75 Languages, 1 Model: Parsing Universal Dependencies Universally

Dan Kondratyuk[1,2] and **Milan Straka**[1]
[1]Charles University, Institute of Formal and Applied Linguistics
[2]Saarland University, Department of Computational Linguistics
dankondratyuk@gmail.com, straka@ufal.mff.cuni.cz

Abstract

We present UDify, a multilingual multi-task model capable of accurately predicting universal part-of-speech, morphological features, lemmas, and dependency trees simultaneously for all 124 Universal Dependencies treebanks across 75 languages. By leveraging a multilingual BERT self-attention model pretrained on 104 languages, we found that fine-tuning it on all datasets concatenated together with simple softmax classifiers for each UD task can meet or exceed state-of-the-art UPOS, UFeats, Lemmas, (and especially) UAS, and LAS scores, without requiring any recurrent or language-specific components. We evaluate UDify for multilingual learning, showing that low-resource languages benefit the most from cross-linguistic annotations. We also evaluate for zero-shot learning, with results suggesting that multilingual training provides strong UD predictions even for languages that neither UDify nor BERT have ever been trained on. Code for UDify is available at https://github.com/hyperparticle/udify.

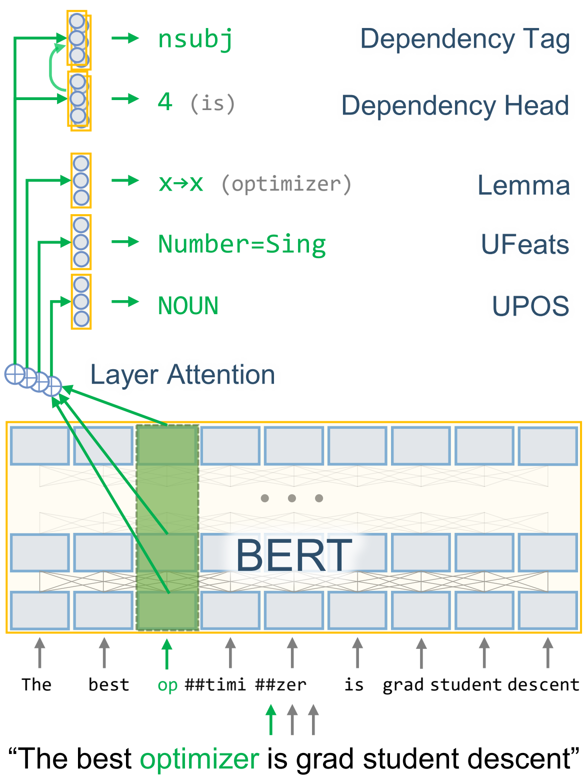


Figure 1: An illustration of the UDify network architecture with task-specific layer attention, inputting word tokens and outputting UD annotations for each token.

1 Introduction

In the absence of annotated data for a given language, it can be considerably difficult to create models that can parse the language's text accurately. Multilingual modeling presents an attractive way to circumvent this low-resource limitation. In a similar way learning a new language can enhance the proficiency of a speaker's previous languages (Abu-Rabia and Sanitsky, 2010), a model which has access to multilingual information can begin to learn generalizations across languages that would not have been possible through monolingual data alone. Works such as McDonald et al. (2011); Naseem et al. (2012); Duong et al. (2015); Ammar et al. (2016); de Lhoneux et al. (2018); Kitaev and Klein (2018); Mulcaire et al. (2019) consistently demonstrate how pairing the training data of similar languages can boost evaluation scores of models predicting syntactic information like part-of-speech and dependency trees. Multilinguality not only can improve a model's evaluation performance, but can also reduce the cost of training multiple models for a collection of languages (Johnson et al., 2017; Smith et al., 2018).

However, scaling to a higher number of languages can often be problematic. Without an ample supply of training data for the considered languages, it can be difficult to form appropriate generalizations and especially difficult if those

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2779–2795, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

languages are distant from each other. But recent techniques in language model pretraining can profit from a drastically larger supply of unsupervised text, demonstrating the capability of transferring contextual sentence-level knowledge to boost the parsing accuracy of existing NLP models (Howard and Ruder, 2018; Peters et al., 2018; Devlin et al., 2018).

One such model, BERT (Devlin et al., 2018), introduces a self-attention (Transformer) network that results in state-of-the-art parsing performance when fine-tuning its contextual embeddings. And with the release of a multilingual version pretrained on the entirety of the top 104 resourced languages of Wikipedia, BERT is remarkably capable of capturing an enormous collection of cross-lingual syntactic information. Conveniently, these languages nearly completely overlap with languages supported by the Universal Dependencies treebanks, which we will use to demonstrate the ability to scale syntactic parsing up to 75 languages and beyond.

The Universal Dependencies (UD) framework provides syntactic annotations consistent across a large collection of languages (Nivre et al., 2018; Zeman et al., 2018). This makes it an excellent candidate for analyzing syntactic knowledge transfer across multiple languages. UD offers tokenized sentences with annotations ideal for multi-task learning, including lemmas (LEMMAS), treebank-specific part-of-speech tags (XPOS), universal part-of-speech tags (UPOS), morphological features (UFEATS), and dependency edges and labels (DEPS) for each sentence.

We propose UDify, a semi-supervised multi-task self-attention model automatically producing UD annotations in any of the supported UD languages. To accomplish this, we perform the following:

1. We input all sentences into a pretrained multilingual BERT network to produce contextual embeddings, introduce task-specific layer-wise attention similar to ELMo (Peters et al., 2018), and decode each UD task simultaneously using softmax classifiers.

2. We apply a heavy amount of regularization to BERT, including input masking, increased dropout, weight freezing, discriminative fine-tuning, and layer dropout.

3. We train and fine-tune the model on the entirety of UD by concatenating *all* available training sets together.

We evaluate our model with respect to UDPipe Future, one of the winners of the CoNLL 2018 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies (Straka, 2018; Zeman et al., 2018). In addition, we analyze languages that multilingual training benefits prediction the most, and evaluate the model for zero-shot learning, i.e., treebanks which do not have a training set. Finally, we provide evidence from our experiments and other related work to help explain why pretrained self-attention networks excel in multilingual dependency parsing.

Our work uses the AllenNLP library built for the PyTorch framework. Code for UDify and a release of the fine-tuned BERT weights are available at `https://github.com/hyperparticle/udify`.

2 Multilingual Multi-Task Learning

In this section, we detail the multilingual training setup and the UDify multi-task model architecture. See Figure 1 for an architecture diagram.

2.1 Multilingual Pretraining with BERT

We leverage the provided BERT base multilingual cased pretrained model[1], with a self-attention network of 12 layers, 12 attention heads per layer, and hidden dimensions of 768 (Devlin et al., 2018). The model was trained by predicting randomly masked input words on the entirety of the top 104 languages with the largest Wikipedias. BERT uses a wordpiece tokenizer (Wu et al., 2016), which segments all text into (unnormalized) sub-word units.

2.2 Cross-Linguistic Training Issues

Table 1 displays a list of vocabulary sizes, indicating that UD treebanks possess nearly 1.6M unique tokens combined. To sidestep the problem of a ballooning vocabulary, we use BERT's wordpiece tokenizer directly for all inputs. UD expects predictions to be along word boundaries, so we take the simple approach of applying the tokenizer to each word using UD's provided segmentation. For prediction, we use the outputs of BERT corresponding to the first wordpiece per word, ignoring

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

Token	Vocab Size
Word Form	1,588,655
BERT Wordpieces	119,547
UPOS	17
XPOS	19,126
UFeats	23,974
Lemmas (tags)	109,639
Deps	251

Table 1: Vocabulary sizes of words and tags over all of UD v2.3, with a total of 12,032,309 word tokens and 668,939 sentences.

the rest[2].

In addition, the XPOS annotations are not universal across languages, or even across treebanks. Because each treebank can possess a different annotation scheme for XPOS which can slow down inference, we omit training and evaluation of XPOS from our experiments.

2.3 Multi-Task Learning with UD

For predicting UD annotations, we employ a multi-task network based on UDPipe Future (Straka, 2018), but with all embedding, encoder, and projection layers replaced with BERT. The remaining components include the prediction layers for each task detailed below, and layer attention (see Section 3.1). Then we compute softmax cross entropy loss on the output logits to train the network. For more details on reasons behind architecture choices, see Appendix A.

UPOS As is standard for neural sequence tagging, we apply a softmax layer along each word input, computing a probability distribution over the tag vocabulary to predict the annotation string.

UFeats Identical to UPOS prediction, we treat each UFeats string as a separate token in the vocabulary. We found this to produce higher evaluation accuracy than predicting each morphological feature separately. Only a small subset of the full Cartesian product of morphological features is valid, eliminating invalid combinations.

Lemmas Similar to Chrupała (2006); Müller et al. (2015), we reduce the problem of lemmatization to a sequence tagging problem by predicting a class representing an edit script, i.e., the sequence of character operations to transform the word form to the lemma. To precompute the tags, we first find the longest common substring between the form and the lemma, and then compute the shortest edit script converting the prefix and suffix of the form into the prefix and suffix of the lemma using the Wagner–Fischer algorithm (Wagner and Fischer, 1974). Upon predicting a lemma edit script, we apply the edit operations to the word form to produce the final lemma. See also Straka (2018) for more details. We chose this approach over a sequence-to-sequence architecture like Bergmanis and Goldwater (2018) or Kondratyuk et al. (2018), as this significantly reduces training efficiency.

Deps We use the graph-based biaffine attention parser developed by Dozat and Manning (2016); Dozat et al. (2017), replacing the bidirectional LSTM layers with BERT. The final embeddings are projected through arc-head and arc-dep feed-forward layers, which are combined using biaffine attention to produce a probability distribution of arc heads for each word. We then decode each tree with the Chu-Liu/Edmonds algorithm (Chu, 1965; Edmonds, 1967).

3 Fine-Tuning BERT on UD Annotations

We employ several strategies for fine-tuning BERT for UD prediction, finding that regularization is absolutely crucial for producing a high-scoring network.

3.1 Layer Attention

Empirical results suggest that when fine-tuning BERT, combining the output of the last several layers is more beneficial for the downstream tasks than just using the last layer (Devlin et al., 2018). Instead of restricting the model to any subset of layers, we devise a simple layer-wise dot-product attention where the network computes a weighted sum of all intermediate outputs of the 12 BERT layers using the same weights for each token. This is similar to how ELMo mixes the output of multiple recurrent layers (Peters et al., 2018).

More formally, let $\boldsymbol{w}_i$ be a trainable scalar for BERT embeddings $\mathbf{BERT}_{ij}$ at layer i with a token at position j, and let c be a trainable scalar. We compute contextual embeddings $\boldsymbol{e}^{(\text{task})}$ such that

$$\boldsymbol{e}_j^{(\text{task})} = c \sum_i \mathbf{BERT}_{ij} \cdot \operatorname{softmax}(\boldsymbol{w})_i \quad (1)$$

To prevent the UD classifiers from overfitting to the information in any single layer, we devise

[2]We found last, max, or average pooling of the wordpieces were not any better or worse for evaluation. Kitaev and Klein (2018) report similar results.

layer dropout, where at each training step, we set each parameter w_i to $-\infty$ with probability 0.1. This effectively redistributes probability mass to all other layers, forcing the network to incorporate the information content of all BERT layers. We compute layer attention per task, using one set of $c, \boldsymbol{w}$ parameters for each of UPOS, UFeats, Lemmas, and Deps.

3.2 Transfer Learning with ULMFiT

The ULMFiT strategy defines several useful methods for fine-tuning a network on a pretrained language model (Howard and Ruder, 2018). We apply the same methods, with a few minor modifications.

We split the network into two parameter groups, i.e., the parameters of BERT and all other parameters. We apply discriminative fine-tuning, setting the base learning rate of BERT to be $5e^{-5}$ and $1e^{-3}$ everywhere else. We also freeze the BERT parameters for the first epoch to increase training stability.

While ULMFiT recommends decaying the learning rate linearly after a linear warmup, we found that this is prone to training divergence in self-attention networks, introducing vanishing gradients and underfitting. Instead, we apply an inverse square root learning rate decay with linear warmup (Noam) seen in training Transformer networks for machine translation (Vaswani et al., 2017).

3.3 Input Masking

The authors of BERT recommend not to mask words randomly with `[MASK]` when fine-tuning the network. However, we discovered that masking often reduces the tendency of the classifiers to overfit to BERT by forcing the network to rely on the context of surrounding words. This *word dropout* strategy has been observed in other works showing improved test performance on a variety of NLP tasks (Iyyer et al., 2015; Bowman et al., 2016; Clark et al., 2018; Straka, 2018).

4 Experiments

We evaluate UDify with respect to every test set in each treebank. As there are too many results to fit within one page, we display a salient subset of scores and compare them with UDPipe Future. The full results are listed in Appendix A.

We do not directly reference metrics from other models in the CoNLL 2018 Shared Task, as the tables of results do not assume gold word segmentation and may not provide a fair comparison. Instead, we retrained the open source UDPipe Future model using gold segmentation and report results here due to its architectural similarity to UDify and its strong performance.

Note that the UDPipe Future baseline does not itself use BERT. Evaluation of BERT utilization in UDPipe Future can be found in Straka et al. (2019).

4.1 Datasets

For all experiments, we use the full Universal Dependencies v2.3 corpus available on LINDAT (Nivre et al., 2018). We omit the evaluation of datasets that do not have their training annotations freely available, i.e., Arabic NYUAD (ar_nyuad), English ESL (en_esl), French FTB (fr_ftb), Hindi English HEINCS (qhe_heincs), and Japanese BCCWJ (ja_bccwj).

To train the multilingual model, we concatenate all available training sets together, similar to McDonald et al. (2011). Before each epoch, we shuffle all sentences and feed mixed batches of sentences to the network, where each batch may contain sentences from any language or treebank, for a total of 80 epochs[3].

4.2 Hyperparameters

A summary of hyperparameters can be found in Table 6 in Appendix A.1.

4.3 Probing for Syntax

Hewitt and Manning (2019) introduce a structural probe for identifying dependency structures in contextualized word embeddings. This probe evaluates whether syntax trees (i.e., unlabeled undirected dependency trees) can be easily extracted as a global property of the embedding space using a linear transformation of the network's contextual word embeddings. The probe trains a weighted adjacency matrix on the layers of contextual embeddings produced by BERT, identifying a linear transformation where squared L2 distance between embedding vectors encodes the distance between words in the parse tree. Edges are decoded by computing the minimum spanning tree on the weight matrix (the lowest sum of edge distances).

[3] We train on a GTX 1080 Ti for approximately 25 days. See Appendix A.1 for more details

Treebank	Model	UPOS	Feats	Lem	UAS	LAS
Czech PDT (cs_pdt)	UDPipe	99.18	97.23	**99.02**	93.33	91.31
	Lang	99.18	96.87	98.72	94.35	92.41
	UDify	99.18	96.85	98.56	94.73	92.88
	UDify+Lang	**99.24**	**97.44**	98.93	**95.07**	**93.38**
German GSD (de_gsd)	UDPipe	94.48	90.68	**96.80**	85.53	81.07
	Lang	94.77	91.73	96.34	87.54	83.39
	UDify	94.55	90.65	94.82	87.81	83.59
	UDify+Lang	**95.29**	**91.94**	96.74	**88.11**	**84.13**
English EWT (en_ewt)	UDPipe	96.29	97.10	**98.25**	89.63	86.97
	Lang	**96.82**	**97.27**	97.97	**91.70**	**89.38**
	UDify	96.21	96.17	97.35	90.96	88.50
	UDify+Lang	96.57	96.96	97.90	91.55	89.06
Spanish AnCora (es_ancora)	UDPipe	**98.91**	**98.49**	**99.17**	92.34	90.26
	Lang	98.60	98.14	98.52	92.82	90.52
	UDify	98.53	97.84	98.09	92.99	90.50
	UDify+Lang	98.68	98.25	98.68	**93.35**	**91.28**
French GSD (fr_gsd)	UDPipe	97.63	**97.13**	**98.35**	90.65	88.06
	Lang	**98.05**	96.26	97.96	92.77	90.61
	UDify	97.83	96.59	97.48	**93.60**	**91.45**
	UDify+Lang	97.96	96.73	98.17	93.56	91.45
Russian SynTagRus (ru_syntagrus)	UDPipe	**99.12**	**97.57**	**98.53**	93.80	92.32
	Lang	98.90	96.58	95.16	94.40	92.72
	UDify	98.97	96.35	94.43	94.83	93.13
	UDify+Lang	99.08	97.22	96.58	**95.13**	**93.70**
Belarusian HSE (be_hse)	UDPipe	93.63	73.30	87.34	78.58	72.72
	Lang	95.88	76.12	84.52	83.94	79.02
	UDify	**97.54**	**89.36**	85.46	**91.82**	**87.19**
	UDify+Lang	97.25	85.02	**88.71**	90.67	86.98
Buryat BDT (bxr_bdt)	UDPipe	40.34	32.40	58.17	32.60	18.83
	Lang	52.54	37.03	54.64	29.63	15.82
	UDify	**61.73**	**47.86**	**61.06**	**48.43**	**26.28**
	UDify+Lang	61.73	42.79	58.20	33.06	18.65
Upper Sorbian UFAL (hsb_ufal)	UDPipe	62.93	41.10	68.68	45.58	34.54
	Lang	73.70	46.28	58.02	39.02	28.70
	UDify	84.87	48.63	**72.73**	**71.55**	**62.82**
	UDify+Lang	**87.58**	**53.19**	71.88	71.40	60.65
Kazakh KTB (kk_ktb)	UDPipe	55.84	40.40	63.96	53.30	33.38
	Lang	73.52	46.60	57.84	50.38	32.61
	UDify	**85.59**	**65.14**	**77.40**	**74.77**	**63.66**
	UDify+Lang	81.32	60.50	67.30	69.16	53.14
Lithuanian HSE (lt_hse)	UDPipe	81.70	60.47	**76.89**	51.98	42.17
	Lang	83.40	54.34	58.77	51.23	38.96
	UDify	**90.47**	**68.96**	67.83	**79.06**	**69.34**
	UDify+Lang	84.53	56.98	58.21	58.40	39.91

Table 2: Test set scores for a subset of high-resource (top) and low-resource (bottom) languages in comparison to UDPipe Future without BERT, with 3 UDify configurations: **Lang**, fine-tune on the treebank. **UDify**, fine-tune on all UD treebanks combined. **UDify+Lang**, fine-tune on the treebank using BERT weights saved from fine-tuning on all UD treebanks combined.

Model	Configuration	UPOS	Feats	Lem	UAS	LAS
UDPipe	w/o BERT	**93.76**	**91.04**	**94.63**	84.37	79.76
UDify	Task Layer Attn	93.40	88.72	90.41	**85.69**	**80.43**
UDify	Global Layer Attn	93.12	87.53	89.03	85.07	79.49
UDify	Sum Layers	93.02	87.20	88.70	84.97	79.33

Table 3: Ablation comparing the average of scores over all treebanks: task-specific layer attention (4 sets of $c, \boldsymbol{w}$ computed for the 4 UD tasks), global layer attention (one set of $c, \boldsymbol{w}$ for all tasks), and simple sum of layers ($c = 1$ and $\boldsymbol{w} = \mathbf{1}$).

Treebank		UPOS	Feats	Lem	UAS	LAS
Breton KEB	**br_keb**	63.67	46.75	53.15	63.97	40.19
Tagalog TRG	**tl_trg**	61.64	35.27	75.00	64.73	39.38
Faroese OFT	fo_oft	77.86	35.71	53.82	69.28	61.03
Naija NSC	pcm_nsc	56.59	52.75	97.52	47.13	33.43
Sanskrit UFAL	sa_ufal	40.21	18.45	37.60	41.73	19.80

Table 4: Test set results for zero-shot learning, i.e., no UD training annotations available. Languages that are pretrained with BERT are bolded.

Treebank	Model	UUAS
English EWT (en_ewt)	BERT	65.48
	BERT+finetune en_ewt	**79.67**

Table 5: UUAS test scores calculated on the predictions produced by the syntactic structural probe (Hewitt and Manning, 2019) using the English EWT treebank, on the unmodified multilingual cased BERT model and the same BERT model fine-tuned on the treebank.

We train the structural probe on unmodified and fine-tuned BERT using the default hyperparameters of Hewitt and Manning (2019) to evaluate whether the representations affected by fine-tuning BERT on dependency trees would more closely match the structure of these trees.

5 Results

We show scores of UPOS, UFeats (Feats), and Lemma (Lem) accuracies, along with unlabeled and labeled attachment scores (UAS, LAS) evaluated using the offical CoNLL 2018 Shared Task evaluation script.[4] Results for a salient subset of high-resource and low-resource languages are shown in Table 2, with a comparison between UDPipe Future and UDify fine-tuning on all languages. In addition, the table compares UDify with fine-tuning on either a single language or both languages (fine-tuning multilingually, then fine-tuning on the language with the saved multilingual weights) to provide a reference point for multilingual influences on UDify. We provide

[4]`https://universaldependencies.org/conll18/evaluation.html`

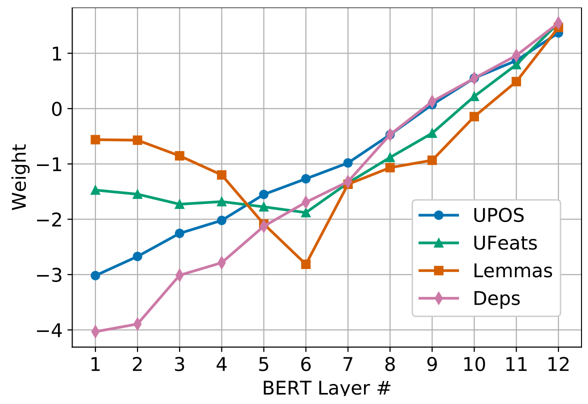


Figure 2: The unnormalized BERT layer attention weights $\boldsymbol{w}_i$ contributing to layer i for each task after training. A linear change in weight scales each BERT layer exponentially due to the softmax in Equation 1

a full table of scores for all treebanks in Appendix A.4.

A more comprehensive overview is shown in Table 3, comparing different attention strategies applied to UDify. We display an average of scores over all (89) treebanks with a training set. For zero-shot learning evaluation, Table 4 displays a subset of test set evaluations of treebanks that do not have a training set, i.e., Breton, Tagalog, Faroese, Naija, and Sanskrit. We plot the layer attention weights $\boldsymbol{w}$ after fine-tuning BERT in Figure 2, showing a set of weights per task. And Table 5 compares the unlabeled undirected attachment scores (UUAS) of dependency trees produced using a structural probe on both the unmodified multilingual cased BERT model and the extracted BERT model fine-tuned on the English EWT treebank.

6 Discussion

In this section, we discuss the most notable features of the results.

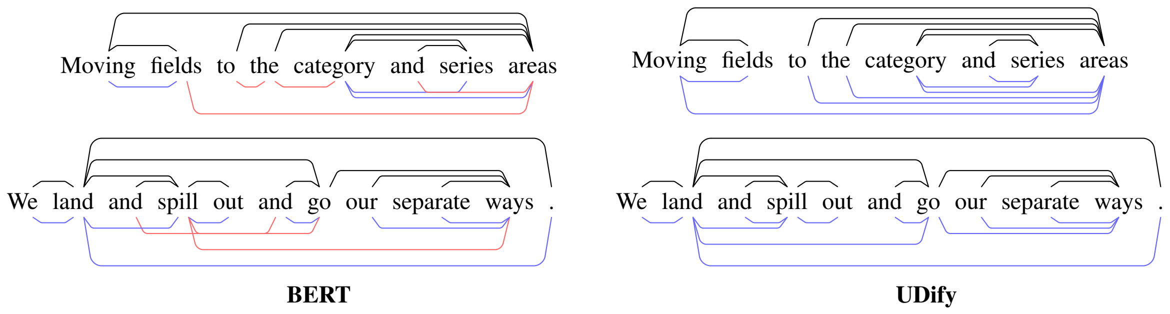
6.1 Model Performance

On average, UDify reveals a strong set of results that are comparable in performance with the state-of-the-art in parsing UD annotations. UDify excels in dependency parsing, exceeding UDPipe Future by a large margin especially for low-resource languages. UDify slightly underperforms with respect to Lemmas and Universal Features, likely due to UDPipe Future additionally using character-level embeddings (Santos and Zadrozny, 2014; Ling et al., 2015; Ballesteros et al., 2015; Kim et al., 2016), while (for simplicity) UDify does not. Additionally, UDify severely underperforms the baseline on a few low-resource languages, e.g., cop_scriptorum. We surmise that this is due to using mixed batches on an unbalanced training set, which skews the model towards predicting larger treebanks more accurately. However, we find that fine-tuning on the treebank individually with BERT weights saved from UDify eliminates most of these gaps in performance.

Echoing results seen in Smith et al. (2018), UDify also shows strong improvement leveraging multilingual data from other UD treebanks. In low-resource cases, fine-tuning BERT on all treebanks can be far superior to fine-tuning monolingually. A second round of fine-tuning on an individual treebank using UDify's BERT weights can improve this further, especially for treebanks that underperform the baseline. However, for languages that are already display strong results, we typically notice worse evaluation performance across all the evaluation metrics. This indicates that multilingual fine-tuning really is superior to single language fine-tuning with respect to these high-performing languages, showing improvements of up to 20% reduction in error.

Interestingly, Slavic languages tend to perform the best with multilingual training. While languages like Czech and Russian possess the largest UD treebanks and do not differ as much in performance from monolingual fine-tuning, evidenced by the improvements over single-language fine-tuning, we can see a large degree of morphological and syntactic structure has transferred to low-resource Slavic languages like Upper Sorbian, whose treebank contains only 646 sentences. But this is not only true of Slavic languages, as the Turkic language Kazakh (with less than 1,000 training sentences) has also improved significantly.

The zero-shot results indicate that fine-tuning on BERT can result in reasonably high scores on languages that do not have a training set. It can be seen that a combination of BERT pretraining and multilingual learning can improve predictions for Breton and Tagalog, which implies that the network has learned representations of syntax that cross lingual boundaries. Furthermore, despite the fact that neither BERT nor UDify have directly observed Faroese, Naija, or Sanskrit, we see unusually high performance in these languages. This can be partially attributed to each language closely resembling another: Faroese is very close to Ice-



Moving fields to the category and series areas
We land and spill out and go our separate ways .
BERT
UDify

Figure 3: Examples of minimum spanning trees produced by the syntactic probe are shown below each sentence, evaluated on BERT (left) and on UDify (right). Gold dependency trees are shown above each sentence in black. Matched and unmatched spanning tree edges are shown in blue and red respectively.

landic, Naija (Nigerian Pidgin) is a variant of English, and Sanskrit is an ancient Indian language related to Greek, Latin, and Hindi.

Table 3 shows that layer attention on BERT for each task is beneficial for test performance, much more than using a global weighted average. In fact, Figure 2 shows that each task prefers the layers of BERT differently, uniquely extracting the optimal information for a task. All tasks favor the information content in the last 3 layers, with a tendency to disprefer layers closer to the input. However, an interesting observation is that for Lemmas and UFeats, the classifier prefers to also incorporate the information of the first 3 layers. This meshes well with the linguistic intuition that morphological features are more closely related to the surface form of a word and rely less on context than other syntactic tasks. Curiously enough, the middle layers are highly dispreferred, meaning that the most useful processing for multilingual syntax (tagging, dependency parsing) occurs in the last 3-4 layers. The results released by Tenney et al. (2019) also agree with the intuition behind the weight distribution above, showing how the different layers of BERT generate hierarchical information like a traditional NLP pipeline, starting with low-level syntax (e.g., POS tagging) and building up to high-level syntactic and semantic dependency parsing.

6.2 Effect of Syntactic Fine-Tuning on BERT

Even without any supervised training, BERT encodes its syntax in the embedding's distance close to human-annotated dependencies. But more notably, the results in Table 5 show that fine-tuning BERT on Universal Dependencies significantly boosts UUAS scores when compared to the gold dependency trees, an error reduction of 41%. This indicates that the self-attention weights have learned a linearly-transformable representation of its vectors more closely resembling annotated dependency trees defined by linguists. Even with just unsupervised pretraining, a global structural property of the vector space of the BERT weights already produces a decent representation of the dependency tree in the squared L2 distance. Following this, it should be no surprise that training with a non-linear graph-based dependency decoder would produce even higher quality dependency trees.

6.3 Attention Visualization

We performed a high-level visual analysis of the BERT attention weights to see if they have changed on any discernible level. Our observations reveal something notable: the attention weights tend to be more sparse, and are more often sensitive to constituent boundaries like clauses and prepositions. Figure 4 illustrates this point, showing the attention weights of a particular attention head on an example sentence. We find similar behavior in 13 additional attention heads for the provided example sentence.

We see that some of the attention structure remains after fine-tuning. Previously, the attention head was mostly sensitive to previous words and punctuation. But after fine-tuning, it demonstrates more fine-grained attention towards immediate wordpieces, prepositions, articles, and adjectives. We found similar evidence in other attention heads, which implies that fine-tuning on UD produces attention that more closely resembles localized dependencies within constituents. We also find that BERT base heavily preferred to attend to punctuation, while UDify BERT does to a much lesser degree.

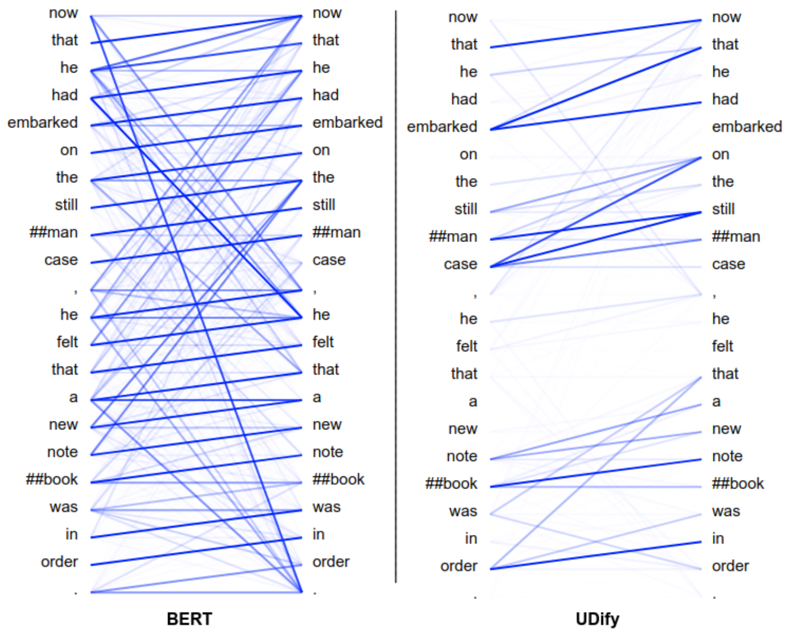


Figure 4: Visualization of BERT attention head 4 at layer 11, comparing the attended words on an English sentence between BERT base and UDify BERT after fine-tuning. The right column indicates the attended words (keys) with respect to the words in the left column (queries). Darker lines indicate stronger attention weights.

6.4 Factors that Enable BERT to Excel at Dependency Parsing and Multilinguality

Goldberg (2019) assesses the syntactic capabilities of BERT and concludes that BERT is remarkably capable of processing syntactic tasks despite not being trained on any supervised data. Conducting similar experiments, Vig (2019) and Sileo (2019) visualize the attention heads within each BERT layer, showing a number of distinct attention patterns, including attending to previous/next words, related words, punctuation, verbs/nouns, and coreference dependencies.

This neat delegation of certain low-level information processing tasks to the attention heads hints at why BERT might excel at processing syntax. We see that from the analysis on BERT fine-tuned with syntax using the syntactic probe and attention visualization, BERT produces a representation that keeps constituents close in its vector space, and improves this representation to more closely resemble human annotated dependency trees when fine-tuned on UD as seen in Figure 3. Furthermore, Ahmad et al. (2018) provide results consistent with their claim that self-attention networks can be more robust than recurrent networks to the change of word order, observing that self-attention networks capture less word order information in their architecture, which is what allows them to generally perform better at cross-lingual parsing. Wu and Dredze (2019) also analyze multilingual BERT and report that the model retains both language-independent as well as language-specific information related to each input sentence, and that the shared embedding space with the input wordpieces correlates strongly with cross-lingual generalization.

From the evidence above, we can see that the combination of strong regularization paired with the ability to capture long-range dependencies with self-attention and contextual pretraining on an enormous corpus of raw text are large contributors that enable robust multilingual modeling with respect to dependency parsing. Pretraining self-attention networks introduces a strong syntactic bias that is capable of generalizing across languages. The dependencies seen in the output dependency trees are highly correlated with the implicit dependencies learned by the self-attention, showing that self-attention is remarkably capable of modeling syntax by picking up on common syntactic patterns in text. The introduction of multilingual data also shows that these attention heads provide a surprising amount of capacity that do not degrade the performance considerably when compared to monolingual training. E.g., Devlin et al. (2018) report that the fine-tuning on the multilingual BERT model results in a small degradation in English fine-tune performance with 104 pretrained languages compared to an equivalent model pretrained only on English. This also hints that the BERT model can be compressed significantly without compromising heavily on evaluation performance.

7 Related Work

This work's main contribution in combining treebanks for multilingual UD parsing is most similar to the Uppsala system for the CoNLL 2018 Shared Task (Smith et al., 2018). Uppsala combines treebanks of one language or closely related languages together over 82 treebanks and parses all UD annotations in a multi-task pipeline architecture for a total of 34 models. This approach reduces the number of models required to parse each language while also showing results that are no worse than training on each treebank individually, and in especially low-resource cases, significantly improved. Combining UD treebanks in a language-agnostic way was first introduced in Vilares et al. (2016), which train bilingual parsers on pairs of UD treebanks, showing similar improvements.

Other efforts in training multilingual models include Johnson et al. (2017), which demonstrate a machine translation model capable of supporting translation between 12 languages. Recurrent models have also shown to be capable of scaling to a larger number of languages as seen in Artetxe and Schwenk (2018), which define a scalable approach to train massively multilingual embeddings using recurrent networks on an auxiliary task, e.g., natural language inference. Schuster et al. (2019) produce context-independent multilingual embeddings using a novel embedding alignment strategy to allow models to improve the use of cross-lingual information, showing improved results in dependency parsing.

8 Conclusion

We have proposed and evaluated UDify, a multilingual multi-task self-attention network fine-tuned on BERT pretrained embeddings, capable of producing annotations for any UD treebank, and exceeding the state-of-the-art in UD dependency parsing in a large subset of languages while being comparable in tagging and lemmatization accuracy. Strong regularization and task-specific layer attention are highly beneficial for fine-tuning, and coupled with training multilingually, also reduce the number of required models to train down to one. Multilingual learning is most beneficial for low-resource languages, even ones that do not possess a training set, and can be further improved by fine-tuning monolingually using BERT weights saved from UDify's multilingual training. All these results indicate that self-attention networks are remarkably capable of capturing syntactic patterns, and coupled with unsupervised pretraining are able to scale to a large number of languages without degrading performance.

Acknowledgments

The work described herein has been supported by OP VVV VI LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project CZ.02.1.01/0.0/0.0/16_013/0001781) and it has been supported and has been using language resources developed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

Dan Kondratyuk has been supported by the Erasmus Mundus program in Language & Communication Technologies (LCT), and by the German Federal Ministry of Education and Research (BMBF) through the project DEEPLEE (01IW17001).

References

Salim Abu-Rabia and Ekaterina Sanitsky. 2010. Advantages of bilinguals over monolinguals in learning a third language. *Bilingual Research Journal*, 33(2):173–199.

Wasi Uddin Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2018. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. *arXiv preprint arXiv:1811.00570*.

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

Mikel Artetxe and Holger Schwenk. 2018. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *arXiv preprint arXiv:1812.10464*.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Meth-*

ods in Natural Language Processing, pages 349–359.

Toms Bergmanis and Sharon Goldwater. 2018. Context sensitive neural lemmatization with lematus. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1391–1400.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *CoNLL 2016*, page 10.

Grzegorz Chrupała. 2006. Simple data-driven context-sensitive lemmatization. *Procesamiento del Lenguaje Natural*, 37.

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

Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le. 2018. Semi-supervised sequence modeling with cross-view training. *arXiv preprint arXiv:1809.08370*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 845–850.

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

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

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

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. 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)*, volume 1, pages 1681–1691.

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nikita Kitaev and Dan Klein. 2018. Multilingual constituency parsing with self-attention and pre-training. *arXiv preprint arXiv:1812.11760*.

Daniel Kondratyuk, 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.

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

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of the conference on empirical methods in natural language processing*, pages 62–72. Association for Computational Linguistics.

Phoebe Mulcaire, Jungo Kasai, and Noah Smith. 2019. Polyglot contextual representations improve crosslingual transfer. *arXiv preprint arXiv:1902.09697*.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2268–2274.

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

Joakim Nivre, Mitchell Abrams, Željko Agić, and Ahrenberg. 2018. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *arXiv preprint arXiv:1902.09492*.

Damien Sileo. 2019. Understanding bert transformer: Attention isnt all you need. *Towards Data Science*.

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

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

Milan Straka, Jana Straková, and Jan Hajič. 2019. Evaluating Contextualized Embeddings on 54 Languages in POS Tagging, Lemmatization and Dependency Parsing. *arXiv preprint arXiv:1908.07448*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*.

Ashish Vaswani, 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.

Jesse Vig. 2019. Visualizing attention in transformer-based language models. *arXiv preprint arXiv:1904.02679*.

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

Robert A Wagner and Michael J Fischer. 1974. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium. Association for Computational Linguistics.

A Appendix

In this section, we detail and explain hyperparameter choices and miscellaneous details related to model training and display the full tables of evaluation results of UDify across all UD languages.

A.1 Hyperparameters

Upon concatenating all training sets, we shuffle all the sentences, bundle them into batches of 32 sentences each, and train UDify for a total of 80 epochs before stopping. We hold the learning rate constant until we unfreeze BERT in the second epoch, where we and linearly warm up the learning rate for the next 8,000 batches and then apply inverse square root learning rate decay for the remaining epochs. For the dependency parser, we use feedforward tag and arc dimensions of 300

HYPERPARAMETER	VALUE
Dependency tag dimension	256
Dependency arc dimension	768
Optimizer	Adam
β_1, β_2	0.9, 0.99
Weight decay	0.01
Label Smoothing	0.03
Dropout	0.5
BERT dropout	0.2
Mask probability	0.2
Layer dropout	0.1
Batch size	32
Epochs	80
Base learning rate	$1e^{-3}$
BERT learning rate	$5e^{-5}$
Learning rate warmup steps	8000
Gradient clipping	5.0

Table 6: A summary of model hyperparameters.

and 800 respectively. We apply a small weight decay penalty of 0.01 to ensure that the weights remain small after each update. For optimization we use the Adam optimizer and we compute softmax cross entropy loss to train the network. We use a default β_1 value of 0.9 and lower the β_2 value from the typical 0.999 to 0.99. The reasoning is to increase the decay rate of the second moment in the Adam optimizer to reduce the chance of the optimizer being too optimistic with respect to the gradient history. We clip the gradient updates to a maximum L2 magnitude of 5.0. A summary of hyperparameters can be found in Table 6.

To speed up training, we employ bucketed batching, sorting all sentences by their length and grouping similar length sentences into each batch. However, to ensure that most sentences do not get grouped within the same batch, we fuzz the lengths of each sentence by a maximum of 10% of its true length when grouping sentences together.

Despite using all the regularization strategies shown previously, we still observe overfitting and must apply more aggressive techniques. To further regularize the network, we also increase the attention and hidden dropout rates of BERT from 0.1 to 0.2, and we also apply a dropout rate of 0.5 to all BERT layers before computing layer attention for each of the four tasks and applying a layer dropout with probability 0.1. We increase the masking probability of each wordpiece from 0.15 to 0.2.

With all these regularization strategies and hyperparameter choices combined, we are able to fine-tune BERT for far more epochs before the network starts to overfit, i.e., 80 as opposed to around 10. Even so, we believe even more regularization can improve test performance.

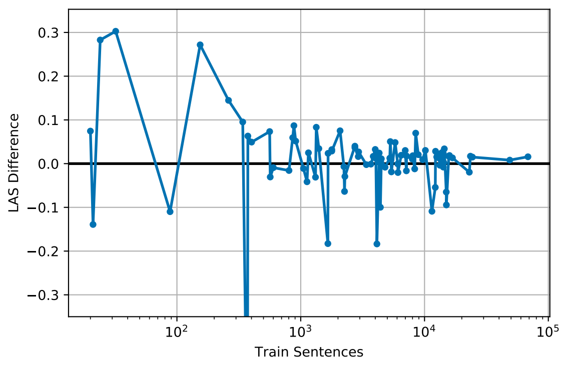


Figure 5: A plot of the difference in LAS between UDify and UDPipe Future with respect to the number of training sentences in each treebank.

The final multilingual UDify model was trained over approximately 25 days on an NVIDIA GTX 1080 Ti taking an average of 8 hours per epoch. We use half-precision (fp16) training to be able to keep the BERT model in memory. One notable aspect of training is that while we observed the model start to level out in validation performance at around epoch 30, the model continually made small, incremental improvements over each subsequent epoch, resulting in far higher scores than if the model training was terminated early. This can be partially attributed to the decaying inverse square root learning rate.

Due to the high training times, we are only able to report on a small number of training experiments for the most relevant and useful results. Prior to developing the final model, we conducted fine-tuning experiments on pairs of languages to find a set of hyperparameters that worked best for multilingual learning. After this, we gradually scaled up training to 3 languages, 5 languages, 15 languages, and then finally the model presented above. We had high doubts, and wanted to see where the limit was in multilingual training. We were pleasantly surprised to find that this simple training scheme was able to scale up so well to all UD treebanks.

A.2 Training Size Effect on Performance

To gain a better understanding of where the largest score improvements in UDify occur, we plot the LAS improvement UDify provides over UDPipe Future for each treebank, ordered by the size (number of sentences) of the training set, see Fig-

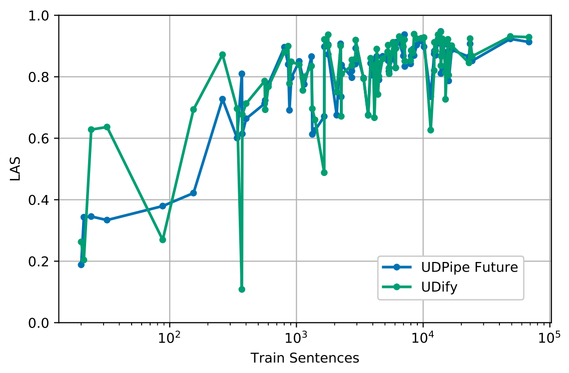


Figure 6: A plot of LAS between with respect to the number of training sentences in each treebank.

ure 5. The results show that the largest improvements tend to occur on small treebanks with less than 3,000 training examples. For absolute LAS values, see Figure 6, which indicates that more training resources tend to improve evaluation performance overall.

A.3 Miscellaneous Details

Our results show that modeling language-specific properties is not strictly necessary to achieve high-performing cross-lingual representations for dependency parsing, though we caution that the model can also likely be improved by these techniques.

Fine-tuning BERT on UD introduces a syntactic bias in the network, and we are interested in observing any differences in transfer learning by fine-tuning this new "UD-BERT" on other tasks. We leave a comprehensive evaluation of injecting syntactic bias into language models with respect to knowledge transfer for future work.

We note that saving the weights of BERT and fine-tuning a second round can improve performance as demonstrated in Stickland et al. (2019). The improvements of UDify+Lang over just UDify can be partially attributed to this, but we can see that even these improvements can be inferior to fine-tuning on all UD treebanks.

BERT limits its positional encoding to 512 wordpieces, causing some sentences in UD to be too long to fit into the model. We use a sliding window approach to break up long sentences into windows of 512 wordpieces, overlapping each window by 256 wordpieces. After feeding the windows into BERT, we select the first 256 wordpieces of each window and any remaining wordpieces in the last window to represent the contextual embeddings of each word in the original sentence.

A.4 Full Results of UD Scores

We show in Tables 7, 8, 9, and 10 UDify scores evaluated on all 124 treebanks with the official CoNLL 2018 Shared Task evaluation script. For comparison, we also include the full test evaluation of UDPipe Future on the subset of 89 treebanks with a training set. We also add a column indicating the size of each treebank, i.e., the number of sentences in the training set.

Treebank	Model	UPOS	UFeats	Lemmas	UAS	LAS	CLAS	MLAS	BLEX	Size
Afrikaans AfriBooms	UDPipe	**98.25**	**97.66**	**97.46**	**89.38**	**86.58**	**81.44**	**77.66**	**77.82**	1.3k
	UDify	97.48	96.63	95.23	86.97	83.48	77.42	70.57	70.93	1.3k
Akkadian PISANDUB	UDify	**19.92**	**99.51**	**2.32**	**27.65**	**4.54**	**3.27**	**1.04**	**0.30**	0
Amharic ATT	UDify	**15.25**	**43.95**	**58.04**	**17.38**	**3.49**	**4.88**	**0.23**	**2.53**	0
Ancient Greek PROIEL	UDPipe	**97.86**	**92.44**	**93.51**	**85.93**	**82.11**	**77.70**	**67.16**	**71.22**	15.0k
	UDify	91.20	82.29	76.16	78.91	72.66	66.07	50.79	47.27	15.0k
Ancient Greek Perseus	UDPipe	**93.27**	**91.39**	**85.02**	**78.85**	**73.54**	**67.60**	**53.87**	**53.19**	11.5k
	UDify	85.67	81.67	70.51	70.51	62.64	55.60	39.15	35.05	11.5k
Arabic PADT	UDPipe	**96.83**	**94.11**	**95.28**	87.54	**82.94**	**79.77**	**73.92**	**75.87**	6.1k
	UDify	96.58	91.77	73.55	**87.72**	82.88	79.47	70.52	50.26	6.1k
Arabic PUD	UDify	**79.98**	**40.32**	**0.00**	**76.17**	**67.07**	**65.10**	**10.67**	**0.00**	0
Armenian ArmTDP	UDPipe	93.49	**82.85**	**92.86**	78.62	71.27	65.77	**48.11**	**60.11**	561
	UDify	**94.42**	76.90	85.63	**85.63**	**78.61**	**73.72**	46.80	59.14	561
Bambara CRB	UDify	**30.86**	**57.96**	**20.42**	**30.28**	**8.60**	**6.56**	**1.04**	**0.76**	0
Basque BDT	UDPipe	**96.11**	**92.48**	**96.29**	**86.11**	**82.86**	**81.79**	**72.33**	**78.54**	5.4k
	UDify	95.45	86.80	90.53	84.94	80.97	79.52	63.60	71.56	5.4k
Belarusian HSE	UDPipe	93.63	73.30	**87.34**	78.58	72.72	69.14	46.20	58.28	261
	UDify	**97.54**	**89.36**	85.46	**91.82**	**87.19**	**85.05**	**71.54**	**68.66**	261
Breton KEB	UDify	**62.78**	**47.12**	**51.31**	**63.52**	**39.84**	**35.14**	**4.64**	**16.34**	0
Bulgarian BTB	UDPipe	**98.98**	**97.82**	**97.94**	93.38	90.35	87.01	**83.63**	**84.42**	8.9k
	UDify	98.89	96.18	93.49	**95.54**	**92.40**	**89.59**	83.43	80.44	8.9k
Buryat BDT	UDPipe	40.34	32.40	58.17	32.60	18.83	12.36	1.26	**6.49**	20
	UDify	**61.73**	**47.45**	**61.03**	**48.43**	**26.28**	**20.61**	**5.51**	11.68	20
Cantonese HK	UDify	**67.11**	**91.01**	**96.01**	**46.82**	**32.01**	**33.35**	**14.29**	**31.26**	0
Catalan AnCora	UDPipe	98.88	**98.37**	**99.07**	93.22	91.06	87.18	84.48	86.18	13.1k
	UDify	**98.89**	98.34	98.14	**94.25**	**92.33**	**89.27**	**86.21**	**86.61**	13.1k
Chinese CFL	UDify	**83.75**	**82.72**	**98.75**	**62.46**	**42.48**	**43.46**	**21.07**	**42.22**	0
Chinese GSD	UDPipe	94.88	99.22	**99.99**	84.64	80.50	76.79	71.04	76.78	4.0k
	UDify	**95.35**	**99.35**	99.97	**87.93**	**83.75**	**80.33**	**74.36**	**80.28**	4.0k
Chinese HK	UDify	**82.86**	**86.47**	**100.00**	**65.53**	**49.32**	**47.84**	**22.85**	**47.84**	0
Chinese PUD	UDify	**92.68**	**98.40**	**100.00**	**79.08**	**56.51**	**55.22**	**40.92**	**55.22**	0
Coptic Scriptorium	UDPipe	**94.70**	**96.35**	**95.49**	**85.58**	**80.97**	**72.24**	**64.45**	**68.48**	371
	UDify	27.17	52.85	55.71	27.58	10.82	6.50	0.19	1.44	371
Croatian SET	UDPipe	**98.13**	**92.25**	**97.27**	91.10	86.78	84.11	**73.61**	81.19	7.0k
	UDify	98.02	89.67	95.34	**94.08**	**89.79**	**87.70**	72.72	**82.00**	7.0k
Czech CAC	UDPipe	**99.37**	**96.34**	**98.57**	92.99	90.71	88.84	84.30	87.18	23.5k
	UDify	99.14	95.42	98.32	**94.33**	**92.41**	**91.03**	**84.68**	**89.21**	23.5k
Czech CLTT	UDPipe	98.88	91.59	98.25	86.90	84.03	80.55	71.63	79.20	861
	UDify	**99.17**	**93.66**	**98.86**	**91.69**	**89.96**	**87.59**	**79.50**	**86.79**	861
Czech FicTree	UDPipe	**98.55**	**95.87**	**98.63**	92.91	89.75	86.97	**81.04**	85.49	10.2k
	UDify	98.34	91.82	98.13	**95.19**	**92.77**	**90.99**	77.77	**88.39**	10.2k
Czech PDT	UDPipe	**99.18**	**97.23**	**99.02**	93.33	91.31	89.64	86.15	88.60	68.5k
	UDify	**99.18**	96.69	98.52	**94.73**	**92.88**	**91.64**	**87.13**	**89.95**	68.5k
Czech PUD	UDify	**97.93**	**93.98**	**96.94**	**92.59**	**87.95**	**84.85**	**77.39**	**82.81**	0
Danish DDT	UDPipe	**97.78**	**97.33**	**97.52**	86.88	84.31	81.20	**76.29**	**78.51**	4.4k
	UDify	97.50	95.41	94.60	**87.76**	**84.50**	**81.60**	73.76	75.15	4.4k
Dutch Alpino	UDPipe	96.83	96.33	**97.09**	91.37	88.38	83.51	77.28	79.82	12.3k
	UDify	**97.67**	**97.66**	95.44	**94.23**	**91.21**	**87.32**	**82.81**	**80.76**	12.3k
Dutch LassySmall	UDPipe	96.50	96.42	**97.41**	90.20	86.39	81.88	77.19	78.83	5.8k
	UDify	**96.70**	**96.57**	95.10	**94.34**	**91.22**	**88.03**	**82.06**	**81.40**	5.8k

Table 7: The full test results of UDify on 124 treebanks (part 1 of 4). The Size column indicates the number of training sentences.

Treebank	Model	UPOS	UFeats	Lemmas	UAS	LAS	CLAS	MLAS	BLEX	Size
English EWT	UDPipe	**96.29**	**97.10**	**98.25**	89.63	86.97	84.02	79.00	82.36	12.5k
	UDify	96.21	96.02	97.28	**90.96**	**88.50**	**86.25**	**79.80**	**83.39**	12.5k
English GUM	UDPipe	**96.02**	**96.82**	**96.85**	87.27	84.12	78.55	**73.51**	**74.68**	2.9k
	UDify	95.44	94.12	93.15	**89.14**	**85.73**	**83.03**	72.55	74.30	2.9k
English LinES	UDPipe	**96.91**	**96.31**	**96.45**	84.15	79.71	77.44	**71.38**	73.22	2.7k
	UDify	95.31	91.34	94.50	**87.33**	**83.71**	**82.95**	68.62	**76.23**	2.7k
English PUD	UDify	**96.18**	**93.50**	**94.20**	**91.52**	**88.66**	**87.83**	**75.61**	**80.57**	0
English ParTUT	UDPipe	96.10	**95.51**	**97.74**	90.29	87.27	82.58	**76.44**	80.33	1.8k
	UDify	**96.16**	92.61	96.45	**92.84**	**90.14**	**86.28**	74.59	**82.01**	1.8k
Erzya JR	UDify	**46.66**	**31.82**	**45.73**	**31.90**	**16.38**	**10.83**	**0.58**	**2.83**	0
Estonian EDT	UDPipe	**97.64**	**96.23**	**95.30**	88.00	85.18	83.62	78.72	**78.51**	24.4k
	UDify	97.44	95.13	86.56	**89.53**	**86.67**	**85.17**	**79.20**	69.31	24.4k
Faroese OFT	UDify	**77.46**	**35.20**	**51.09**	**67.24**	**59.26**	**51.17**	**2.39**	**21.92**	0
Finnish FTB	UDPipe	**96.65**	**96.62**	**95.49**	**90.68**	**87.89**	**85.11**	**80.58**	**81.18**	15.0k
	UDify	93.80	90.38	88.80	86.37	81.40	81.01	68.16	70.15	15.0k
Finnish PUD	UDify	**96.48**	**93.84**	**84.64**	**89.76**	**86.58**	**86.64**	**77.83**	**69.12**	0
Finnish TDT	UDPipe	**97.45**	**95.43**	**91.45**	**89.88**	**87.46**	**85.87**	**80.43**	**76.64**	12.2k
	UDify	94.43	90.48	82.89	86.42	82.03	82.62	70.89	63.66	12.2k
French GSD	UDPipe	97.63	**97.13**	**98.35**	90.65	88.06	84.35	79.76	82.39	14.5k
	UDify	**97.83**	96.17	97.34	**93.60**	**91.45**	**88.54**	**81.61**	**84.51**	14.5k
French PUD	UDify	**91.67**	**59.65**	**100.00**	**88.36**	**82.76**	**81.74**	**25.24**	**81.74**	0
French ParTUT	UDPipe	**96.93**	**94.43**	**95.70**	**92.17**	**89.63**	**84.62**	**75.22**	**78.07**	804
	UDify	96.12	88.36	93.97	90.55	88.06	83.19	63.03	74.03	804
French Sequoia	UDPipe	**98.79**	**98.09**	**98.57**	92.37	**90.73**	**87.55**	**84.51**	**85.93**	2.2k
	UDify	97.89	88.97	97.15	**92.53**	90.05	86.67	67.98	82.52	2.2k
French Spoken	UDPipe	95.91	100.00	**96.92**	82.90	77.53	71.82	68.24	69.47	1.2k
	UDify	**96.23**	**98.67**	96.59	**85.24**	**80.01**	**75.40**	**69.74**	**72.77**	1.2k
Galician CTG	UDPipe	**97.84**	**99.83**	**98.58**	**86.44**	**83.82**	**78.58**	**72.46**	**77.21**	2.3k
	UDify	96.51	97.10	97.08	84.75	80.89	74.62	65.86	72.17	2.3k
Galician TreeGal	UDPipe	**95.82**	**93.96**	**97.06**	82.72	**77.69**	71.69	**63.73**	**68.89**	601
	UDify	94.59	80.67	94.93	**84.08**	76.77	**73.06**	49.76	66.99	601
German GSD	UDPipe	94.48	**90.68**	**96.80**	85.53	81.07	76.26	58.82	72.13	13.8k
	UDify	**94.55**	90.43	94.42	**87.81**	**83.59**	**80.03**	**61.27**	**72.48**	13.8k
German PUD	UDify	**89.49**	**30.66**	**94.77**	**89.86**	**84.46**	**80.50**	**2.10**	**72.95**	0
Gothic PROIEL	UDPipe	**96.61**	**90.73**	**94.75**	85.28	**79.60**	**76.92**	**66.70**	**72.93**	3.4k
	UDify	95.55	85.97	80.57	**85.61**	79.37	76.26	63.09	58.65	3.4k
Greek GDT	UDPipe	**97.98**	**94.96**	**95.82**	92.10	89.79	85.71	**78.60**	**79.72**	1.7k
	UDify	97.72	93.29	89.43	**94.33**	**92.15**	**88.67**	77.89	71.83	1.7k
Hebrew HTB	UDPipe	**97.02**	**95.87**	**97.12**	89.70	86.86	81.45	**75.52**	**78.14**	5.2k
	UDify	96.94	93.41	94.15	**91.63**	**88.11**	**83.04**	72.55	74.87	5.2k
Hindi HDTB	UDPipe	**97.52**	**94.15**	**98.67**	94.85	**91.83**	**88.21**	**78.49**	**86.83**	13.3k
	UDify	97.12	92.59	98.23	**95.13**	91.46	87.80	75.54	86.10	13.3k
Hindi PUD	UDify	**87.54**	**22.81**	**100.00**	**71.64**	**58.42**	**53.03**	**3.32**	**53.03**	0
Hungarian Szeged	UDPipe	95.76	**91.75**	**95.05**	84.04	79.73	78.65	**67.63**	**73.63**	911
	UDify	**96.36**	86.16	90.19	**89.68**	**84.88**	**83.93**	64.27	72.21	911
Indonesian GSD	UDPipe	**93.69**	**95.58**	**99.64**	85.31	78.99	76.76	**67.74**	**76.38**	4.5k
	UDify	93.36	93.32	98.37	**86.45**	**80.10**	**78.05**	66.93	76.31	4.5k
Indonesian PUD	UDify	**76.10**	**44.23**	**100.00**	**77.47**	**56.90**	**54.88**	**7.41**	**54.88**	0
Irish IDT	UDPipe	**92.72**	**82.43**	**90.48**	**80.39**	**72.34**	**63.48**	**46.49**	**55.32**	567
	UDify	90.49	71.84	81.27	80.05	69.28	60.02	34.39	43.07	567
Italian ISDT	UDPipe	98.39	**98.11**	**98.66**	93.49	91.54	87.34	84.28	85.49	13.1k
	UDify	**98.51**	98.01	97.72	**95.54**	**93.69**	**90.40**	**86.54**	**86.70**	13.1k
Italian PUD	UDify	**94.73**	**58.16**	**96.08**	**94.18**	**91.76**	**90.05**	**25.55**	**83.74**	0
Italian ParTUT	UDPipe	**98.38**	97.77	**98.16**	92.64	90.47	85.05	81.87	82.99	1.8k
	UDify	98.21	**98.38**	97.55	**95.96**	**93.68**	**89.83**	**86.83**	**86.44**	1.8k

Table 8: The full test results of UDify on 124 treebanks (part 2 of 4).

Treebank	Model	UPOS	UFeats	Lemmas	UAS	LAS	CLAS	MLAS	BLEX	Size
Japanese GSD	UDPipe	**98.13**	**99.98**	**99.52**	**95.06**	**93.73**	**88.35**	**86.37**	**88.04**	7.1k
	UDify	97.08	99.97	98.80	94.37	92.08	86.19	82.99	85.12	7.1k
Japanese Modern	UDify	**74.94**	**96.14**	**79.70**	**74.99**	**55.62**	**42.67**	**30.89**	**35.47**	0
Japanese PUD	UDify	**97.89**	**99.98**	**99.31**	**94.89**	**93.62**	**87.92**	**84.86**	**87.15**	0
Kazakh KTB	UDPipe	55.84	40.40	63.96	53.30	33.38	27.06	**4.82**	15.10	32
	UDify	**85.59**	**65.49**	**77.18**	**74.77**	**63.66**	**61.84**	34.23	**45.51**	32
Komi Zyrian IKDP	UDify	**59.92**	**39.32**	**57.56**	**36.01**	**22.12**	**17.45**	**1.54**	**6.80**	0
Komi Zyrian Lattice	UDify	**38.57**	**29.45**	**55.33**	**28.85**	**12.99**	**10.79**	**0.72**	**3.28**	0
Korean GSD	UDPipe	**96.29**	**99.77**	**93.40**	**87.70**	**84.24**	**82.05**	**79.74**	**76.35**	4.4k
	UDify	90.56	99.63	82.84	82.74	74.26	71.72	65.94	57.58	4.4k
Korean Kaist	UDPipe	**95.59**	100.00	**94.30**	**88.42**	**86.48**	**84.12**	**80.72**	**79.22**	23.0k
	UDify	94.67	**99.98**	85.89	87.57	84.52	82.05	78.27	68.99	23.0k
Korean PUD	UDify	**64.43**	**60.47**	**70.47**	**63.57**	**46.89**	**45.29**	**16.26**	**30.94**	0
Kurmanji MG	UDPipe	53.36	**41.54**	**69.58**	**45.23**	**34.32**	**29.41**	**2.74**	19.39	21
	UDify	**60.23**	37.78	58.08	35.86	20.40	14.75	1.42	**7.28**	21
Latin ITTB	UDPipe	98.34	**96.97**	**98.99**	91.06	88.80	86.40	**82.35**	85.71	16.8k
	UDify	**98.48**	95.81	98.08	**92.43**	**90.12**	**87.93**	82.24	**85.97**	16.8k
Latin PROIEL	UDPipe	**97.01**	**91.53**	**96.32**	83.34	78.66	76.20	**67.40**	**73.65**	15.9k
	UDify	96.79	89.49	91.79	**84.85**	**80.52**	**77.96**	67.18	71.00	15.9k
Latin Perseus	UDPipe	88.40	79.10	**81.45**	71.20	61.28	56.32	41.58	45.09	1.3k
	UDify	**90.96**	**82.09**	81.08	**78.33**	**69.60**	**65.95**	**50.26**	**51.33**	1.3k
Latvian LVTB	UDPipe	**96.11**	**93.01**	**95.46**	87.20	83.35	80.90	**71.92**	**76.64**	7.2k
	UDify	96.02	89.78	91.00	**89.33**	**85.09**	**82.34**	69.51	72.58	7.2k
Lithuanian HSE	UDPipe	81.70	60.47	**76.89**	51.98	42.17	38.93	18.17	28.70	154
	UDify	**90.47**	**70.00**	67.17	**79.06**	**69.34**	**66.00**	**36.21**	**36.35**	154
Maltese MUDT	UDPipe	**95.99**	100.00	**100.00**	**84.65**	**79.71**	**71.49**	**66.75**	**71.49**	1.1k
	UDify	91.98	**99.89**	**100.00**	83.07	75.56	65.08	58.14	65.08	1.1k
Marathi UFAL	UDPipe	80.10	**67.23**	**81.31**	70.63	61.41	57.44	**29.34**	**45.87**	374
	UDify	**88.59**	59.22	72.82	**79.37**	**67.72**	**60.13**	21.71	39.25	374
Naija NSC	UDify	**55.44**	**51.32**	**97.03**	**45.75**	**32.16**	**31.62**	**4.73**	**29.33**	0
North Sami Giella	UDPipe	**92.54**	**90.03**	**88.31**	**78.30**	**73.49**	**70.94**	**62.40**	**61.45**	2.3k
	UDify	90.21	83.55	71.50	74.30	67.13	64.41	51.20	40.63	2.3k
Norwegian Bokmaal	UDPipe	**98.31**	**97.14**	**98.64**	92.39	90.49	88.18	84.06	86.53	15.7k
	UDify	98.18	96.36	97.33	**93.97**	**92.18**	**90.40**	**85.02**	**87.13**	15.7k
Norwegian Nynorsk	UDPipe	**98.14**	**97.02**	**98.18**	92.09	90.01	87.68	82.97	85.47	14.2k
	UDify	**98.14**	96.55	97.18	**94.34**	**92.37**	**90.39**	**85.01**	**86.71**	14.2k
Norwegian NynorskLIA	UDPipe	89.59	86.13	93.93	68.08	60.07	54.89	44.47	50.98	340
	UDify	**95.01**	**93.36**	**96.13**	**75.40**	**69.60**	**65.33**	**56.90**	**62.27**	340
Old Church Slavonic PROIEL	UDPipe	**96.91**	**90.66**	**93.11**	**89.66**	**85.04**	**83.41**	**73.63**	**77.81**	4.1k
	UDify	84.23	71.30	65.70	76.71	66.67	64.10	46.25	43.88	4.1k
Old French SRCMF	UDPipe	**96.09**	**97.81**	**100.00**	**91.74**	**86.83**	**83.85**	**79.91**	**83.85**	13.9k
	UDify	95.73	96.98	**100.00**	**91.74**	86.65	83.49	78.85	83.49	13.9k
Persian Seraji	UDPipe	**97.75**	**97.78**	**97.44**	**90.05**	**86.66**	**83.26**	**81.23**	**80.93**	4.8k
	UDify	96.22	94.73	92.55	89.59	85.84	81.98	76.65	74.74	4.8k
Polish LFG	UDPipe	**98.80**	**95.49**	**97.54**	96.58	**94.76**	93.01	**87.04**	**90.26**	13.8k
	UDify	**98.80**	87.71	94.04	**96.67**	94.58	**93.03**	76.50	85.15	13.8k
Polish SZ	UDPipe	98.34	**93.04**	**97.16**	93.39	**91.24**	**89.39**	**81.06**	**85.99**	6.1k
	UDify	**98.36**	67.11	93.92	**93.67**	89.20	87.31	48.47	80.24	6.1k
Portuguese Bosque	UDPipe	97.07	**96.40**	**98.46**	91.36	**89.04**	**85.19**	**76.67**	**83.06**	8.3k
	UDify	**97.10**	89.70	91.60	**91.37**	87.84	84.13	69.09	78.64	8.3k
Portuguese GSD	UDPipe	**98.31**	**99.92**	**99.30**	93.01	91.63	87.67	**85.96**	86.94	9.7k
	UDify	98.04	95.75	98.95	**94.22**	**92.54**	**89.37**	82.32	**87.90**	9.7k
Portuguese PUD	UDify	**90.14**	**51.16**	**99.79**	**87.02**	**80.17**	**74.10**	**17.51**	**74.10**	0
Romanian Nonstandard	UDPipe	96.68	**90.88**	**94.78**	89.12	84.20	78.91	**65.93**	**73.44**	8.0k
	UDify	**96.83**	88.89	89.33	**90.36**	**85.26**	**80.41**	64.68	68.11	8.0k

Table 9: The full test results of UDify on 124 treebanks (part 3 of 4).

Treebank	Model	UPOS	UFeats	Lemmas	UAS	LAS	CLAS	MLAS	BLEX	Size
Romanian RRT	UDPipe	**97.96**	**97.53**	**98.41**	91.31	86.74	82.57	79.02	**81.09**	8.0k
	UDify	97.73	96.12	95.84	**93.16**	**88.56**	**84.87**	**79.20**	79.92	8.0k
Russian GSD	UDPipe	**97.10**	**92.66**	**97.37**	88.15	84.37	82.66	**74.07**	**80.03**	3.9k
	UDify	96.91	87.45	77.73	**90.71**	**86.03**	**84.51**	67.24	62.08	3.9k
Russian PUD	UDify	**93.06**	**63.60**	**77.93**	**93.51**	**87.14**	**83.96**	**37.25**	**61.86**	0
Russian SynTagRus	UDPipe	**99.12**	**97.57**	**98.53**	93.80	92.32	90.85	**87.91**	**89.17**	48.8k
	UDify	98.97	96.29	94.47	**94.83**	**93.13**	**91.87**	86.91	85.44	48.8k
Russian Taiga	UDPipe	93.18	82.87	89.99	75.45	69.11	65.31	48.81	57.21	881
	UDify	**95.39**	**88.47**	**90.19**	**84.02**	**77.80**	**75.12**	**59.71**	**65.15**	881
Sanskrit UFAL	UDify	**37.33**	**17.63**	**37.38**	**40.21**	**18.56**	**15.38**	**0.85**	**4.12**	0
Serbian SET	UDPipe	**98.33**	**94.35**	**97.36**	92.70	89.27	87.08	**79.14**	84.18	2.9k
	UDify	98.30	92.22	95.86	**95.68**	**91.95**	**90.30**	78.45	**84.93**	2.9k
Slovak SNK	UDPipe	96.83	**90.82**	**96.40**	89.82	86.90	84.81	74.00	81.37	8.5k
	UDify	**97.46**	89.30	93.80	**95.92**	**93.87**	**92.86**	**77.33**	**85.12**	8.5k
Slovenian SSJ	UDPipe	98.61	**95.92**	**98.25**	92.96	91.16	88.76	**83.85**	**86.89**	6.5k
	UDify	**98.73**	93.44	96.50	**94.74**	**93.07**	**90.94**	81.55	86.38	6.5k
Slovenian SST	UDPipe	93.79	86.28	**95.17**	73.51	67.51	63.46	52.67	60.32	2.1k
	UDify	**95.40**	**89.81**	95.15	**80.37**	**75.03**	**71.19**	**61.32**	**67.24**	2.1k
Spanish AnCora	UDPipe	**98.91**	**98.49**	**99.17**	92.34	90.26	86.39	**83.97**	**85.51**	14.3k
	UDify	98.53	97.89	98.07	**92.99**	**90.50**	**87.26**	83.43	84.85	14.3k
Spanish GSD	UDPipe	**96.85**	**97.09**	**98.97**	90.71	**88.03**	**82.85**	**75.98**	**81.47**	14.2k
	UDify	95.91	95.08	96.52	**90.82**	87.23	82.83	72.47	78.08	14.2k
Spanish PUD	UDify	**88.98**	**54.58**	**100.00**	**90.45**	**83.08**	**77.42**	**18.06**	**77.42**	0
Swedish LinES	UDPipe	96.78	**89.43**	**97.03**	86.07	81.86	80.32	66.48	77.38	2.7k
	UDify	**96.85**	87.24	92.70	**88.77**	**85.49**	**85.61**	**66.99**	**77.62**	2.7k
Swedish PUD	UDify	**96.36**	**80.04**	**88.81**	**89.17**	**86.10**	**85.25**	**57.12**	**72.92**	0
Swedish Sign Language SSLC	UDPipe	**68.09**	100.00	**100.00**	**50.35**	**37.94**	**39.51**	**30.96**	**39.51**	88
	UDify	63.48	**96.10**	**100.00**	40.43	26.95	30.12	23.29	30.12	88
Swedish Talbanken	UDPipe	97.94	**96.86**	**98.01**	89.63	86.61	84.45	79.67	**82.26**	4.3k
	UDify	**98.11**	95.92	95.50	**91.91**	**89.03**	**87.26**	**80.72**	81.31	4.3k
Tagalog TRG	UDify	**60.62**	**35.62**	**73.63**	**64.04**	**40.07**	**36.84**	**0.00**	**13.16**	0
Tamil TTB	UDPipe	91.05	**87.28**	**93.92**	74.11	66.37	63.71	**55.31**	**59.58**	401
	UDify	**91.50**	83.21	80.84	**79.34**	**71.29**	**69.10**	53.62	54.84	401
Telugu MTG	UDPipe	93.07	99.03	**100.00**	91.26	**85.02**	**81.76**	**77.75**	**81.76**	1.1k
	UDify	**93.48**	**99.31**	**100.00**	**92.23**	83.91	79.92	76.10	79.92	1.1k
Thai PUD	UDify	**56.78**	**62.48**	**100.00**	**49.05**	**26.06**	**18.42**	**3.77**	**18.42**	0
Turkish IMST	UDPipe	**96.01**	**92.55**	**96.01**	74.19	**67.56**	63.83	**56.96**	**61.37**	3.7k
	UDify	94.29	84.49	87.71	**74.56**	67.44	**63.87**	49.42	54.10	3.7k
Turkish PUD	UDify	**77.34**	**24.59**	**84.31**	**67.68**	**46.07**	**39.95**	**2.61**	**32.50**	0
Ukrainian IU	UDPipe	97.59	**92.66**	**97.23**	88.29	85.25	81.90	**73.81**	79.10	5.3k
	UDify	**97.71**	88.63	94.00	**92.83**	**90.30**	**88.15**	72.93	**81.04**	5.3k
Upper Sorbian UFAL	UDPipe	62.93	41.10	68.68	45.58	34.54	27.18	**3.37**	16.65	24
	UDify	**84.87**	**48.84**	**72.68**	**71.55**	**62.82**	**56.04**	16.19	**37.89**	24
Urdu UDTB	UDPipe	93.66	81.92	**97.40**	87.50	81.62	75.20	55.02	73.07	4.0k
	UDify	**94.37**	**82.80**	96.68	**88.43**	**82.84**	**77.00**	**56.70**	**73.97**	4.0k
Uyghur UDT	UDPipe	**89.87**	**88.30**	**95.31**	**78.46**	**67.09**	**60.85**	**47.84**	**57.08**	1.7k
	UDify	75.88	70.80	79.70	65.89	48.80	38.95	21.75	31.31	1.7k
Vietnamese VTB	UDPipe	89.68	**99.72**	**99.55**	70.38	62.56	60.03	55.56	59.54	1.4k
	UDify	**91.29**	99.58	99.21	**74.11**	**66.00**	**63.34**	**58.71**	**62.61**	1.4k
Warlpiri UFAL	UDify	**33.44**	**18.15**	**39.17**	**21.66**	**7.96**	**7.49**	**0.00**	**0.88**	0
Yoruba YTB	UDify	**50.86**	**78.32**	**85.56**	**37.62**	**19.09**	**16.56**	**6.30**	**12.15**	0

Table 10: The full test results of UDify on 124 treebanks (part 4 of 4).