Adaptation of Multilingual Transformer Encoder for Robust Enhanced Universal Dependency Parsing

Han He
Computer Science
Emory University
Atlanta GA 30322, USA
han.he@emory.edu

Jinho D. Choi
Computer Science
Emory University
Atlanta GA 30322, USA
jinho.choi@emory.edu

Abstract

This paper presents our enhanced dependency parsing approach using transformer encoders, coupled with a simple yet powerful ensemble algorithm that takes advantage of both tree and graph dependency parsing. Two types of transformer encoders are compared, a multilingual encoder and language-specific encoders. Our dependency tree parsing (DTP) approach generates only primary dependencies to form trees whereas our dependency graph parsing (DGP) approach handles both primary and secondary dependencies to form graphs. Since DGP does not guarantee the generated graphs are acyclic, the ensemble algorithm is designed to add secondary arcs predicted by DGP to primary arcs predicted by DTP. Our results show that models using the multilingual encoder outperform ones using the language specific encoders for most languages. Moreover, the ensemble models generally show higher labeled attachment score on enhanced dependencies (ELAS) than the DTP and DGP models. As the result, our best parsing models rank the third place on the macro-average ELAS over 17 languages.

1 Introduction

Dependency parsing can generate computational structures for a wide range of typologically different languages, which provides structural relations that have been found to be useful for various NLP applications. However, these applications often require richer dependency relations carrying on deep semantics, which are missing in traditional dependency trees. Thus, enhanced dependencies emerge to explicitly capture deep semantic relations over surface structures (Schuster and Manning, 2016).

Recently, there has been lots of interests in constructing and parsing advanced graph structures beyond tree representations. Choi (2017) introduce deep dependency graphs that address several limitations in UD tree structures. Schuster et al. (2017) analyze gapping constructions in the enhanced UD representation. Nivre et al. (2018) evaluate both rule-based and data-driven systems for adding enhanced dependencies to existing treebanks. Apart from syntactic relations, researchers are moving towards semantic dependency parsing (Oepen et al., 2015) for more direct analysis of entities and events. The efforts of treebank construction stimulates the interest of many researchers in improving the performance of semantic parsers (Dozat and Manning, 2018; Du et al., 2015; Almeida and Martins, 2015).

This paper presents our parsing approach to the Shared Task on Enhanced Universal Dependencies at IWPT 2020 (Nivre et al., 2016; Bouma et al.).[1] Our system is a simplified version of the transformer-based dependency parsers presented by He and Choi (2020), which employs the deep biaffine dependency parsing decoder (Dozat and Manning, 2017) over the transformer encoder, BERT (Devlin et al., 2019). We simplify their network by removing the LSTM and fine-tuning their static transformer encoder. In order to effectively predict the enhanced dependencies, we also ensemble the dependency tree parser with an dependency graph parser through a greedy searching algorithm. At last, we perform extensive experiments on the effects of transformer encoders to perform a detailed analysis.

Our experiments show that the multilingual encoder has a substantial advantage over the language-specific encoders. Moreover, our analysis shows that tree parsing model can accurately predict primary dependencies in long sentences, while graph parsing model excels at label prediction. By taking advantages from both sides, our ensemble models outperform individual models in most languages.[2]

[1]This work purely focuses on parsing not pre-steps such as sentence split or tokenization, although we recognize that it is important to address the pre-steps to win this competition.
[2]All our resources are available at https://github.

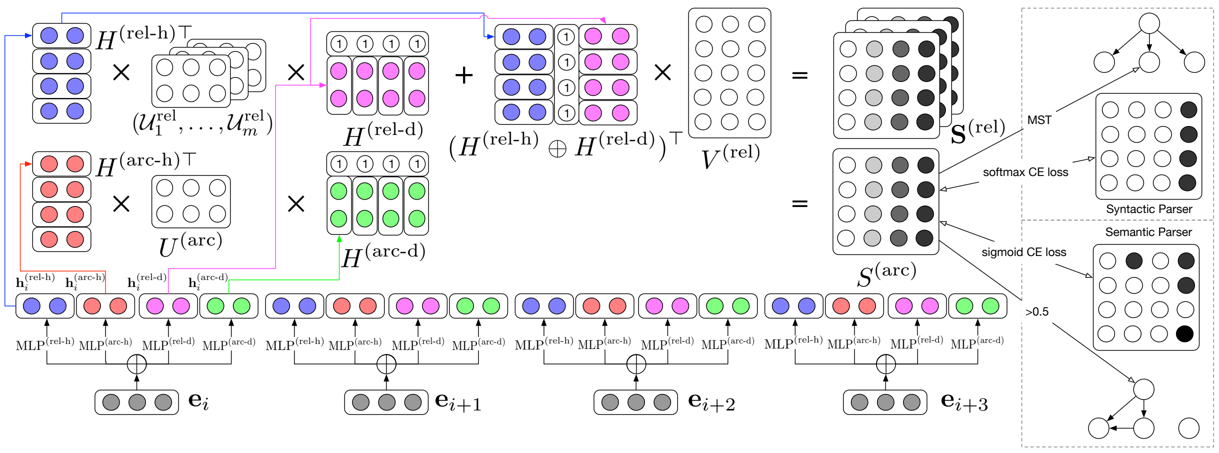
$H^{(\text{rel-h})\top}$
$(\mathcal{U}_1^{\text{rel}}, \ldots, \mathcal{U}_m^{\text{rel}})$
$H^{(\text{rel-d})}$
$(H^{(\text{rel-h})} \oplus H^{(\text{rel-d})})^\top$
$V^{(\text{rel})}$
$\mathbf{S}^{(\text{rel})}$
$H^{(\text{arc-h})\top}$
$U^{(\text{arc})}$
$H^{(\text{arc-d})}$
$S^{(\text{arc})}$
MST
softmax CE loss
sigmoid CE loss
>0.5
Syntactic Parser
Semantic Parser
$\mathbf{h}_i^{(\text{rel-h})}$ $\mathbf{h}_i^{(\text{arc-h})}$ $\mathbf{h}_i^{(\text{rel-d})}$ $\mathbf{h}_i^{(\text{arc-d})}$
MLP$^{(\text{rel-h})}$ MLP$^{(\text{arc-h})}$ MLP$^{(\text{rel-d})}$ MLP$^{(\text{arc-d})}$
$\mathbf{e}_i$
$\mathbf{e}_{i+1}$
$\mathbf{e}_{i+2}$
$\mathbf{e}_{i+3}$

Figure 1: The overview of our transformer-based biaffine dependency parsing model.

2 Approach

2.1 Preprocessing

The data in the training and development sets are already sentence segmented and tokenized. For the test set, UDPipe is used to segment raw input into sentences, where each sentence gets split into a list of tokens (Straka and Straková, 2017). A custom script written by us is used to remove multiwords but retain their splits (e.g., remove *vámonos* but retain *vámos nos*), as well as to collapse empty nodes in the CoNLL-U format.

2.2 Transformer Encoder

Our parsing models use contextualized embeddings generated by transformer encoders such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2020) or ALBERT (Lan et al., 2020) that are pretrained on large corpora for language modeling. Each sentence in the preprocessed data by Section 2.1 is fed into a transformer encoder that further splits every token into sub-tokens using SentencePiece (Kudo and Richardson, 2018). The sub-token embeddings from the last layer of the transformer encoder are fed into the biaffine decoder in Section 2.3.

2.3 Biaffine Decoder

Our dependency parsing approach is based on the biaffine decoder that has shown state-of-the-art results on syntactic tree and semantic graph parsing in both English and Chinese (He and Choi, 2020). This model is simplified from the original biaffine parser introduced by Dozat and Manning (2017) such that trainable token embeddings are removed and lemmas are used instead of word forms. This section proposes an even more simplified approach that no longer uses embeddings from POS tags, so it can be easily adapted to languages that may not have dedicated POS taggers, and drops the Bidirectional LSTM encoder while integrating the transformer encoder directly into the biaffine decoder to minimize the redundancy of multiple encoders for the generation of contextualized embeddings.

com/emorynlp/iwpt-shared-task-2020

Every token w_i in the input sentence is split into one or more sub-tokens by the transformer encoder (Section 2.2). The contextualized embedding that corresponds to the first sub-token of w_i is treated as the embedding of w_i, say $\mathbf{e}_i$, and fed into four types of multilayer perceptron (MLP) layers to extract features for w_i being a head (*-h) or a dependent (*-d) for the arc relations (arc-*) and the labels (rel-*) (k and l are the dimensions of the arc and label representations, respectively):

$$\begin{aligned}
\mathbf{h}_i^{(\text{arc-h})} &= \text{MLP}^{(\text{arc-h})}(\mathbf{e}_i) \in \mathbb{R}^{k\times 1} \\
\mathbf{h}_i^{(\text{arc-d})} &= \text{MLP}^{(\text{arc-d})}(\mathbf{e}_i) \in \mathbb{R}^{k\times 1} \\
\mathbf{h}_i^{(\text{rel-h})} &= \text{MLP}^{(\text{rel-h})}(\mathbf{e}_i) \in \mathbb{R}^{l\times 1} \\
\mathbf{h}_i^{(\text{rel-d})} &= \text{MLP}^{(\text{rel-d})}(\mathbf{e}_i) \in \mathbb{R}^{l\times 1}
\end{aligned}$$

All feature vectors, $\mathbf{h}_1^*, \ldots, \mathbf{h}_n^*$, from each representation are stacked into a matrix (n is the number of tokens in a sentence); these matrices together are used to predict dependency relations among every token pairs. Note that bias terms are appended to the feature vectors $\mathbf{h}_i^{(\text{*-d})}$ that represent dependent nodes to estimate the likelihood of a certain relation given only the head node:

$$
\begin{aligned}
H^{\text{(arc-h)}} &= (\mathbf{h}_1^{\text{(arc-h)}}, \dots, \mathbf{h}_n^{\text{(arc-h)}}) \in \mathbb{R}^{k \times n} \\
H^{\text{(arc-d)}} &= (\mathbf{h}_1^{\text{(arc-d)}}, \dots, \mathbf{h}_n^{\text{(arc-d)}}) \oplus \mathbf{1} \in \mathbb{R}^{(k+1) \times n} \\
H^{\text{(rel-h)}} &= (\mathbf{h}_1^{\text{(rel-h)}}, \dots, \mathbf{h}_n^{\text{(rel-h)}}) \in \mathbb{R}^{l \times n} \\
H^{\text{(rel-d)}} &= (\mathbf{h}_1^{\text{(rel-d)}}, \dots, \mathbf{h}_n^{\text{(rel-d)}}) \oplus \mathbf{1} \in \mathbb{R}^{(l+1) \times n}
\end{aligned}
$$

The bilinear and biaffine classifiers are then used for the arc and label predictions respectively, where $U^{\text{(arc)}}$, $U_i^{\text{(rel)}}$ and $V^{\text{(rel)}}$ are trainable parameters, and m is the number of dependency labels. In particular, a separate weight matrix $U_i^{\text{(rel)}}$ is dedicated to the prediction of each label:

$$
\begin{aligned}
S^{\text{(arc)}} &= H^{\text{(arc-h)}\top} \cdot U^{\text{(arc)}} \cdot H^{\text{(arc-d)}} \in \mathbb{R}^{n \times n} \\
\mathcal{U}_i^{\text{(rel)}} &= H^{\text{(rel-h)}\top} \cdot U_i^{\text{(rel)}} \cdot H^{\text{(rel-d)}} \in \mathbb{R}^{n \times n} \\
\mathbf{S}^{\text{(rel)}} &= (\mathcal{U}_1^{\text{(rel)}}, \dots, \mathcal{U}_m^{\text{(rel)}}) \\
&\quad + (H^{\text{(rel-h)}} \oplus H^{\text{(rel-d)}})^\top \cdot V^{\text{(rel)}} \in \mathbb{R}^{m \times n \times n}
\end{aligned}
$$

2.4 Dependency Tree & Graph Parsing

The arc score matrix $S^{\text{(arc)}}$ and the label score tensor $\mathbf{S}^{\text{(rel)}}$ generated by the bilinear and biaffine classifiers can be used for both dependency tree parsing (DTP) and graph parsing (DGP). For DTP, which takes only the primary dependencies to learn tree structures during training, the Chu-Liu-Edmond's Maximum Spanning Tree (MST) algorithm is applied to $S^{\text{(arc)}}$ for the arc prediction, then the label with largest score in $\mathbf{S}^{\text{(rel)}}$ corresponding to the arc is taken for the label prediction ($\mathcal{A}_{\text{DTP}}$: the list of predicted arcs, $\mathcal{L}_{\text{DTP}}$: the labels predicted for $\mathcal{A}_{\text{DTP}}$, $\mathcal{I}$: the indices of $\mathcal{A}_{\text{DTP}}$ in $\mathbf{S}^{\text{(rel)}}$):

$$
\begin{aligned}
\mathcal{A}_{\text{DTP}} &= \text{MST}(S^{\text{(arc)}}) \\
\mathcal{L}_{\text{DTP}} &= \text{argmax}(\mathbf{S}^{\text{(rel)}}[\mathcal{I}(\mathcal{A}_{\text{DTP}})])
\end{aligned}
$$

For DGP, which takes the primary as well as the secondary dependencies in the enhanced types to learn graph structures during training, the sigmoid function is applied to $S^{\text{(arc)}}$ instead of the softmax function (Figure 1) so that zero to many heads can be predicted per node by measuring the pairwise losses. Then, the same logic can be used to predict the labels for those arcs as follows:

$$
\begin{aligned}
\mathcal{A}_{\text{DGP}} &= \text{SIGMOID}(S^{\text{(arc)}}) \\
\mathcal{L}_{\text{DGP}} &= \text{argmax}(\mathbf{S}^{\text{(rel)}}[\mathcal{I}(\mathcal{A}_{\text{DGP}})])
\end{aligned}
$$

It is worth mentioning that the performance of DTP is generally better than the one achieved by DGP for finding the primary dependencies that form tree structures; however, DTP completely dismisses the secondary dependencies so that DGP outperforms DTP for the overall performance on the enhanced dependencies. Section 2.5 describes our ensemble parsing approach that adapts the best of both worlds by taking the predictions of primary dependencies from DTP and augmenting them with the predictions of secondary dependencies from DGP.

2.5 Ensemble Parsing

The UD guidelines require a graph formed by only primary dependencies to be always a spanning tree, while such a restriction is not applied to graphs with secondary dependencies. We find that the majority of dependency graphs in the training set, however, can be viewed as directed acyclic graphs (DAGs). In fact, all graphs can be transformed into DAGs by removing 0.87% of the secondary dependencies. Therefore, our ensemble parsing method focuses on building maximum spanning DAGs (MSDAGs) by combining arcs from both dependency trees and graphs generated by the DTP and DGP models, respectively (Section 2.4).[3]

Unfortunately, finding MSDAGs from the output of the DGP model is NP-hard (Schluter, 2014). Thus, we design an ensemble approach that finds approximate MSDAGs using a greedy algorithm. Given the score matrices $S_{\text{DTP}}^{\text{(arc)}}$ and $S_{\text{DGP}}^{\text{(arc)}}$ from the DTP and DGP models respectively and the label score tensor $\mathbf{S}_{\text{DGP}}^{\text{(rel)}}$ from the DGP model, Algorithm 1 is applied to compute the MSDAG:
The algorithm begins by initializing scores related to the root in $\mathbf{S}_{\text{DGP}}^{\text{(rel)}}$ (`L1-3`). The label matrix R is created by taking the argmax of every dependent and head pair (d, h) in $\mathbf{S}_{\text{DGP}}^{\text{(rel)}}$ such that each cell contains the most likely label for that pair (`L4`). Given the arc list $\mathcal{A}_{\text{DTP}}$ from the DTS model (`L5`), the graph G is generated by taking all arcs in $\mathcal{A}_{\text{DTP}}$ and their corresponding labels in R (`L6-9`).[4] Finally, given the arc list $\mathcal{A}_{\text{DGP}}$ from the DGP model sorted in descending order (`L10`), arcs in $\mathcal{A}_{\text{DGP}}$ are greedily added to G, as long as they do not create any

[3]The motivation behind this DAG approach was to reduce potential confusion in learning caused by cyclic structures, which we later realized may have not been necessary, but we described this approach here for the replicability of our work.

[4]$\mathbf{S}_{\text{DGP}}^{\text{(rel)}}$ is used to find the labels of arcs predicted by both the DTP and DGP models. From our experiments, we find that the DGP model outperforms the DTP model for the label predictions of even primary dependencies, which may be due to the greater number of labels in DGP training data; thus, $\mathbf{S}_{\text{DGP}}^{\text{(rel)}}$ is used for all types of dependencies instead of $\mathbf{S}_{\text{DTP}}^{\text{(rel)}}$.

	AR	BG	CS	EN	ET	FI	FR	IT	LT	LV	NL	PL	RU	SK	SV	TA	UK
TRN	6,075	8,907	102,133	12,543	25,749	12,217	2,231	13,121	2,341	10,156	18,051	31,496	48,814	8,483	4,303	400	5,496
DEV	909	1,115	11,182	2,002	3,125	1,364	412	564	617	1,664	1,394	3,960	6,584	1,060	504	80	672
TST	794	1,112	12,713	2,800	3,588	2,616	2,679	482	652	1,835	1,154	4,923	6,495	1,052	2,258	122	905
> 256	4	0	0	0	2	0	0	0	0	0	2	0	0	0	0	0	0

(a) Sentence counts. The > 256 row shows the number of sentences in the test set whose lengths are greater than 256 tokens.

	AR	BG	CS	EN	ET	FI	FR	IT	LT	LV	NL	PL	RU	SK	SV	TA	UK
TRN	254.3K	124.3K	1783.1K	204.6K	361.8K	163.0K	51.9K	294.4K	47.6K	167.6K	261.0K	388.5K	870.5K	80.6K	66.6K	6.8K	92.4K
DEV	34.2K	16.1K	187.3K	25.1K	44.6K	18.3K	10.3K	12.7K	11.6K	26.0K	22.9K	48.0K	118.5K	12.4K	9.8K	1.4K	12.6K
TST	30.8K	15.7K	220.5K	46.3K	58.7K	36.9K	35.3K	11.2K	10.8K	26.4K	22.6K	65.7K	117.4K	13.0K	39.3K	2.1K	17.1K

(b) Token counts in thousands (K).

Table 1: Statistics of the training (TRN), development (DEV), and test (TST) sets preprocessed by UDPipe. AR: Arabic, BG: Bulgarian, CS: Czech, EN: English, ET: Estonian, FI: Finnish, FR: French, IT: Italian, LT: Lithuanian, LV: Latvian, NL: Dutch, PL: Polish, RU: Russian, SK: Slovak, SV: Swedish, TA: Tamil, UK: Ukrainian.

Algorithm 1: Ensemble parsing algorithm

```
Input: $S^{(\text{arc})}_{\text{DTP}}$, $S^{(\text{arc})}_{\text{DGP}}$, and $\mathbf{S}^{(\text{rel})}_{\text{DGP}}$
Output: $G$, that is an approximate MSDAG
1  $r \leftarrow \text{root\_index}(\mathcal{A}_{\text{DTP}})$
2  $\mathbf{S}^{(\text{rel})}_{\text{DGP}}[\text{root}, :, :] \leftarrow -\infty$
3  $\mathbf{S}^{(\text{rel})}_{\text{DGP}}[\text{root}, r, r] \leftarrow +\infty$
4  $R \leftarrow \text{argmax}(\mathbf{S}^{(\text{rel})}_{\text{DGP}})) \in \mathbb{R}^{n \times n}$
5  $\mathcal{A}_{\text{DTP}} \leftarrow \text{MST}(S^{(\text{arc})}_{\text{DTP}})$
6  $G \leftarrow \emptyset$
7  foreach arc $(d, h) \in \mathcal{A}_{DTP}$ do
8      $G \leftarrow G \cup \{(d, h, R[d, h]\}$
9  end
10 $\mathcal{A}_{\text{DGP}} \leftarrow \text{sorted\_descend}(\text{SIGMOID}(S^{(\text{arc})}_{\text{DGP}}))$
11 foreach arc $(d, h) \in \mathcal{A}_{DGP}$ do
12     $G^{(d,h)} \leftarrow G \cup \{(d, h, R[d, h]\}$
13     if is_acyclic$(G^{(d,h)})$ then
14         $G \leftarrow G^{(d,h)}$
15     end
16 end
```

cycle in G (L11-16).

2.6 Postprocessing

As mentioned in Section 2.1, empty nodes in the enhanced dependencies are collapsed before training using the script provided by the UD project.[5] Once dependency structures are generated by any parsing model, empty nodes are restored using our custom script.[6] At last, the postprocessing script provided by the UD project is applied to normalize the Unicode encoding and amend the `SpaceAfter=No` annotation as recommended by the organizers.[7]

3 Experiments

3.1 Datasets

Table 1 shows the statistics of data splits used for our experiments, that are preprocessed by UDPipe trained on UD v2.5 (Straka and Straková, 2017). Training and development sets for treebanks from the same languages are concatenated together. In particular, the following treebanks are merged for Czech, Estonian, and Dutch such that no individual models are developed for those treebanks:

- Czech: `UD_Czech-CAC/FicTree/PDT`
- Estonian: `UD_Estonian-EDT/EWT`
- Dutch: `UD_Dutch-Alpino/LassySmall`

Since transformer encoders usually restrict the input sequence length to be under 512 sub-tokens, our

[5] https://github.com/UniversalDependencies/tools/blob/master/enhanced_collapse_empty_nodes.pl

[6] https://gist.github.com/hankcs/776e7d95c19e5ff5da8469fe4e9ab050

[7] https://github.com/UniversalDependencies/tools/blob/master/conllu-quick-fix.pl

	AR	BG	CS	EN	ET	FI	FR	IT	LT	LV	NL	PL	RU	SK	SV	TA	UK
DTP	61.28	79.58	78.37	75.70	68.08	70.37	85.29	75.21	63.91	71.68	71.08	74.33	74.62	70.12	70.92	**54.26**	77.04
DGP	63.56	86.66	79.38	82.31	75.94	72.03	74.35	86.46	61.59	71.58	76.94	70.39	83.19	81.37	77.39	40.10	79.56
ENS	**67.26**	**88.19**	**85.51**	83.24	**81.36**	80.54	81.97	87.83	**66.12**	**79.19**	**80.72**	**82.39**	**88.60**	**82.72**	**78.19**	46.67	**79.69**
DTP	49.38	55.76	71.73	76.99	44.61	72.40	**86.23**	75.50	-	-	70.95	57.35	63.51	30.41	-	-	-
DGP	43.71	45.25	68.47	83.22	43.90	79.38	78.87	86.45	-	-	76.46	51.90	63.44	26.03	-	-	-
ENS	48.02	52.16	51.05	**85.30**	51.82	**82.96**	81.45	**88.52**	-	-	80.02	59.59	71.19	30.08	-	-	-

(a) Labeled attachment score on enhanced dependencies (ELAS) on the test sets.

	AR	BG	CS	EN	ET	FI	FR	IT	LT	LV	NL	PL	RU	SK	SV	TA	UK
DTP	**71.83**	**89.61**	**87.20**	84.61	**82.81**	85.43	87.47	90.49	**73.55**	**81.17**	**83.67**	**88.29**	**89.71**	**86.83**	**81.72**	**58.66**	**85.36**
DGP	65.49	87.31	80.38	82.74	76.42	72.62	75.49	86.98	63.10	71.95	77.37	74.54	83.47	83.39	78.32	41.36	80.18
ENS	69.46	88.84	86.63	83.68	81.98	81.44	83.34	88.35	68.24	79.66	81.21	86.79	88.93	84.73	79.11	48.50	80.34
DTP	56.69	62.27	79.53	**86.24**	53.31	**88.19**	**88.28**	**90.89**	-	-	83.47	67.58	76.05	34.50	-	-	-
DGP	46.75	46.84	69.79	83.67	44.62	80.10	80.12	86.92	-	-	76.97	55.58	64.16	26.82	-	-	-
ENS	51.37	54.01	53.45	85.76	52.74	83.70	82.83	89.04	-	-	80.59	63.82	72.13	31.19	-	-	-

(b) Labeled attachment score on enhanced dependencies where labels are restricted to the UD relation (EULAS).

Table 2: Parsing results on the test sets for all languages. For both (a) and (b), the rows 2-4 show the results by the multilingual encoder and the rows 5-7 show the results by the language-specific encoders if available.

parsing models cannot handle sentences beyond this length. As the distribution of sentence lengths in each dataset is measured, we find out that most sentences consist of fewer than 256 tokens. Thus, we discard sentences beyond 256 tokens from all training and development sets. For such sentences in the test sets, we rely on the parsing outputs from UDPipe; this choice is made due to the negligible numbers of those sentences (Table 1a) although it can be obviously improved.

3.2 Encoder Models

Two types of transformer encoders are used for the development of our models. One is the multilingual BERT (mBERT) pretrained on a mixture of large corpora in 100 languages (Devlin et al., 2019). The mBERT encoder uses one model to generate token embeddings for all languages, which encourages transfer learning in multilingual parsing. The other is language specific encoders that have been made to public by the community. Table 3 shows details about 12 language-specific encoders.[8] More details about the sources of these models are described in Section A.2.

3.3 Development Configuration

Following He and Choi (2020), we use the AdamW optimizer (Loshchilov and Hutter, 2019) with a linear learning rate warm-up and decay for finetuning the pretrained encoders. For the decoder weights, we use the Adam optimizer (Kingma and Ba, 2015) with a learning rate 20 times smaller than the one for finetuning. For the contextualized embeddings, we apply a shared dropout mask for each time step similar to variational dropout often used for recurrent neural networks (Gal and Ghahramani, 2016).

The KMeans clustering algorithm is adopted to bucket sentences into mini-batches according to their lengths counted by sub-tokens. The NVIDIA RTX GPUs with 24GB memory are used to develop these models. Unfortunately, most of our models cannot be fit into GPUs with smaller memory due to the extensive memory use of both the encoder and the decoder. We will explore innovative ways of reducing our parsing models such as teacher-student learning (Shin et al., 2019).

3.4 Parsing Results

All models are evaluated with 5 metrics, unlabeled attachment score (UAS), labeled attachment score (LAS), content labeled attachment score (CLAS), LAS on enhanced dependencies where labels are restricted to the UD relation (EULAS), and LAS on enhanced dependencies (ELAS). Models with the highest ELAS on the development sets are used to generate the final parse outputs on the test sets. Table 2 shows the ELAS and EULAS on the test sets for all languages. Detailed parsing results evaluated with all 5 metrics are described in Section A.3. For ELAS, our ensemble models (ENS) outperform the other models on 15 out of 17 languages. The only 2 exceptions are French and Tamil; these two languages consist of relatively fewer numbers of multi-head tokens as illustrated in Figure 2. Out of 12 languages with language-specific encoders (Table 3), models using the multilingual encoder outperform 8 of them, indicating the promise of the multilingual encoder to build robust parsing models for low-resource languages. The 4 exceptions

[8]We could not find public models for the other 5 languages.

Lang.	Encoder	Corpus	Provider
AR	BERT	8.2 B	Hugging Face
EN	ALBERT	16 GB	Hugging Face
ET	BERT	N/A	TurkuNLP
FR	RoBERTa	138 GB	Hugging Face
FI	BERT	24 B	Hugging Face
IT	BERT	13 GB	Hugging Face
NL	BERT	N/A	Hugging Face
PL	BERT	1.8 B	Hugging Face
SV	BERT	3 B	Hugging Face
BG	BERT	N/A	Hugging Face
CS	BERT	N/A	Hugging Face
SK	BERT	N/A	Hugging Face

Table 3: Language-specific transformer encoders to develop our models. The corpus column shows the corpus size used to pretrain each encoder (B: billion tokens, GB: gigabytes). BERT and RoBERTa adapt the base models whereas ALBERT adapts the large model. Publications and resource links are shown in Table 5.

are English, Finnish, French, and Italian, which either use more advanced encoding methods or their language models are trained on larger corpora.

For EULAS, the multilingual encoding approach still outperforms 8 out of the 12 languages as for ELAS. However, DTP models completely outperform both DGP and ENS models, indicating that the primary and secondary dependencies are not distinguishable by our current DGP approach, which complies with the fact that DGP is trained on enhanced relations rather than the basic dependencies DTP is trained on. We believe the performance of DGP could be improved through ad-hoc strategies to handle enhancement of case and lemma.

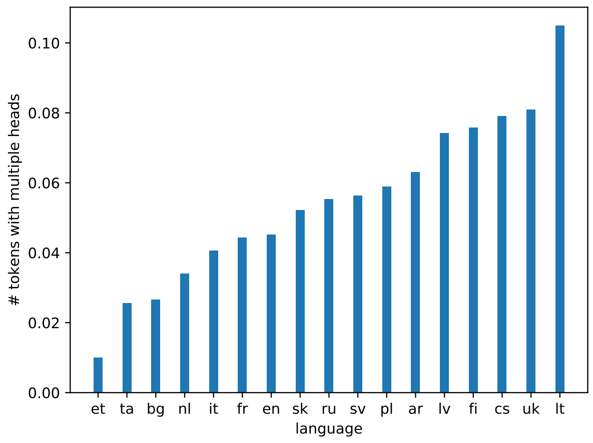

Figure 2: Percentages of tokens with multiple heads.

4 Analysis

This section analyzes factors that affect our models the most, common error made across languages and what possible improvement that can be made.

4.1 Data Size

We use the same hyper-parameters for all datasets, which may have led to possible overfitting (or underfitting). To verify this, we compute the differences between the ELAS scores of our models and that of the highest models from other teams. We then plot the differences as a function of the log training data size and fit the differences to a linear regression model shown in Figure 3.

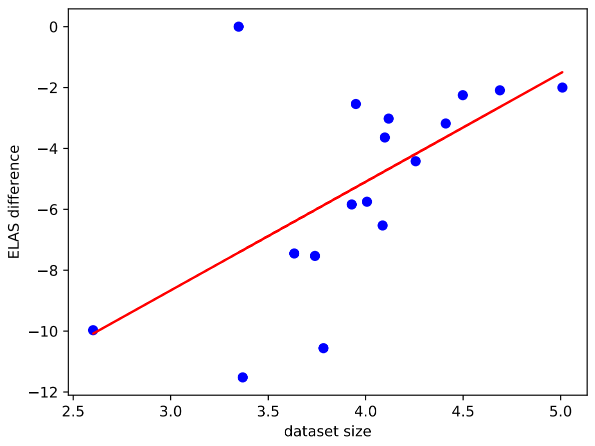

Figure 3: Difference in ELAS between our parser and the top team as a function of dataset size.

We also fit the differences with sentence scores as random effects to another regression model, finding that the p values for sentence scores and dataset sizes are 0.760 and 0.001. It shows that our system performs relatively better on larger datasets while overfits to smaller datasets, suggesting that decreasing model capacity may improve ELAS for languages with less training data.

4.2 OOV

To investigate the performance of each model on Out-Of-Vocabulary (OOV) tokens, we evaluate them on the OOV-only subset of English treebank. As shown in Figure 4, language specific encoder outperforms multilingual encoder in therse model settings, which is not surprising.

4.3 Sentence Length

We evaluate the ELAS of English treebank offline relative to sentence length with gold tokenization and sentence split. As shown in Figure 5, DTP models are very stable on long sentences while the performance of DGP models dramatically drops with the increase of sentence length. Performance of multilingual encoder models tends to drop faster than their language-specific counterparts.

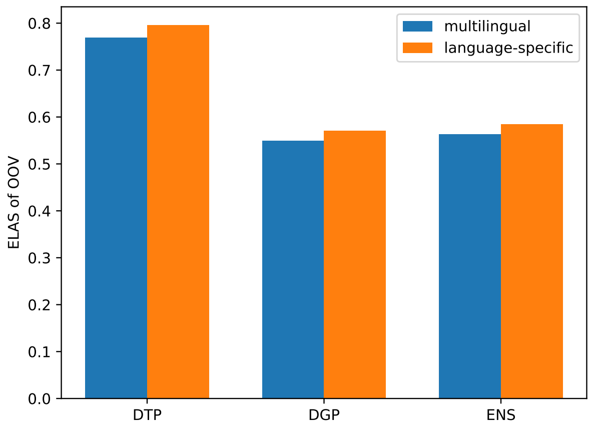

Figure 4: ELAS of Out-Of-Vocabulary tokens.

5 Related Work

Our work in utilizing multilingual Transformers as the encoder for parser model is most closely related to the UDify system (Kondratyuk and Straka, 2019). UDify is a multilingual multi-task model leveraging a multilingual BERT to accurately predict universal part-of-speech, morphological features, lemmas, and dependency trees simultaneously across 75 languages. UDify concatenates all training sets together to encourage knowledge transferring across languages, which benefits low-resource languages the most. In our multilingual BERT approach, each model is trained separately.

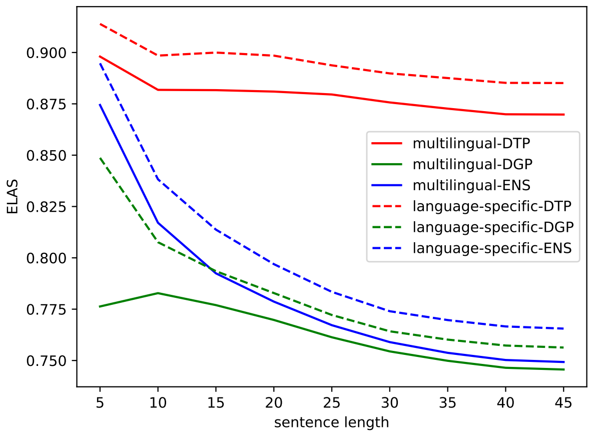

Figure 5: ELAS of the English treebank relative to sentence length.

For the encoder, pre-trained Transformers (Devlin et al., 2019; Liu et al., 2020; Lan et al., 2020) has been shown to be effective in tagging and parsing tasks without heavily engineered decoders(He and Choi, 2020). The encoder representations embed entire syntax trees according to the structural probe (Hewitt and Manning, 2019), encouraging the application of Transformers in parsing task. Not only in the embedding space, syntactic structures are used in the Tree Transformer (Wang et al., 2019), where constituent attention is gradually learned bottom-up layer by layer. Our parser employs pre-trained transformer models in Section 3.2.

For the decoder, the deep biaffine attention (Dozat and Manning, 2017) dominates the graph based approach since its establishment. The top ranked graph-based dependency parser at the CoNLL 2017 Shared Task (Dozat et al., 2017) adopts biaffine attention with rich character level features. With a parsing algorithm other than MST, the biaffine parser is successfully extended to semantic dependency parsing (Dozat and Manning, 2018). The current state-of-the-art dependency parsing records on English Penn Treebank (Marcus et al., 1993) and Chinese Treebank (Xue et al., 2005) are maintained by the Head-Driven Phrase Structure parser (Zhou and Zhao, 2019), which jointly learns constituency parsing and dependency parsing with layers including biaffine attention. Apart from parsing, biaffine attention has also been applied to graph related task including relation extraction (Nguyen and Verspoor, 2019) and coreference resolution (Zhang et al., 2018).

6 Conclusion

This paper describes our parsing approach to enhanced universal dependencies for the IWPT 2020 shared task. We find that the multilingual BERT encoder is able to parse various languages without language specific network design. Our proposed ensemble method is shown to be beneficial for the secondary dependency prediction.

In the future, we will improve the secondary dependency prediction in a more systematic way. We believe our current approach generating dependency graphs satisfying the tree constraint of primary dependencies can be further improved if the constraint can be applied to biaffine attention during training time, and the MSDAGs constraint can be relaxed for better performance. We leave these exciting topics for future work.

Acknowledgments

We gratefully acknowledge the support of the AWS Machine Learning Research Awards (MLRA). Any contents in this material are those of the authors and do not necessarily reflect the views of AWS.

References

Mariana SC Almeida and André FT Martins. 2015. Lisbon: Evaluating turbosemanticparser on multiple languages and out-of-domain data. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 970–973.

Mikhail Arkhipov, Maria Trofimova, Yuri Kuratov, and Alexey Sorokin. 2019. Tuning multilingual transformers for language-specific named entity recognition. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 89–93, Florence, Italy. Association for Computational Linguistics.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 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*. Association for Computational Linguistics.

Jinho D. Choi. 2017. Deep Dependency Graph Conversion in English. In *Proceedings of the 15th International Workshop on Treebanks and Linguistic Theories*, TLT'17, pages 35–62, Bloomington, IN.

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 *Proceedings of the 5th International Conference on Learning Representations*, ICLR'17.

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*, ACL'18, pages 484–490.

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

Yantao Du, Fan Zhang, Xun Zhang, Weiwei Sun, and Xiaojun Wan. 2015. Peking: Building Semantic Dependency Graphs with a Hybrid Parser. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval'15, pages 927–931.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Han He and Jinho D. Choi. 2020. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS'20.

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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, ICLR'15.

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.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{bert}a: A robustly optimized {bert} pretraining approach.

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

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. Camembert: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Dat Quoc Nguyen and Karin Verspoor. 2019. End-to-end neural relation extraction using deep biaffine attention. In *European Conference on Information Retrieval*, pages 729–738. Springer.

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

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.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926.

Natalie Schluter. 2014. On maximum spanning dag algorithms for semantic dag parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 61–65.

Sebastian Schuster, Matthew Lamm, and Christopher D Manning. 2017. Gapping constructions in universal dependencies v2. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 123–132.

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.

Bonggun Shin, Hao Yang, and Jinho D. Choi. 2019. The Pupil Has Become the Master: Teacher-Student Model-Based Word Embedding Distillation with Ensemble Learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pages 3439–3445.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree transformer: Integrating tree structures into self-attention. 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 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase Structure Annotation of a Large Corpus. *Natural Language Engineering*, 11(2):207–238.

Rui Zhang, Cícero Nogueira dos Santos, Michihiro Yasunaga, Bing Xiang, and Dragomir Radev. 2018. Neural coreference resolution with deep biaffine attention by joint mention detection and mention clustering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 102–107, Melbourne, Australia. Association for Computational Linguistics.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.

A Supplemental Materials

A.1 Hyperparameters

Table 4 shows the hyperparameters used to train models for all languages.

Transformer	
Max sequence length	256
Warm up steps	10%
Learning rate	$1e^{-5}$
End learning rate	0
Weight decay rate	0
Adam β_1	0.9
Adam β_2	0.999
Adam ϵ	$1e^{-6}$
Parser	
MLP$^{(arc)}$	500
MLP$^{(rel)}$	100
Clip norm	5
Learning rate	$1e^{-3}$
Adam β_1	0.9
Adam β_2	0.9
Adam ϵ	$1e^{-12}$
Anneal factor	0.75
Anneal every	5000
Dropout Rates	
Embeddings	33%
MLP	33%
Optimizer	
Batch size	≈ 150
Train epochs	1000

Table 4: Hyperparameters used for our experiments.

A.2 Language-Specific Encoders

Table 5 shows the authors and the sources of the language-specific transformer decoders used to develop our parsing models.

A.3 Parsing Results

Table 6 shows the parsing results using the 5 evaluation metrics, unlabeled attachment score (UAS), labeled attachment score (LAS), content labeled attachment score (CLAS), LAS on enhanced dependencies where labels are restricted to the UD relation (EULAS), and LAS on enhanced dependencies (ELAS).

Lang.	Source	Link
AR	Unknown	huggingface.co/asafaya/bert-base-arabic
EN	Lan et al. (2020)	huggingface.co/albert-xxlarge-v2
ET	Unknown	dl.turkunlp.org/estonian-bert/etwiki-bert/pytorch/
FI	Virtanen et al. (2019)	huggingface.co/TurkuNLP/bert-base-finnish-cased-v1
FR	Liu et al. (2020); Martin et al. (2020)	huggingface.co/dbmdz/bert-base-italian-cased
IT	Unknown	huggingface.co/dbmdz/bert-base-italian-cased
NL	Unknown	huggingface.co/wietsedv/bert-base-dutch-cased
PL	Unknown	huggingface.co/dkleczek/bert-base-polish-uncased-v1
SV	Unknown	huggingface.co/KB/bert-base-swedish-cased
BG	Arkhipov et al. (2019)	huggingface.co/DeepPavlov/bert-base-bg-cs-pl-ru-cased
CS	Arkhipov et al. (2019)	huggingface.co/DeepPavlov/bert-base-bg-cs-pl-ru-cased
SK	Arkhipov et al. (2019)	huggingface.co/DeepPavlov/bert-base-bg-cs-pl-ru-cased

Table 5: The authors and sources of the language specific transformer encoders used to develop our models.

		Multilingual Encoder					**Language-Specific Encoder**					**Rank**
		UAS	**LAS**	**CLAS**	**EULAS**	**ELAS**	**UAS**	**LAS**	**CLAS**	**EULAS**	**ELAS**	
AR	DTP	**79.05**	**74.75**	**71.76**	**71.83**	61.28	67.06	58.80	51.77	56.69	49.38	
	DGP	73.51	69.50	65.28	65.49	63.56	59.84	51.99	43.88	46.75	43.71	3
	ENS	79.05	74.75	71.76	69.46	**67.26**	67.06	58.80	51.77	51.37	48.02	
BG	DTP	**94.37**	**91.77**	**89.19**	**89.61**	79.58	70.99	63.54	55.96	62.27	55.76	
	DGP	92.64	90.00	86.44	87.31	86.66	59.45	52.72	41.07	46.84	45.25	3
	ENS	94.37	91.77	89.19	88.84	**88.19**	70.99	63.54	55.96	54.01	52.16	
CS	DEP	**93.72**	**91.91**	**90.74**	**87.20**	78.37	86.83	83.58	80.84	79.53	71.73	
	SDP	85.49	83.81	78.52	80.38	79.38	75.95	72.96	66.06	69.79	68.47	3
	ENS	93.72	91.91	90.74	86.63	**85.51**	86.83	83.58	80.84	53.45	51.05	
EN	DTP	89.37	87.01	84.63	84.61	75.70	**90.94**	**88.70**	**87.01**	**86.24**	76.99	
	DGP	87.17	84.79	81.20	82.74	82.31	87.62	85.43	81.82	83.67	83.22	3
	ENS	89.37	87.01	84.63	83.68	83.24	90.94	88.70	87.01	85.76	**85.30**	
ET	DTP	**87.35**	**84.25**	**82.58**	**82.81**	68.08	62.39	54.23	50.25	53.31	44.61	
	DGP	80.73	78.11	73.93	76.42	75.94	52.39	45.98	39.66	44.62	43.90	2
	ENS	87.35	84.25	82.58	81.98	**81.36**	62.39	54.23	50.25	52.74	51.82	
FI	DEP	91.10	88.95	87.37	85.43	70.37	**93.32**	**91.84**	**90.90**	**88.19**	72.40	
	SDP	78.61	76.71	71.06	72.62	72.03	85.89	84.29	80.89	80.10	79.38	3
	ENS	91.10	88.95	87.37	81.44	80.54	93.32	91.84	90.90	83.70	**82.96**	
FR	DTP	92.32	88.49	84.27	87.47	85.29	**92.52**	**89.33**	**85.53**	**88.28**	**86.23**	
	DGP	82.39	79.07	67.42	75.49	74.35	87.46	84.70	77.91	80.12	78.87	1
	ENS	92.32	88.49	84.27	83.34	81.97	92.52	89.33	85.53	82.83	81.45	
IT	DTP	94.96	93.32	89.88	90.49	75.21	**95.03**	**93.66**	**90.67**	**90.89**	75.50	
	DGP	92.33	90.70	85.30	86.98	86.46	91.38	90.13	83.67	86.92	86.45	4
	ENS	94.96	93.32	89.88	88.35	87.83	95.03	93.66	90.67	89.04	**88.52**	
LT	DTP	**81.97**	**77.63**	**75.27**	**73.55**	63.91	-	-	-	-	-	
	DGP	72.23	68.60	63.21	63.10	61.59	-	-	-	-	-	4
	ENS	81.97	77.63	75.27	68.24	**66.12**	-	-	-	-	-	
LV	DTP	**89.07**	**85.98**	**83.79**	**81.17**	71.68	-	-	-	-	-	
	DGP	78.35	75.92	69.89	71.95	71.58	-	-	-	-	-	3
	ENS	89.07	85.98	83.79	79.66	**79.19**	-	-	-	-	-	
NL	DTP	**88.75**	**86.29**	**81.18**	**83.67**	71.08	88.46	86.02	80.87	83.47	70.95	
	DGP	83.18	80.98	72.41	77.37	76.94	82.80	80.60	72.31	76.97	76.46	3
	ENS	88.75	86.29	81.18	81.21	**80.72**	88.46	86.02	80.87	80.59	80.02	
PL	DTP	**94.27**	**91.88**	**90.35**	**88.29**	74.33	76.45	70.03	64.65	67.58	57.35	
	DGP	80.23	77.88	73.53	74.54	70.39	65.67	60.41	51.18	55.58	51.90	2
	ENS	94.27	91.88	90.35	86.79	**82.39**	76.45	70.03	64.65	63.82	59.59	
RU	DTP	**94.20**	**92.87**	**91.71**	**89.71**	74.62	82.31	78.43	74.97	76.05	63.51	
	DGP	87.18	86.13	81.47	83.47	83.19	71.19	68.04	60.93	64.16	63.44	3
	ENS	94.20	92.87	91.71	88.93	**88.60**	82.31	78.43	74.97	72.13	71.19	
SK	DTP	**92.64**	**90.61**	**89.29**	**86.83**	70.12	46.19	35.72	27.97	34.50	30.41	
	DGP	89.91	87.78	85.48	83.39	81.37	38.37	29.66	20.49	26.82	26.03	3
	ENS	92.64	90.61	89.29	84.73	**82.72**	46.19	35.72	27.97	31.19	30.08	
SV	DTP	**88.29**	**85.23**	**83.63**	**81.72**	70.92	-	-	-	-	-	
	DGP	85.53	82.41	79.58	78.32	77.39	-	-	-	-	-	4
	ENS	88.29	85.23	83.63	79.11	**78.19**	-	-	-	-	-	
TA	DTP	**65.57**	**58.69**	**54.73**	**58.66**	**54.26**	-	-	-	-	-	
	DGP	50.95	45.54	40.35	41.36	40.10	-	-	-	-	-	3
	ENS	65.57	58.69	54.73	48.50	46.67	-	-	-	-	-	
UK	DTP	**91.01**	**88.91**	**86.45**	**85.36**	77.04	-	-	-	-	-	
	DGP	88.50	86.30	82.93	80.18	79.56	-	-	-	-	-	3
	ENS	91.01	88.91	86.45	80.34	**79.69**	-	-	-	-	-	
AVG	DTP	**88.71**	**85.80**	**83.34**	**82.85**	71.87	56.03	52.58	49.49	51.00	44.40	
	DGP	81.70	79.07	74.00	75.36	74.28	40.95	38.22	33.70	36.03	35.25	3
	ENS	88.71	85.80	83.34	80.07	**78.83**	50.92	47.66	44.74	43.95	43.01	

Table 6: Parsing results on the test sets evaluated by the 5 metrics, UAS, LAS, CLAS, EULAS, and ELAS. The Rank column indicates the ranking of our best model for the corresponding language. AR: Arabic, BG: Bulgarian, CS: Czech, EN: English, ET: Estonian, FI: Finnish, FR: French, IT: Italian, NL: Dutch, LT: Lithuanian, LV: Latvian, PL: Polish, RU: Russian, SK: Slovak, SV: Swedish, TA: Tamil, UK: Ukrainian.