, we did not use a layer attention for either our model or the baselines.

Training Time and Model Size. Compared 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 when they are trained for dependency

Table A1
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	Family	Word Order	Train	Test
Arabic	ar	PADT	Afro-Asiatic, Semitic	VSO	6.1k	680
Basque	eu	BDT	Basque	SOV	5.4k	1,799
Chinese	zh	GSD	Sino-Tibetan	SVO	4.0k	500
English	en	EWT	IE, Germanic	SVO	12.5k	2,077
Finnish	fi	TDT	Uralic, Finnic	SVO	12.2k	1,555
Hebrew	he	HTB	Afro-Asiatic, Semitic	SVO	5.2k	491
Hindi	hi	HDTB	IE, Indic	SOV	13.3k	1,684
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*	6,491
Swedish	sv	Talbanken	IE, Germanic	SVO	4.3k	1,219
Turkish	tr	IMST	Turkic, Southwestern	SOV	3.7k	975

Table A2
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.

Language	Code	Treebank(s)	Family	Test
Akkadian	akk	PISANDUB	Afro-Asiatic, Semitic	1,074
Amharic	am	ATT	Afro-Asiatic, Semitic	101
Assyrian	aii	AS	Afro-Asiatic, Semitic	57
Bambara	bm	CRB	Mande	1,026
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	1,004
Erzya	myv	JR	Uralic, Mordvin	1,550
Faroese	fo	OFT	IE, Germanic	1,207
Karelian	krl	KKPP	Uralic, Finnic	228
Kazakh	kk	KTB	Turkic, Northwestern	1,047
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	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 B1
Hyperparameter setting. Note that typo. pred. mask and loss scaling factor only used in joint typology prediction model 4.

Hyperparameter	Value	Hyperparameter	Value
Dependency tag dimension	256	Typo. pred. mask ratio	0.2
Dependency arc dimension	768	Typo. pred. loss scaling factor (λ)	0.8
POS layer dimension	768	Optimizer	Adam
Morph. tagging layer dimension	768	β_1, β_2	0.9, 0.99
NER layer dimension	768	Weight decay	0.01
Batch size	32	Label smoothing	0.03
Dependency parsing epochs	80	Dropout	0.5
POS tagging epochs	30	BERT dropout	0.2
Morphological tagging epochs	30	Mask probability	0.2
NER epochs	30	Base learning rate	$1e^{-3}$
LR warm up ratio	0.125	BERT learning rate	$5e^{-5}$
Language embedding size	32	Adapter size	256

parsing. In terms of number of *trainable* parameters, UDify has 191M total number of parameters whereas UDapter uses 550M parameters in total, 302M for adapters (32×9.4M) and 248M for biaffine attention (32×7.8M), 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 is 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 (13×191M).

Appendix C: Zero-Shot Results

Table C1 shows LAS scores on all 30 low-resource 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. Moreover, Table C2 shows POS tagging (accuracy), morphological tagging (F1), and NER (F1) results for low-resource languages for UDapter and multi-udify.

Appendix D: Typological Feature Prediction

Tables D1 and D2 show accuracies for the typological feature prediction (Section 4) for each high-resource and low-resource languages, respectively. Note that for low-resource languages, we only evaluate the languages with at least 30 available typological features (except geo features). Finally, Table D3 shows the dependency parsing results (LAS) for UDapter with and without joint typological feature prediction for 30 low-resource languages (zero-shot). Note that our base model (UDapter) uses KNN-based predictions (Littell et al. 2017) for missing typological features, whereas typo-pred models use joint prediction model.

Table C1
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.

	orig. udify	multi-udify	udapter	udapter-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)
lr-avg	34.1	35.3	**36.5**	20.4

Table C2
POS tagging (Accuracy), morphological tagging (F1), and NER (F1) results of UDapter and multi-UDify models (Kondratyuk and Straka 2019) for all low-resource languages. Missing values in morphological tagging refers to languages with no morphology and missing values in NER refers to no dataset for the corresponding language.

	POS (Acc)		Morphological Tagging (F1)		NER (F1)	
	multi-udify	udapter	multi-udify	udapter	multi-udify	udapter
aii	14.6	14.8	31.5	31.5	–	–
akk	20.9	20.4	–	–	–	–
am	10.9	10.9	63.4	63.4	2.5	2.9
be	94.6	96.9	92.3	92.0	80.1	80.3
bho	61.8	63.1	68.1	66.4	–	–
bm	36.5	35.8	28.5	54.8	–	–
br	70.6	72.2	51.1	59.9	77.6	74.3
bxr	65.1	65.6	34.7	38.4	–	–
cy	70.8	69.7	42.9	43.8	65.9	73.4
fo	80.0	79.6	58.4	56.3	78.2	80.3
gsw	65.0	65.9	–	–	–	–
gun	34.5	36.3	28.8	48.2	68.1	65.5
hsb	76.8	78.8	60.7	37.4	79.0	79.7
kk	85.0	83.4	52.2	52.5	72.7	65.9
kmr	48.5	48.8	29.3	13.3	–	–
koi	45.6	48.9	33.0	22.5	–	–
kpv	36.2	36.7	27.9	22.8	–	–
krl	78.9	78.3	67.2	70.3	–	–
mdf	53.5	54.7	21.8	25.7	–	–
mr	67.5	66.5	51.7	51.2	75.9	74.1
myv	50.7	52.8	20.0	19.1	–	–
olo	74.5	76.6	59.9	62.4	–	–
pcm	62.8	54.7	–	–	–	–
sa	41.4	42.2	40.7	20.6	54.0	47.0
ta	68.6	70.3	37.8	41.7	69.8	69.0
te	81.6	84.2	–	–	62.2	62.4
tl	73.3	78.4	31.2	33.3	81.4	68.9
wbp	37.6	34.1	17.8	15.8	–	–
yo	65.1	63.7	58.7	64.1	–	–
yue	68.1	66.3	–	–	76.4	75.2
lr-avg	58.0	58.4	44.4	44.3	67.4	65.6

Table D1
Accuracy of typological feature prediction over 5-fold cross validation for high-resource languages (in-training). Majority refers to the majority class baseline. Typo-pred and typo-pred**+geo** refer to our model without and with geographical features, respectively.

	ar	en	eu	fi	he	hi	it	ja	ko	ru	sv	tr	zh	HR-AVG
majority	61.1	64.5	67.8	69.9	71.4	62.8	72.3	73.5	70.9	71.1	67.7	69.2	63.4	68.1
typo-pred	90.0	91.9	86.1	93.2	88.2	93.9	93.5	90.8	86.5	95.5	85.6	86.2	86.2	89.8
typo-pred **+geo**	90.4	93.2	86.4	95.3	88.4	92.5	94.5	91.2	86.3	95.5	87.2	84.7	86.5	90.2

Table D2
Results (accuracy) for typological feature predictions over 5-fold cross validation for low-resource languages (zero-shot). Only the languages with at least 30 available typological features (except geo features) are evaluated. Majority refers to the majority class baseline.

	am	be	bho	bm	br	bxr	cy	fo	gsw	gun	kmr	koi
majority	67.9	51.0	64.3	69.6	63.5	64.9	62.0	55.1	38.9	81.1	67.4	57.0
typo-pred	78.6	83.3	91.0	83.7	82.5	81.1	78.6	86.0	87.1	79.9	79.5	89.7
typo-pred **+geo**	78.3	88.3	84.5	80.9	84.5	80.3	78.6	84.2	93.3	77.3	78.2	89.3

	kpv	mr	myv	pcm	sa	ta	te	tl	wbp	yo	yue	LR-AVG
majority	72.0	67.8	62.1	75.8	70.7	70.5	70.1	69.5	50.3	74.3	75.2	65.3
typo-pred	86.4	79.8	78.5	88.8	80.6	78.3	80.7	78.8	66.4	77.2	81.3	81.6
typo-pred **+geo**	86.4	81.2	84.4	89.9	80.7	78.2	83.0	80.0	60.7	75.1	83.3	81.8

Table D3
Dependency parsing (LAS) results for UDapter with and without typological feature prediction. # missing shows the number of unavailable features in 289 typological features (syntax + phonology + inventory). Note that the base UDapter uses KNN-based predictions (Littell et al. 2017) for missing typological features.

	#missing features	no pred	typo pred	udapter knn	no pred **+geo**	typo pred **+geo**	udapter knn **+geo**
aii	289	1.8	3.5	14.3	9.9	11.7	13.7
akk	289	3.6	5.2	8.2	6.2	7.0	6.5
am	24	1.4	3.5	5.9	2.9	1.5	6.9
be	256	59.8	75.1	79.3	78.5	78.9	79.7
bho	229	31.0	33.6	37.3	31.6	34.4	36.7
bm	53	7.9	10.7	8.1	8.1	10.0	8.7
br	41	61.9	60.7	58.5	60.8	59.8	56.7
bxr	229	26.3	25.9	28.9	26.4	27.3	27.7
cy	208	43.9	54.3	54.4	51.5	55.0	54.8
fo	256	61.6	66.7	69.2	66.7	68.4	67.1
gsw	261	47.4	49.5	45.5	47.2	49.8	48.9
gun	119	8.0	8.8	8.4	9.1	6.7	9.6
hsb	289	42.5	48.3	54.2	53.6	53.1	53.5
kk	275	46.5	58.6	60.7	60.2	61.3	59.7
kmr	46	12.7	12.3	12.1	12.4	13.2	13.6
koi	258	18.6	23.3	23.1	23.6	24.8	23.5
kpv	59	12.6	12.1	12.5	12.8	12.1	12.4
krl	289	34.2	40.4	48.4	44.0	46.5	46.9
mdf	289	20.9	24.2	26.6	25.3	27.1	26.4
mr	65	45.2	44.2	44.4	43.0	44.7	44.9
myv	245	17.9	19.8	19.2	19.4	20.7	19.4
olo	289	31.0	36.7	43.3	40.9	40.7	43.5
pcm	131	33.8	37.0	36.7	35.1	34.8	33.9
sa	112	20.2	23.0	22.2	20.7	22.0	22.4
ta	61	44.8	44.9	46.1	42.4	47.0	44.8
te	47	70.5	71.0	71.1	69.2	69.1	68.4
tl	31	65.4	66.8	69.5	60.3	62.0	70.2
wbp	234	9.9	10.8	12.1	11.5	16.6	15.8
yo	13	42.7	42.6	42.7	41.5	40.8	42.2
yue	28	35.5	34.9	32.8	35.2	32.3	36.5
lr-avg	–	32.0	34.9	36.5	35.0	36.0	36.5

Acknowledgments

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 for all experiments.

References

Aharoni, Roee, 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. https://doi.org/10.18653/v1/N19-1388

Ammar, Waleed, 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. https://doi.org/10.1162/tacl_a_00109

Arivazhagan, Naveen, 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.

Artetxe, Mikel, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637. https://doi.org/10.18653/v1/2020.acl-main.421

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

Conneau, Alexis, 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. https://doi.org/10.18653/v1/2020.acl-main.747

de Lhoneux, Miryam, 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. https://doi.org/10.18653/v1/D18-1543

de Lhoneux, Miryam, 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. https://doi.org/10.18653/v1/K17-3022

de Marneffe, Marie Catherine, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308. https://doi.org/10.1162/coli_a_00402

Devlin, Jacob, 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.

Dong, Daxiang, 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. https://doi.org/10.3115/v1/P15-1166

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

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

Duong, Long, 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. https://doi.org/10.18653/v1/D15-1040

Edmonds, Jack. 1967. Optimum branchings. *Journal of Research of the National Bureau of*

Standards B, 71(4):233–240. https://doi.org/10.6028/jres.071B.032

Fernández-González, Daniel and Carlos Gómez-Rodríguez. 2021. Dependency parsing with bottom-up hierarchical pointer networks. *arXiv preprint arXiv:2105.09611*.

Firat, Orhan, 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. https://doi.org/10.18653/v1/N16-1101

Fisch, Adam, 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. https://doi.org/10.18653/v1/D19-1574

Gillick, Dan, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1296–1306. https://doi.org/10.18653/v1/N16-1155

Greenberg, Joseph H. 1963. Some universals of grammar with particular reference to the order of meaningful elements. In Joseph H. Greenberg, editor, *Universals of Human Language*. MIT Press, Cambridge, MA, pages 73–113.

Ha, Thanh-Le, Jan Niehues, and Alex Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the 13th International Conference on Spoken Language Translation*. International Workshop on Spoken Language Translation. https://aclanthology.org/2016.iwslt-1.6

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

Hendrycks, Dan and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with Gaussian Error Linear Units. *International Conference on Learning Representations*.

Houlsby, Neil, 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.

Howard, Jeremy 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. https://doi.org/10.18653/v1/P18-1031

Inoue, Go, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Joint prediction of morphosyntactic categories for fine-grained Arabic part-of-speech tagging exploiting tag dictionary information. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 421–431. https://doi.org/10.18653/v1/K17-1042

Johnson, Melvin, 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. https://doi.org/10.1162/tacl_a_00065

Kondratyuk, Dan 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. https://doi.org/10.18653/v1/D18-1532

Kondratyuk, Daniel, Tomáš Gavenčiak, Milan Straka, and Jan Hajič. 2018. LemmaTag: Jointly tagging and lemmatizing for morphologically rich languages with BRNNs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4921–4928. https://doi.org/10.18653/v1/D19-1279

Kulmizev, Artur, 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. https://doi.org/10.18653/v1/D19-1277

Lewis, M. Paul, Gary F. Simons, and C. D. Fennig. 2015. *Ethnologue: Languages of the World*. 18th ed. *Dallas, Texas: SIL International*.

Lin, Yu Hsiang, 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. https://doi.org/10.18653/v1/P19-1301

Littell, Patrick, 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. https://doi.org/10.18653/v1/E17-2002

Malaviya, Chaitanya, 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. https://doi.org/10.18653/v1/D17-1268

McDonald, Ryan, 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.

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

Mulcaire, Phoebe, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. 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 3912–3918. https://doi.org/10.18653/v1/N19-1392

Naseem, Tahira, 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.

Neubig, Graham 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. https://doi.org/10.18653/v1/D18-1103

Nivre, Joakim, 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 Há 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, Phuong Lê Hòng, 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, Luong Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúókun, 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.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043.

Östling, Robert 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. https://doi.org/10.18653/v1/E17-2102

Pan, Xiaoman, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), pages 1946–1958. https://doi.org/10.18653/v1/P17-1178

Peters, Matthew, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. https://doi.org/10.18653/v1/N18-1202

Pfeiffer, Jonas, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673. https://doi.org/10.18653/v1/2020.emnlp-main.617

Pires, Telmo, 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. https://doi.org/10.18653/v1/P19-1493

Platanios, Emmanouil Antonios, 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. https://doi.org/10.18653/v1/D18-1039

Ponti, Edoardo Maria, 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. https://doi.org/10.1162/coli_a_00357

Popel, Martin, 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.

Rahimi, Afshin, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164. https://doi.org/10.18653/v1/P19-1015

Rebuffi, Sylvestre Alvise, 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. https://doi.org/10.1109/CVPR.2018.00847

Scholivet, Manon, 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. https://doi.org/10.18653/v1/N19-1393

Smith, Aaron, 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. https://doi.org/10.18653/v1/K18-2011

Stickland, Asa Cooper 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.

Straka, Milan. 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.

Täckström, Oscar, 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.

Täckström, Oscar, 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.

Tiedemann, Jörg. 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.

Tran, Ke 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. https://doi.org/10.18653/v1/D19-6132

Tsai, Henry, Jason Riesa, Melvin Johnson, Naveen Arivazhagan, Xin Li, and Amelia Archer. 2019. Small and practical BERT models for sequence labeling. 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 3632–3636. https://doi.org/10.18653/v1/D19-1374

Üstün, Ahmet, 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. https://doi.org/10.18653/v1/2020.emnlp-main.180

van der Goot, Rob, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 2021. Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 176–197. https://doi.org/10.18653/v1/2021.eacl-demos.22

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

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Vilares, David, 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. https://doi.org/10.18653/v1/P16-2069

Wang, Xinyu and Kewei Tu. 2020. Second-order neural dependency parsing with message passing and end-to-end training. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 93–99.

Wu, Shijie 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. https://doi.org/10.18653/v1/D19-1077

Wu, Shijie and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130. https://doi.org/10.18653/v1/2020.repl4nlp-1.16

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, Songlin and Kewei Tu. 2021. Headed span-based projective dependency parsing. *arXiv preprint arXiv:2108.04750*.

Zeman, Daniel and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proceedings of the IJCNLP-08 Workshop on NLP for Less Privileged Languages*, pages 35–42.

Zhang, Yuan 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. https://doi.org/10.18653/v1/D15-1213