RobertNLP at the IWPT 2020 Shared Task: Surprisingly Simple Enhanced UD Parsing for English

Stefan Grünewald[1,2] **Annemarie Friedrich**[2]
[1]Institut für Maschinelle Sprachverarbeitung, University of Stuttgart
[2]Bosch Center for Artificial Intelligence, Renningen, Germany
stefan.gruenewald|annemarie.friedrich@de.bosch.com

Abstract

This paper presents our system at the *IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*. Using a biaffine classifier architecture (Dozat and Manning, 2017) which operates directly on fine-tuned RoBERTa embeddings, our parser generates enhanced UD graphs by predicting the best dependency label (or absence of a dependency) for each pair of tokens in the sentence. We address label sparsity issues by replacing lexical items in relations with placeholders at prediction time, later retrieving them from the parse in a rule-based fashion. In addition, we ensure structural graph constraints using a simple set of heuristics. On the English blind test data, our system achieves a very high parsing accuracy, ranking 1st out of 10 with an ELAS F1 score of 88.94 %.

1 Introduction

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

Research on UD parsing has so far mostly focused on producing syntax trees according to the basic UD specification (e.g., in the CoNLL 2017 and 2018 Shared Tasks). Prior work on inducing enhanced UD graphs (Nyblom et al., 2013; Simi and Montemagni, 2018; Nivre et al., 2018) infers enhanced UD representations by first parsing text into basic UD trees and then adding enhanced relations by applying rule-based or machine-learning modules. This approach has the disadvantage of propagating errors in the basic layer to the enhanced parse. For our submission to the IWPT 2020 Shared Task (Bouma et al., 2020), we follow an alternative approach. We do not distinguish between the basic UD tree and the enhanced part of the graph, instead treating all types of dependencies equally and extracting them jointly.

Following the approach of Dozat and Manning (2018), we use a biaffine classifier architecture in which we predict 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. Similar to Kondratyuk and Straka (2019), we extract the inputs for the biaffine classifier directly from fine-tuned contextualized word embeddings, RoBERTa (Liu et al., 2019b) in our case, using a scalar mixture of hidden layers (Liu et al., 2019a). We overcome the problem of sparsity issues caused by enhanced UD's large lexicalized label set by replacing lexical items with placeholders at prediction time and later retrieving them from the full parse via a set of rules. Surprisingly, this simple approach, combined with a straightforward heuristic ensuring that each node receives a head, results in valid enhanced UD graphs for 99% of all sentences in the English blind test data.

Despite being conceptually simple and easy to implement, our system sets a new state of the art for enhanced UD parsing for English, scoring first out of ten submissions on the blind test data according to the official ELAS evaluation metric. While

Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task, pages 244–251

https://doi.org/10.18653/v1/P17

our system is currently available only for English, adapting it to most other languages should be feasible with relatively little effort.

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 the StanfordNLP system (Qi et al., 2018), which achieved state-of-the-art results for these tasks on the English treebanks in the CoNLL 2018 Shared Task.

2.2 Input Token Representation

We use RoBERTa (Liu et al., 2019b) 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 RoBERTa by feeding each token as identified by StanfordNLP into the RoBERTa tokenizer. In addition, we prepend a special *[root]* token to each sentence, which serves as an artificial head of the *root* relation, which must be present in every sentence. This token receives a fixed, learned embedding instead of a contextualized RoBERTa embedding, but with the same number of dimensions.

Following Kondratyuk and Straka (2019), our model produces an embedding $\mathbf{r}_i$ for the original token at position i by forming a weighted sum of the hidden layers' embeddings at the positions corresponding to the first wordpiece token of the original token. Weights for this scalar mixture of layers are learned during training. Layers are randomly dropped during training to prevent the model from focusing on only a single layer.

We also experimented with using BERT (Devlin et al., 2019) instead of RoBERTa, but found that this yielded lower parsing accuracy (see Sec. 3).

2.3 Dependency Classification

Figure 1 shows an overview of our neural-network based dependency classifier, which simulatenously predicts relation labels or absence of a relation 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.

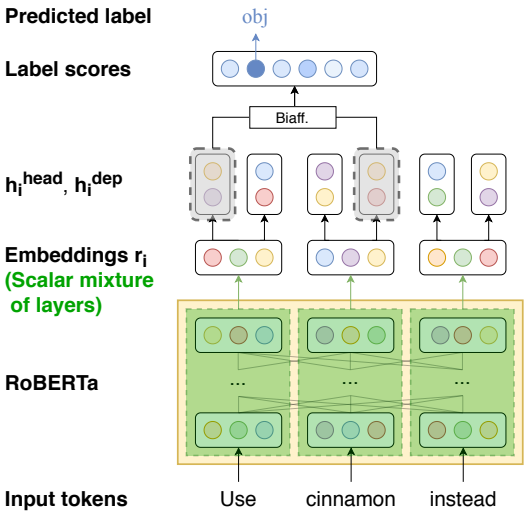


Figure 1: Architecture of neural network predicting dependency relations between pairs of tokens.

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 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 recent approaches that first predict presence or absence of relations and then use a second classifier to predict labels. It has already been proposed by Dozat and Manning (2018), who found that it performed on par with the factorized approach.

Finally, the most likely label $y_{i,j}$ can then be extracted from these 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

[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.

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 dependency labels is huge; the EWT corpus contains 399 unique 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. 2.4 for a detailed description of the reconstruction process). This way, we can remove all lexicalized relations from the label vocabulary, instead adding only five new placeholder labels: *nmod:[case]*, *obl:[case]*, *acl:[mark]*, *advcl:[mark]*, and *conj:[cc]*. We keep all other, non-lexicalized subtyped labels (such as *nmod:poss*). This brings the total label count down to 56 (including ∅).

2.4 Post-processing

The outputs provided by the dependency classifier can be regarded as a 3-dimensional tensor, or in other words, each cell in the matrix as shown in Figure 2 contains the probabilities predicted for the label set with the row label corresponding to the relation's head and the column label corresponding to the relation's dependent. Figure 2 shows the highest-scoring label per entry.

Ensuring graph structure constraints. Using the outputs provided by the dependency classifier, we can assemble a dependency graph by retrieving the highest-scoring dependency (or ∅, i.e., no relation) for each pair of tokens in the sentence (omitting the diagonal as enhanced UD does not allow links starting and ending at the same node) and forming their union.

Although enhanced UD eliminates the requirement that dependency graphs must be trees, it maintains a set of structural constraints. Specifically, each token needs to have at least one head and must be reachable from at least one of the root(s) of the graph.[2] These global constraints are not automatically adhered to by our simple graph construction method, which operates on pairs of tokens. Nonetheless, we observe that around 99 % of sentences are assigned structurally valid graphs as determined by the official validation script.[3]

	[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.

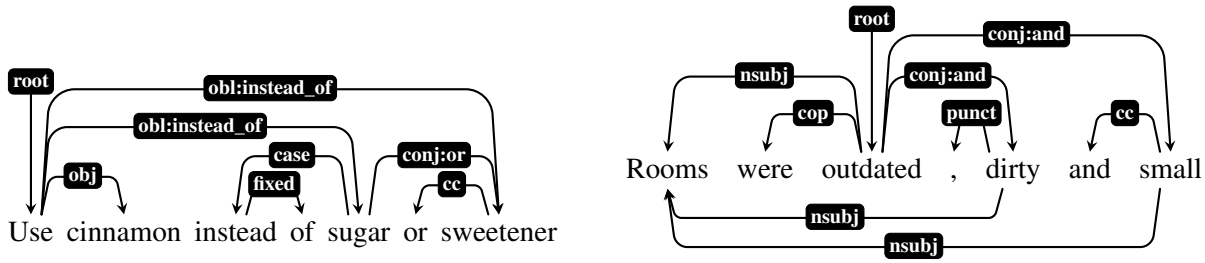
To make the graphs of the remaining 1 % sentences structurally valid, we perform the following steps. In the case of tokens lacking a head, the ∅ label has received the highest score during classification for all possible heads. We now simply retrieve all second-ranked labels and their scores and pick the relation (and corresponding head) that received the highest score across all possible heads.

Further, in order to ensure reachability from the root, in the cases violating this constraint, we fall back to an external dependency tree parse, i.e., a representation of the UD basic layer, for generating candidate links to be added to our graph. We here use the UDify parser (Kondratyuk and Straka, 2019) to predict basic UD trees. We determine the set of nodes V that are not reachable from any root, and for each node $v \in V$ we compute the number of nodes in V that can be reached when starting at v. We then pick the node v_i that can reach the largest number of nodes and check if the head of v_i in the basic layer tree can be when starting at a root in *our* graph. If so, we add the relation between v_i and its head as present in the basic layer tree to our graph, otherwise, we add a dep edge from the sentence's first root to v_i. We repeat this procedure until each node in the graph is reachable from at least one root node.

Re-lexicalization of labels. 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-

[2] Graphs in enhanced UD may have more than one root.

[3] https://github.com/UniversalDependencies/tools/blob/master/validate.py



(a) Dependency graph with lexicalization of labels.

(b) Dependency graph with "conjunction siblings."

Figure 3: Re-lexicalization of dependency labels in the presence of conjunctions.

lexicalize predicted placeholder labels using the following set of rules.

First, if the token has a dependent that is attached via the placeholder of the de-lexicalized relation in question, we lexicalize the relation with the token of this dependent. For example, in Figure 3a, our parser predicts *obl:[case]* and we 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.)

If such a dependent does not exist, it may be due to the presence of a *conj* relation. For example, Figure 3a shows a case where for the de-lexicalized link *obl:[case]* ending at "sweetener," no *case* relation starts at this node. This is due to the presence of a *conj* relation, ending at "sweetener." We hence check if the head of the *conj* relation has an incoming lexicalized edge of the same base relation (here *obl*) and if so, re-lexicalize accordingly.

Similarly, *conj* links ending at siblings in coordinate constructions (here "dirty" and "small") are always lexicalized with the same item (in this case "and"). Unlike "small," the dependent "dirty" does not have its own *cc* dependent that could be used to execute the first step, i.e., to replace the placeholder of *conj:[cc]* with a dependent's token. For such nodes, we hence search the graph for siblings that are linked to the common governor via *conj* relations. If we find any, we use the lexicalized label of the corresponding *conj* relation for all siblings.

The above heuristics return a result for 98.9 % of the de-lexicalized relations predicted for the blind test data; in the remaining cases, we simply remove the placeholder without substituting any lexical material. Provided that the underlying base relation was predicted correctly, we are able to retrieve the correct lexical material for 98.4 % of relations.

Removal of relations. In addition, 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*. However, in around 0.4 % of cases our parser erroneously predicts several of these relations for a single token (e.g., punctuation being attached to several tokens at once). In these cases, we remove all but the most confidently predicted dependency.

3 Experiments

This section describes our submission to the Shared Task, as well as a number of additional experiments we conducted to contextualize our results.

3.1 Experimental Settings and Hyperparameters

We use the training and development sections of the EWT corpus for training and validation, respectively. We use gold-tokenized and gold-segmented sentences as input for our system during training.

For hyperparameter settings, we mostly stick with the values used by Kondratyuk and Straka (2019). An exception to this is the training regime, where we found a low batch size, constant learning rate, no gradient clipping, and the AdamW optimizer (Loshchilov and Hutter, 2019) to yield the best results. The final hyperparameters can be found in Table 1.

Our model was trained using a single nVidia Tesla V100 GPU, stopping early when ELAS F1 score on the development set did not improve for 10 epochs. The best model was found after 63 epochs, i. e., 73 training epochs were performed in total, taking ca. 9 hours. Parsing the English blind test set (3077 sentences) takes around 3 minutes in total, i.e. 0.06 seconds per sentence.

RoBERTa 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	5
Learning rate	$5e^{-6}$
β_1, β_2	0.9, 0.999
Weight decay	0.0

Table 1: Hyperparameter values.

Submission	IWPT-all	EWT	PUD
RobertNLP	**88.94**	**88.06**	**89.97**
TurkuNLP	87.15	86.14	88.35
median	83.41	82.04	85.02
Køpsala	65.37	64.18	66.77
UDify + converter	85.67	84.55	87.00

Table 2: Parsing results (ELAS F1) on English blind test data in the IWPT 2020 Shared Task.

3.2 Results of Submission

Table 2 shows the results (in terms of ELAS F1 score) on the blind English test data for our system as well as the highest- and lowest-ranking competing submissions and the median submission. Our system achieves an ELAS F1 score of 88.94 %, ranking first with a margin of more than 1.5 points over the second-ranking submission.

As an additional baseline, we used the state-of-the-art UDify parser (Kondratyuk and Straka, 2019) to predict basic dependencies and then ran the rule-based converter by Schuster and Manning (2016) on the output to extract enhanced relations. This approach achieved an F-Score of 85.67 %, considerably lower than our system, confirming that our end-to-end graph parsing approach is superior to a pipeline model of basic parsing + rule-based conversion.

3.3 Analysis of Results

We here describe several experiments using variations of the setting used in our official submission. These experiments aim at determining the impact of different factors, including choice of pre-trained embeddings, training data, as well as segmentation and tokenization, on model performance. Some of the experiments described in this section were conducted during the development of our system, others constitute post-evaluation analyses. For consistency, we present results for the blind test data in this section. Most experiments were initially conducted using the development data, showing the same tendencies.

Embeddings	Train	IWPT-test	EWT-dev
BERT-base	EWT	87.49	87.64
RoBERTa-base	EWT	88.17	88.64
BERT-large	EWT	88.18	88.61
RoBERTa-large	EWT	**88.94**[a]	**89.43**
RoBERTa-large	UD2.5[b]	87.85	88.59

Table 3: Effect of embeddings and training data on model performance (ELAS F1, English blind test data). [a]Official submission. [b]Concatenation of EWT, GUM, LinES, and ParTUT training data.

Choice of pre-trained embeddings. We experiment with four different pre-trained embedding models, namely BERT and RoBERTa in their base and large variants respectively. As shown in Table 3, RoBERTa outperforms BERT, and the large variants outperform the base variants, with BERT-large and RoBERTa-base performing roughly equally. The best observed results are achieved by RoBERTa-large (our official submission). The superior performance of RoBERTa may stem from the fact that it was pre-trained on a considerably larger amount of data, and that it dropped the "next sentence prediction" objective, which may be irrelevant or even detrimental for a single-sentence task like syntactic parsing.

Effect of additional training data. While preparing our submission, we experimented with generating additional training data by using the rule-based UD enhancer by Schuster and Manning (2016), which was used to create the gold standard enhanced layers of the EWT and PUD corpora, to build enhanced versions of three other English UD treebanks (GUM, LinES, and ParTUT).

However, we found in preliminary experiments on the dev and test sections of the above mentioned corpora that including this additional training data actually slightly hurts performance if the test data is from a different corpus. This is correlated with the lexical distance between test and training data as computed using the Bhattacharyya distance

$$D_B(p,q) = -\ln \sum_{x \in X} \sqrt{p(x)q(x)} \qquad (6)$$

Corpus	Lex. dist.	ELAS F1
EWT	0.142	88.94
GUM	0.204	87.98
LinES	0.240	88.20
ParTUT	0.248	87.69

Table 4: Lexical (Bhattacharyya) distance and parsing accuracy between the blind test data and the different training corpora. The rightmost column indicates parsing performance on the IWPT test set when adding the respective corpus to the EWT training data. (First line is EWT only.)

between the respective vocabulary probability distributions (Bhattacharya, 1943; Ruder and Plank, 2017).

As the lexical distance of the blind test set and EWT is much smaller than the ones between the test set and the other corpora (see Table 4), our official submission's model was trained only on EWT. Post-evaluation experiments (see rightmost column) confirm that when including corpora with higher lexical distance, parsing accuracy decreases. In addition, parsing results on the blind test set when including all additional data (results see last line in Table 3) confirm this approach. However, if a different test set showed greater similarity to other corpora, including them as training data would likely be beneficial. As one of the anonymous reviewers points out, in addition to lexical similarity, factors such as mean dependency distance or average sentence length may also play a role. In conclusion, our experiments once more highlight that selecting good training corpora for an application domain is a critical factor and an interesting direction for further research.

Effect of segmentation and tokenization. While our parser was trained on gold-tokenized and gold-segmented sentences, the Shared Task required parsing from raw text. In order to determine the extent to which automatic segementation and tokenization impacts results, we run our parser on the gold-tokenized and gold-segmented version of the test data.

We observe an ELAS F1 score of 90.80, which constitutes an increase of nearly 2 points over the results obtained using automatic segmentation. This indicates that our system is rather sensitive to these kinds of errors and would greatly benefit from improvements in segmentation accuracy. It might also be possible to increase the robustness of our system w. r. t. these errors by training it on system-predicted sentence segmentation.

Performance on basic vs. enhanced relations. We further evaluate how performance of our parser varies between (a) relations that result from enhancements, i. e., relations which are exclusive to the enhanced layer, and (b) relations that occur in the basic layer as well. Because our parser does not differentiate between basic and enhanced relations internally, we can only compute recall for the two classes, but not precision and F1.[4] We perform this evaluation for gold-segmented and gold-tokenized input.

Recall is considerably lower on relations exclusive to the enhanced layer (83.64 %) as opposed to relations that are also present in the basic layer (91.60 %), indicating that predicting the former is indeed a more difficult task compared to predicting the latter, as might be expected. The result further suggests that it might be promising to use our parser architecture in combination with a spanning tree algorithm to predict basic-layer style trees as well (e. g. in a multi-task setting). This would also eliminate the need to rely on external parser input to post-process dependency graphs for the rare cases of invalid graphs.

Performance on individual label types. Finally, while our system achieves a high parsing accuracy overall, we also compute F1-Scores for each individual label type in order to obtain a finer-grained picture of its strengths and weaknesses. Again, we perform this evaluation for gold-segmented and gold-tokenized input. A selection of the results is displayed in Table 5.

As might be expected, the label types on which our parser performs best are highly common functional relations such as *det* and *case*, as well as frequent content word dependencies such as *nsubj* and *amod*. More interestingly, it also performs close to the average on *nsubj:xsubj*, which is not only considerably rarer than the aforementioned relations but also exclusive to the enhanced representation, demonstrating that our joint approach is capable of capturing these dependencies as well.

Somewhat more challenging are the *flat* and *compound* labels (85.53 and 83.51 F-Score, respectively), which are used to annotate multiword ex-

[4]We compare to the gold standard which distinguishes between basic and enhanced relations. Our parser does not differentiate between basic and enhanced relations, i.e., the full graph is constructed without internally identifying the subgraph corresponding to the basic syntactic tree.

Label	Freq.	ELAS F1
det	3879	99.04
case	4481	97.16
nsubj	3708	94.78
amod	2552	92.49
(Total/avg.)	48298	90.80
nsubj:xsubj	569	88.22
flat	482	85.53
punct	5519	84.28
compound	2005	83.51
appos	347	66.31
parataxis	301	59.35
list	251	47.72

Table 5: Parsing accuracy (ELAS F1) for a selection of label types. Scores were computed on gold-segmented test data. *Freq.* denotes the number of occurrences of the label in the gold annotations.

pressions. The computational identification and treatment of such expressions is very challenging and constitutes a long-standing research area in itself (Gregoire et al., 2007; Savary et al., 2018, 2019).

The *punct* relation harbors perhaps the greatest potential for improvement, yielding an F-Score of only 84.28 despite being extremely common. This is likely due to the rather complex set of rules that determines which token a piece of punctuation is attached to.[5] However, it might also be the label where improvements are most difficult to achieve, as the gold standard itself contains inconsistencies,[6] leading to a noisy training signal.

Finally, out of all label types which occur more than 200 times in the test data, the worst performance is observed on *appos*, *parataxis*, and *list*. While their low frequency is almost certainly part of the reason for this, it is also worth noting that these dependencies are unusual in that they represent "side-by-side" relations between words rather than more obviously hierarchical structures (as is the case for most other label types). Investigating parser performance on these kinds of constructions in greater detail may present a promising avenue for future work.

4 Discussion and Conclusion

With our submission to the IWPT 2020 Shared Task, we have demonstrated a conceptually simple, yet highly effective method for parsing Enhanced Universal Dependencies from English text.

Although we have focused on English in our submission, we believe that our system should in principle be easily adaptable to other languages as the only language-specific part of out model is its handling of lexicalized relations. While certain other languages (e. g., Czech or Estonian) have more complex label inventories including for example case information as well, this should not pose a problem for our delexicalization strategy. However, the adaptation might require a moderate amount of manual work, and it remains to be seen how effective the lexicalization strategy is for other languages, a question that may be addressed in future work.

[5] See https://universaldependencies.org/u/dep/punct.html.

[6] As noted on the treebank's Github page at https://github.com/UniversalDependencies/UD_English-EWT.

Acknowledgments

We thank Jonas Kuhn, Heike Adel, Jannik Strötgen, Lukas Lange and the anonymous reviewers for their useful comments regarding this work.

References

Anil Bhattacharya. 1943. On a measure of divergence between two statistical populations defined by their population distributions. In *Bulletin of the Calcutta Mathematical Society*, volume 35, pages 99–109.

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*, Seattle, US. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Nicole Gregoire, Stefan Evert, and Su Nam Kim, editors. 2007. *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*. Association for Computational Linguistics, Prague, Czech Republic.

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.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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).

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.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Sebastian Ruder and Barbara Plank. 2017. Learning to select data for transfer learning with Bayesian optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 372–382, Copenhagen, Denmark. Association for Computational Linguistics.

Agata Savary, Carla Parra Escartín, Francis Bond, Jelena Mitrović, and Verginica Barbu Mititelu, editors. 2019. *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*. Association for Computational Linguistics, Florence, Italy.

Agata Savary, Carlos Ramisch, Jena D. Hwang, Nathan Schneider, Melanie Andresen, Sameer Pradhan, and Miriam R. L. Petruck, editors. 2018. *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*. Association for Computational Linguistics, Santa Fe, New Mexico, USA.

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.