RobertNLP at the IWPT 2021 shared task: simple enhanced UD parsing for 17 languages

Stefan Grünewald, Frederik Tobias Oertel, Annemarie Friedrich

Angaben zur Veröffentlichung / Publication details:

Grünewald, Stefan, Frederik Tobias Oertel, and Annemarie Friedrich. 2021. “RobertNLP at the IWPT 2021 shared task: simple enhanced UD parsing for 17 languages.” In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021), August 6, 2021, Bangkok, Thailand (online)*, edited by Stephan Oepen, Kenji Sagae, Reut Tsarfaty, Gosse Bouma, Djamé Seddah, and Daniel Zeman, 196–203. Stroudsburg, PA: Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.iwpt-1.21.

RobertNLP at the IWPT 2021 Shared Task: Simple Enhanced UD Parsing for 17 Languages

Stefan Grünewald[★][♦] Frederik Oertel[♦] Annemarie Friedrich[♦]
[★]Institut für Maschinelle Sprachverarbeitung, University of Stuttgart
[♦]Bosch Center for Artificial Intelligence, Renningen, Germany
stefan.gruenewald|frederiktobias.oertel|annemarie.friedrich
@de.bosch.com

Abstract

This paper presents our multilingual dependency parsing system as used in the *IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies*. Our system consists of an unfactorized biaffine classifier that operates directly on fine-tuned XLM-R embeddings and generates enhanced UD graphs by predicting the best dependency label (or absence of a dependency) for each pair of tokens. To avoid sparsity issues resulting from lexicalized dependency labels, we replace lexical items in relations with placeholders at training and prediction time, later retrieving them from the parse via a hybrid rule-based/machine-learning system. In addition, we utilize model ensembling at prediction time. Our system achieves high parsing accuracy on the blind test data, ranking 3$^{\text{rd}}$ out of 9 with an average ELAS F1 score of 86.97.

1 Introduction

Enhanced Universal Dependencies (Schuster and Manning, 2016) are an extension of the widely used Universal Dependencies (UD) framework for syntactic dependency annotation (de Marneffe et al., 2014). To better model linguistic phenomena such as coordination, raising/control, and relative clauses, enhanced UD extends basic UD trees by including additional dependencies between tokens in order to make relations between content words more explicit. While there is evidence for the utility of enhanced dependencies in downstream applications (Schuster et al., 2017), adding them means that dependency structures are not constrained to trees any more, which makes parsing them a different problem with its own set of challenges.

In the past, research on enhanced UD parsing has mostly focused on rule-based methods for extracting enhanced graphs from existing basic trees (Nyblom et al., 2013; Simi and Montemagni, 2018; Nivre et al., 2018). Furthermore, only a relatively small number of UD treebanks is annotated with enhanced dependencies. Recently, however, interest in enhanced UD has increased, most notably with the IWPT 2020 Shared Task (Bouma et al., 2020), which asked contestants to produce enhanced UD graphs from raw text for 17 languages.

Submission	Score
1. TGIF	**89.24**
2. ShanghaiTech	87.07
3. RobertNLP	86.97
Median	83.64

Table 1: Overview of IWPT 2021 results (avg. ELAS F1 score). Full results can be found at https://universaldependencies.org/iwpt21/results.html.

For our submission to the 2021 edition of the Shared Task (Bouma et al., 2021), we adapt our English-only submission from last year (Grünewald and Friedrich, 2020) to all 17 languages that are part of the competition. The core principles of our system remain the same:

- We do not rely on basic dependencies for creating enhanced graphs. Instead, we directly parse from raw tokens into enhanced UD graphs.
- We use an unfactorized biaffine classifier architecture which predicts the most likely dependency label (or absence of a dependency) for each pair of tokens in the sentence, forming a dependency graph from the union of these predictions.
- Inputs to the biaffine classifier are extracted directly from a fine-tuned transformer-based language model.

Proceedings of the 17th International Conference on Parsing Technologies (IWPT 2021), pages 196–203
Bangkok, Thailand (online), August 6, 2021. ©2021 Association for Computational Linguistics

Instead of a strictly rule-based system as used by Grünewald and Friedrich (2020), we use a hybrid rule-based/machine-learning system to retrieve lexical material for dependency labels at prediction time (see Sec. 2.5). In order to further increase our parser's accuracy as well as its robustness across treebanks, we use model ensembling.

As shown in Table 1, our system achieves high parsing accuracy, ranking 3rd out of 9 with an average ELAS score of 86.97.

2 Our Model

This section describes the components of our parser as submitted to the Shared Task.

2.1 Pre-processing

For tokenization and sentence segmentation, we employ Trankit$_{\text{large}}$ (Nguyen et al., 2021), which achieves state-of-the-art (or near state-of-the-art) results for these tasks on the languages present in the Shared Task. We use the default model for each language.

2.2 Input Token Representation

We use the transformer-based, multilingual XLM-R$_{\text{large}}$ language model (Conneau et al., 2020) to generate contextualized word embeddings for the tokens of the input sentence, fine-tuning the model while training our parser. We create the wordpiece-tokenized input for XLM-R by feeding each token into the XLM-R tokenizer. In addition, we prepend a special *[root]* token to each sentence, which serves as an artificial head of the *root* relation that must be present in every sentence. This token receives a fixed, learned embedding instead of a contextualized XLM-R embedding, but with the same number of dimensions.

The final embedding $\mathbf{r}_i$ for a token at position i is extracted by forming a weighted sum of the internal XLM-R layers at the position corresponding to the first wordpiece of the original token. Following Kondratyuk and Straka (2019), coefficients for this weighted sum are learned during training, while randomly dropping layers to prevent the model from focusing on only a single layer.

2.3 Dependency Classification

Figure 1 shows an overview of our neural-network based dependency classifier, which predicts relation labels (or absence of a relation) between pairs of tokens.

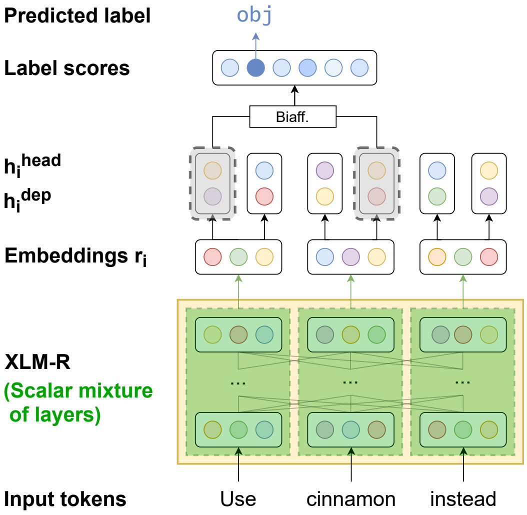

Figure 1: Architecture of neural network predicting dependency relations between pairs of tokens.

Classifier architecture. Our dependency classifier follows the architecture proposed by Dozat and Manning (2018), which is capable of producing general (bi-lexical) dependency graph structures. The approach works by creating, for each input token embedding $\mathbf{r}_i$, a head representation $\mathbf{h}_i^{head}$ and a dependent representation $\mathbf{h}_i^{dep}$ via two single-layer feedforward neural networks:

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

$$\mathbf{h}_i^{dep} = \text{FNN}^{dep}(\mathbf{r}_i) \quad (2)$$

For each ordered pair (i, j) of tokens in the sentence, their respective head and dependent representations are then fed to a biaffine classifier (Eq. 3, Dozat and Manning, 2017), which outputs logits $\mathbf{s}_{i,j}$ over the possible dependency labels.[1]

We encode the absence of a dependency relation between two tokens as simply another label ($\varnothing$). This "unfactorized" approach is in contrast to a "factorized" approach that first predicts presence or absence of relations and then uses a second classifier to predict labels. Dozat and Manning (2018) found that the unfactorized approach performed on par with the factorized approach for *semantic* dependency parsing, and this finding has been shown to also apply to enhanced UD parsing (Grünewald et al., 2021).

Finally, we can extract a probability distribution

[1] Note that this means that each pair of tokens is fed to the classifier twice as an ordered pair, once with i as the potential head and j as the potential dependent, and once the other way around.

$P(y_{i,j})$ over dependency labels from the logits:

$$\text{Biaff}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{U} \mathbf{x}_2 + W(\mathbf{x}_1 \oplus \mathbf{x}_2) + \mathbf{b} \quad (3)$$

$$\mathbf{s}_{i,j} = \text{Biaff}\big(\mathbf{h}_i^{head}, \mathbf{h}_j^{dep}\big) \quad (4)$$

$$P(y_{i,j}) = \text{softmax}(\mathbf{s}_{i,j}) \quad (5)$$

$\mathbf{U}$, W and $\mathbf{b}$ in (3) are learned parameters; $\oplus$ denotes the concatenation operation. The model is trained to minimize cross entropy loss w. r. t. the true dependency label between each pair of tokens.

De-lexicalizing dependency labels. Because enhanced UD adds lexical information to certain dependencies (e.g., *obl:instead_of*), the number of possible dependency labels is very large for most treebanks, with up to over 1100 for Arabic-PADT. Among the languages being part of the Shared Task, French is an exception as its treebanks do not make use of lexicalized labels. To avoid sparsity issues, we strip lexical information from labels during training, instead replacing them with placeholders (e.g., *obl:[case]*) indicating where in the dependency graph the lexical information is expected to be found (see Sec. 2.5 for a detailed description of the reconstruction process). This way, we can remove all lexicalized relations from the label vocabulary, instead adding only a much smaller number of placeholder labels. The basic relation types affected by this process are *nmod*, *obl*, *acl*, *advcl*, and *conj*. We keep all other, non-lexicalized subtype labels, including those that occur together with lexical material (e.g., *obl:järgi:gen* becomes *obl:[case]:gen*). Our procedure reduces label counts substantially, e.g., to 59 for Arabic.

2.4 Assembling the Dependency Graph

The outputs $P(y_{i,j})$ provided by the dependency classifier can be regarded as a 3-dimensional tensor, with one dimension corresponding to the tokens as heads, one dimension corresponding to the tokens as dependents, and the third dimension corresponding to the label set. Figure 2 gives a two-dimensional view of this tensor, with each cell containing the highest-scoring label for a head (row label) and dependent (column label) pair.

Ensembling. Instead of continuing directly with the predicted matrices as described above, we train multiple models with different initializations for each language and then ensemble them by averaging their output probabilities during prediction. In other words, we compute the probabilities of labels $P(y_{i,j})$ between two ordered tokens i and j as

$$P(y_{i,j}) = \frac{1}{m} \sum_{k=1}^{m} \text{softmax}(\mathbf{s}_{i,j}^{(k)}) \quad (6)$$

where m is the number of models and $\mathbf{s}_{i,j}^{(k)}$ is the unnormalized output vector of the k-th model for the token pair (i, j).

For languages for which more than one training treebank is available, we ensemble models trained on different treebanks. For more details on this procedure, see section Sec. 3.1.

Ensuring graph structure constraints. Using the output tensors created via ensembling, we can assemble a dependency graph by retrieving the highest-scoring dependency (or $\varnothing$, i.e., no relation) for each pair of tokens in the sentence and forming their union (omitting the diagonal as enhanced UD does not allow links starting and ending at the same node). Although enhanced UD eliminates the requirement that dependency graphs must be trees, it maintains the structural constraint that every token must be reachable from the root of the graph.[2] Although our system learns to produce graphs that obey this constraint in the vast majority of cases, there are cases where structurally invalid graphs are retrieved. To make these graphs structurally valid, we perform the following heuristic post-processing steps:

1. If the graph has more than one root, we remove all but the most confidently predicted *root* dependency.
2. If there are one or more nodes in the graph that are not reachable from the root, we select the most confidently predicted non-$\varnothing$ edge from a reachable to an unreachable node and add it to the graph. We repeat this step until every node is reachable from the root.

Removal of superfluous dependencies. UD contains several relations that empirically only appear on their own, i.e., whose dependent may have only one incoming edge of this type. These relations are *fixed*, *flat*, *goeswith*, *punct*, and *cc*. If our parser erroneously predicts several of these relations for a single token (e.g., punctuation being

[2] Graphs in enhanced UD may have more than one root, but empirically, the vast majority have only one root. Therefore, we assume exactly one root for each dependency graph for simplicity.

	[root]	Use	cinnamon	instead	of	sugar	or	sweetener
[root]	∅	*root*	∅	∅	∅	∅	∅	∅
Use	∅	∅	*obj*	∅	∅	*obl:[case]*	∅	*obl:[case]*
cinnamon	∅	∅	∅	∅	∅	∅	∅	∅
instead	∅	∅	∅	∅	*fixed*	∅	∅	∅
of	∅	∅	∅	∅	∅	∅	∅	∅
sugar	∅	∅	∅	*case*	∅	∅	∅	*conj:[cc]*
or	∅	∅	∅	∅	∅	∅	∅	∅
sweetener	∅	∅	∅	∅	∅	∅	*cc*	∅

Figure 2: Prediction matrix of the dependency classifier. Cell entries show the highest-scoring label for each ordered pair of tokens, with row/column labels indicating potential heads/dependents respectively.

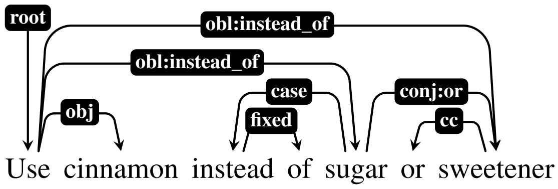

Figure 3: Dependency graph with lexicalized labels.

attached to several tokens at once), we remove all but the most confidently predicted dependency.

2.5 Label Lexicalization

As outlined in Sec. 2.3, lexical information is stripped from dependency labels during training, using the format *base:[placeholder]*. At prediction time, we re-lexicalize predicted placeholder labels using a two-step procedure. First, lexical material is retrieved from the dependency graph using a rule-based heuristic, and then a machine-learning classifier is run on the output to correct potential errors.

Re-lexicalization heuristic. The main rule of our re-lexicalization heuristic checks if the token has a dependent that is attached via the placeholder of the de-lexicalized relation in question. If so, we lexicalize the relation with the token of this dependent. For example, in Figure 3, our parser predicts *obl:[case]* and we hence re-lexicalize this relation with the token(s) of the *case* dependents of "sugar." (Multiword expressions, such as "instead of", are handled by concatenating word forms linked by the *fixed* relation.) In addition, there is a number of more fine-grained rules to handle lexicalization in the context of specific constructions such as coordination. More details are reported by Grünewald and Friedrich (2020).

Treebank	Heuristic	Hybrid
Arabic-PADT	93.4	97.5
Czech-PDT	90.9	99.2
English-EWT	98.4	98.8
Estonian-EDT	98.8	99.8
Latvian-LVTB	99.4	99.7
Polish-PDB	91.8	98.9
Slovak-SNK	93.0	98.0
Tamil-TTB	16.1	66.1

Table 2: Re-lexicalization accuracy (%) on a selection of gold development treebanks.

As can be seen in Table 2, the rule-based heuristic achieves good results in the case of English – the language that it was initially designed for – and for a number of other languages (e.g. Estonian and Latvian), with re-lexicalization accuracies greater than 98 % when evaluating on the gold development data. However, it performs markedly worse for some of the other languages in the Shared Task, such as Arabic, Czech and (especially) Tamil. The main reason for this is that the heuristic can only retrieve word forms directly from the raw sentence, whereas the lexical material in gold dependency labels is lemmatized.

Rule-based label transducer. To increase re-lexicalization accuracy, we perform a second step after running the heuristic on the initial parser output, automatically learning a "label transducer" for each language from treebank data. For each language, we train a RandomForest classifier (Breiman, 2001) that takes as input the lexicalized labels predicted by our heuristic as well as a representation of its sentential context. The label transducer then predicts a new label, which may differ from the initial prediction made by the heuristic. In other words, the label transducer functions as an ML-based error correction mechanism.

The input features for the classifier are (a) a one-hot encoding of the lexicalized label predicted by the heuristic, e.g., *conj:als*; and (b) a binary encoding of all tokens in the graph that are at most 1 dependency edge away from the endpoint of the relation in question. The output space is the set of lexicalized dependency labels as present in the gold training data.

In few cases, the label transducer predicts a different base relation type compared to the one given as input, i.e., it may transform an input of *conj:als* into an output of *nmod:in*. As we observed that

XLM-R embeddings	
Embeddings dimension	1024
Token mask probability	0.15
Layer dropout	0.1
Hidden dropout	0.2
Attention dropout	0.2
Output dropout	0.5
Biaffine classifier	
Hidden size	1024
Dropout	0.33
AdamW Optimizer	
Batch size	32
LR schedule	Noam
Warmup steps	1 epoch
Peak learning rate	$4e^{-5}$
β_1, β_2	0.9, 0.999
Weight decay	0.0

Table 3: Basic hyperparameter values used in training.

such predictions are almost always incorrect, we keep the heuristic's output in these cases.

On the gold development data, we find that including the ML-based transducer in the relexicalization process leads to moderate to large accuracy increases (see Table 2). This is the case particularly for languages where lexical material in dependencies often differs from the raw tokens in the graph (e.g., Arabic and Tamil).

3 Experiments

This section describes our main submission, as well as a number of additional experiments.

3.1 Experimental Settings

We use the provided training and development data for training and validation, respectively. During training, we use gold-segmented sentences and gold tokenization.

For hyperparameter settings, we mostly stick with the values of Grünewald and Friedrich (2020). The exceptions are parameters related to the training process itself, where we use a batch size of 32 in conjunction with an inverse square root ("Noam") learning rate schedule (Vaswani et al., 2017) that reaches a peak LR of $4e^{-5}$ after one epoch of training. We found this configuration to yield results comparable to our previous setup, but at noticeably higher training efficiency. Table 3 shows the full set of hyperparameters.

The above setup works robustly across languages, with Tamil being the only exception, reaching only ca. 54 ELAS F1 on the development data. For the low resource setting of parsing Tamil, we hence use a batch size of 1, a lower learning rate ($1e^{-5}$), as well as a longer warmup time (5 epochs) and higher early stopping patience (40 epochs).

Language	Ensemble composition
Czech	3xPDT, 1xCAC, 1xFictree
Dutch	3xAlpino, 2xLassySmall
English	3xEWT, 2xGUM
Estonian	4xEDT, 1xEWT
Polish	5xPDB

Table 4: Ensemble compositions for languages with more than one training treebank.

We train 5 models per language and ensemble these models for our final predictions (see Sec. 2.4). For languages with more than one training treebank, we train models on all treebanks provided, with more models trained on larger treebanks. The one exception to this is Polish, where we found ensembling of models trained on both the PDB and LFG treebanks to yield worse results than just training on PDB (likely due to systematic annotation differences). Table 4 shows the ensemble composition for all languages with multiple training treebanks.

Each model is trained using a single nVidia Tesla V100 GPU, stopping early when ELAS F1 score on the development set does not improve for 20 epochs, or after at most 24 hours. Training time varies substantially by treebank and correlates with treebank size, with training being fastest for Tamil-TTB (ca. 2 hours on average) and slowest for Russian-SynTagRus and Czech-PDT (both run into the 24-hour time limit).

3.2 Results of Submission

Table 5 shows the results (in terms of ELAS F1 score) on the blind test data for our main submission (rightmost column, ensemble$_{\text{hyb}}$) as well as the 1$^{\text{st}}$- and 2$^{\text{nd}}$-scoring submissions of the Shared Task (TGIF and ShanghaiTech), as well as the median submission for each language. Our system achieves an average ELAS F1 score of 86.97 %, ranking 3$^{\text{rd}}$ with a margin of more than 3 points over the median.

The best results achieved by our system are for Bulgarian and Italian, each with ELAS F1 scores of over 93. In contrast, Tamil is the language that we perform by far the worst on, with an ELAS F1 score of around 59. In an extreme low-resource scenario such as parsing Tamil (where the training data consists of only 400 sentences), adaptions to our framework will be necessary.

	Other teams			RobertNLP		
Language	TGIF	Shanghai	*Median*	single	ensemble$_{\text{heur}}$	ensemble$_{\text{hyb}}$
Arabic	81.23	**82.26**	76.39	81.37	81.12	***81.58***
Bulgarian	**93.63**	92.52	90.84	92.94	92.91	***93.16***
Czech	**92.24**	91.78	89.08	89.99	89.51	***90.21***
Dutch	**91.78**	88.64	84.14	88.02	88.21	***88.37***
English	**88.19**	87.27	85.70	87.29	***87.89***	87.88
Estonian	**88.38**	86.66	84.02	86.10	86.52	***86.55***
Finnish	**91.75**	90.81	89.02	90.77	90.97	**91.01**
French	**91.63**	88.40	87.32	***88.59***	88.51	88.51
Italian	**93.31**	92.88	91.81	93.00	93.16	***93.28***
Latvian	**90.23**	89.17	84.57	88.68	88.80	***88.82***
Lithuanian	**86.06**	80.87	78.04	***80.98***	80.76	80.76
Polish	**91.46**	90.66	88.31	89.49	89.54	***89.78***
Russian	**94.01**	93.59	90.90	92.55	92.33	***92.64***
Slovak	**94.96**	90.25	87.04	89.60	89.29	***89.66***
Swedish	**89.90**	86.62	84.91	87.72	88.02	***88.03***
Tamil	**65.58**	58.94	52.27	58.24	59.00	***59.33***
Ukrainian	**92.78**	88.94	86.92	88.56	***88.86***	***88.86***
Average	**89.24**	87.07	83.64	86.70	86.78	***86.97***

Table 5: Parsing results (ELAS F1) on blind test data in the IWPT 2021 Shared Task. ensemble$_{\text{hyb}}$ is our main submission, using both the re-lexicalization heuristic and the label transducer.

3.3 Analysis of Results

To tease out the effects of re-lexicalization and ensembling, we submitted two more experiments on the blind test data after the official deadline.

Effect of re-lexicalization strategy. In a first experiment, we did not use our machine learning-based label transducer for re-lexicalization of labels, instead relying only on the rule-based heuristic. The results of this experiment can be found in the column labelled "ensemble$_{\text{heur}}$" in Table 5.

Using only the heuristic for label lexicalization results in a modest, but noticeable accuracy hit across languages, reducing the average ELAS F1 score by roughly 0.2. The languages most affected are Czech (-0.70), Arabic (-0.46), Slovak (-0.35), and Tamil (-0.33), in which differences between lexical material in the sentence and their lemmas included in lexicalized labels are frequent. In contrast, many languages see only small or no performance drops (e.g. Lithuanian, Swedish); for English, performance even increases very slightly when removing the label transducer.

These results indicate that while using our hybrid system is beneficial, good results for most languages can also be achieved when relying solely on our re-lexicalization heuristic. This makes it conceivable that in conjunction with a high-accuracy lemmatizer, a purely rule-based system may perform on par with a hybrid system, and we view this as an interesting avenue for future work.

Effect of ensembling. In a second experiment, we did not perform model ensembling for prediction, instead only using a single model for each language. The column labelled "single" in Table 5 reports the best results achieved for each language when using only the best single model.

As can be seen, utilizing only a single model per language results in a moderate average performance drop of 0.27 ELAS F1 points. With the exception of French and Lithuanian, all languages benefit from model ensembling, with Tamil (+1.09), English (+0.59), Estonian (+0.45), and Dutch (+0.35) showing the strongest improvements. As the latter three include data from different treebanks in their blind test sets, this indicates that ensembling may also help parser robustness when mixing models trained on different datasets.

However, although the overall effect of ensembling is notable, our parser nonetheless retains a relatively strong performance even without it, and still would have scored 3$^{\text{rd}}$ in the Shared Task if it was only using single models.

4 Conclusion

In this paper, we have described our submission to the IWPT 2021 Shared Task, which ranked 3$^{\text{rd}}$ out of 9 with an average ELAS F1 score of 86.97. Our model is an extension of the previously English-only system (Grünewald and Friedrich, 2020), demonstrating that the same approach is also yields very good results for other languages with only relatively minor modifications. In post-submission ablation experiments, we find that our parser benefits from model ensembling and a machine learning-assisted approach to label lexicalization.

A remaining issue of our parser is its rather poor performance in a low-resource setting (Tamil). Addressing this weakness, ideally while maintaining the parser's relatively simple core architecture, may be a promising avenue for future work.

References

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2021. From raw text to enhanced universal dependencies: The parsing shared task at iwpt 2021. In *Proceedings of the 17th International Conference on Parsing Technologies (IWPT 2021)*, pages 146–157, Bangkok, Thailand (online). Association for Computational Linguistics.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2020. Overview of the IWPT 2020 shared task on parsing into enhanced Universal Dependencies. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 151–161, Online. Association for Computational Linguistics.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. pages 8440–8451.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Timothy Dozat and Christopher D. Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490, Melbourne, Australia. Association for Computational Linguistics.

Stefan Grünewald and Annemarie Friedrich. 2020. RobertNLP at the IWPT 2020 shared task: Surprisingly simple enhanced UD parsing for English. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 245–252, Online. Association for Computational Linguistics.

Stefan Grünewald, Annemarie Friedrich, and Jonas Kuhn. 2021. Applying Occam's razor to transformer-based dependency parsing: What works, what doesn't, and what is really necessary. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies*, Online. Association for Computational Linguistics.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland. European Language Resources Association (ELRA).

Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 80–90, Online. Association for Computational Linguistics.

Joakim Nivre, Paola Marongiu, Filip Ginter, Jenna Kanerva, Simonetta Montemagni, Sebastian Schuster, and Maria Simi. 2018. Enhancing universal dependency treebanks: A case study. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 102–107, Brussels, Belgium. Association for Computational Linguistics.

Jenna Nyblom, Samuel Kohonen, Katri Haverinen, Tapio Salakoski, and Filip Ginter. 2013. Predicting conjunct propagation and other extended Stanford dependencies. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 252–261, Prague, Czech Republic. Charles University in Prague, Matfyzpress, Prague, Czech Republic.

Sebastian Schuster, Éric Villemonte de La Clergerie, Marie Candito, Benoît Sagot, Christopher Manning, and Djamé Seddah. 2017. Paris and Stanford at EPE 2017: Downstream evaluation of graph-based dependency representations. In *Proceedings of the 2017 Shared Task on Extrinsic Parser Evaluation (EPE 2017)*, pages 47–59.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).

Maria Simi and Simonetta Montemagni. 2018. Bootstrapping enhanced universal dependencies for Italian. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253. CEUR-WS.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.