TGIF: Tree-Graph Integrated-Format Parser for Enhanced UD with Two-Stage Generic- to Individual-Language Finetuning

Tianze Shi
Cornell University
tianze@cs.cornell.edu

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

We present our contribution to the IWPT 2021 shared task on parsing into enhanced Universal Dependencies. Our main system component is a hybrid tree-graph parser that integrates (a) predictions of spanning trees for the enhanced graphs with (b) additional graph edges not present in the spanning trees. We also adopt a finetuning strategy where we first train a language-generic parser on the concatenation of data from all available languages, and then, in a second step, finetune on each individual language separately. Additionally, we develop our own complete set of pre-processing modules relevant to the shared task, including tokenization, sentence segmentation, and multi-word token expansion, based on pre-trained XLM-R models and our own pre-training of character-level language models. Our submission reaches a macro-average ELAS of 89.24 on the test set. It ranks top among all teams, with a margin of more than 2 absolute ELAS over the next best-performing submission, and best score on 16 out of 17 languages.

1 Introduction

The Universal Dependencies (UD; Nivre et al., 2016, 2020) initiative aims to provide cross-linguistically consistent annotations for dependency-based syntactic analysis, and includes a large collection of treebanks (202 for 114 languages in UD 2.8). Progress on the UD parsing problem has been steady (Zeman et al., 2017, 2018), but existing approaches mostly focus on parsing into *basic* UD trees, where bilexical dependency relations among surface words must form single-rooted trees. While these trees indeed contain rich syntactic information, the adherence to tree representations can be insufficient for certain constructions including coordination, gapping, relative clauses, and argument sharing through control and raising (Schuster and Manning, 2016).

The IWPT 2020 (Bouma et al., 2020) and 2021 (Bouma et al., 2021) shared tasks focus on parsing into *enhanced* UD format, where the representation is connected graphs, rather than rooted trees. The extension from trees to graphs allows direct treatment of a wider range of syntactic phenomena, but it also poses a research challenge: how to design parsers suitable for such enhanced UD graphs.

To address this setting, we propose to use a tree-graph hybrid parser leveraging the following key observation: since an enhanced UD graph must be connected, it must contain a spanning tree as a subgraph. These spanning trees may differ from basic UD trees, but still allow us to use existing techniques developed for dependency parsing, including applying algorithms for finding maximum spanning trees to serve as accurate global decoders. Any additional dependency relations in the enhanced graphs not appearing in the spanning trees are then predicted on a per-edge basis. We find that this tree-graph hybrid approach results in more accurate predictions compared to a dependency graph parser that is combined with postprocessing steps to fix any graph connectivity issues.

Besides the enhanced graphs, the shared task setting poses two additional challenges. Firstly, the evaluation is on 17 languages from 4 language families, and not all the languages have large collections of annotated data: the lowest-resource language, Tamil, contains merely 400 training sentences — more than two magnitudes smaller than what is available for Czech. To facilitate knowledge sharing between high-resource and low-resource languages, we develop a two-stage finetuning strategy: we first train a language-generic model on the concatenation of all available training treebanks from all languages provided by the shared task, and then finetune on each language individually.

Secondly, the shared task demands parsing from raw text. This requires accurate text processing pipelines including modules for tokenization, sentence splitting, and multi-word token expansion, in addition to enhanced UD parsing. We build our own models for all these components; notably, we pre-train character-level masked language models on Wikipedia data, leading to improvements on tokenization, the first component in the text processing pipeline. Our multi-word token expanders combine the strengths of pre-trained learning-based models and rule-based approaches, and achieve robust results, especially on low-resource languages.

Our system submission integrates the aforementioned solutions to the three main challenges given by the shared task, and ranks top among all submissions, with a macro-average EULAS of 90.16 and ELAS of 89.24. Our system gives the best evaluation scores on all languages except for Arabic, and has large margins (more than 5 absolute ELAS) over the second-best systems on Tamil and Lithuanian, which are among languages with the smallest training treebanks.

2 TGIF: Tree-Graph Integrated-Format Parser for Enhanced UD

2.1 Tree and Graph Representations for Enhanced UD

The basic syntactic layer in UD is a single-rooted labeled dependency tree for each sentence, whereas the enhanced UD layer only requires that the set of dependency edges for each sentence form a connected graph. In these connected graphs, each word may have multiple parents, there may be multiple roots for a sentence, and the graphs may contain cycles, but there must exist one path from at least one of the roots to each node.[1]

Accompanying the increase in expressiveness of the enhanced UD representation is the challenge to produce structures that correctly satisfy graph-connectivity constraints during model inference. We summarize the existing solutions proposed for the previous run of the shared task at IWPT 2020 (Bouma et al., 2020) into four main categories:

• *Tree-based*: since the overlap between the enhanced UD graphs and the basic UD trees are typically significant, and any deviations tend to be localized and tied to one of several certain syntactic constructions (e.g, argument sharing in a control structure), one can repurpose tree-based parsers for producing enhanced UD graphs. This category of approaches include packing the additional edges from an enhanced graph into the basic tree (Kanerva et al., 2020) and using either rule-based or learning-based approaches to convert a basic UD tree into an enhanced UD graph (Heinecke, 2020; Dehouck et al., 2020; Attardi et al., 2020; Ek and Bernardy, 2020).[2]

• *Graph-based*: alternatively, one can directly focus on the enhanced UD graph with a semantic dependency graph parser that predicts the existence and label of each candidate dependency edge. But there is generally no guarantee that the set of predicted edges will form a connected graph, so a post-processing step is typically employed to fix any connectivity issues. This category of approaches includes the work of Wang et al. (2020), Barry et al. (2020), and Grünewald and Friedrich (2020).[3]

• *Transition-based*: Hershcovich et al. (2020) adapt a transition-based solution. Their system explicitly handles empty nodes through a specialized transition for inserting them; it relies on additional post-processing to ensure connectivity.

• *Tree-Graph Integrated*: He and Choi (2020) integrate a tree parser and a graph parser,[4] where the tree parser produces the basic UD tree, and the graph parser predicts any additional edges. During inference, all nodes are automatically connected through the tree parser, and the graph parser allows flexibility in producing graph structures.[5]

The tree-based approaches are prone to error propagation, since the predictions of the enhanced layer rely heavily on the accuracy of basic UD tree parsing. The graph-based and transition-based approaches natively produce graph structures, but they require post-processing to ensure connectivity. Our system is a tree-graph integrated-format parser that combines the strengths of the available global inference algorithms for tree parsing and the flexibility of a graph parser, without the need to use post-processing to fix connectivity issues.

[1]Enhanced UD graphs additionally allow insertion of phonologically-empty nodes to recover elided elements in gapping constructions. This is currently beyond the scope our system and we use pre- and post-processing collapsing steps to handle empty nodes (§5).

[2]The same idea has also been applied to the task of conjunction propagation prediction (e.g., Grünewald et al., 2021).

[3]Barry et al.'s (2020) parsers use basic UD trees as features, but the output space is not restricted by the basic trees.

[4]He and Choi (2020) describe their combo as an "ensemble" but we prefer the term "integration" for both their method and ours (which is inspired by theirs), since the two components are not, strictly speaking, targeting same structures.

[5]The main difference from the tree-based approaches is that the search space for additional graph edges is unaffected by the predictions of basic UD trees in an integrated approach.

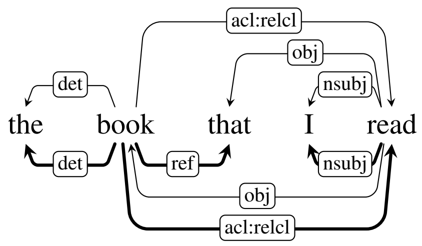

Figure 1: An example with basic UD and enhanced UD annotations above and below the text respectively. The extracted spanning tree (§2.2) is bolded and is different from the basic UD tree.

2.2 Spanning Tree Extraction

A connected graph must contain a spanning tree, and conversely, if we first predict a spanning tree over all nodes, and subsequently add additional edges, then the resulting graph remains connected. Indeed, this property is leveraged in some previously-proposed connectivity post-processing steps (e.g., Wang et al., 2020), but extracting a spanning tree based on scores from graph-prediction models creates a mismatch between training and inference. He and Choi (2020) instead train tree parsers and graph parsers separately and combine their prediction during inference, but their tree parsers are trained on basic UD trees whose edges are not always present in the enhanced UD layer.

Our solution refines He and Choi's (2020) approach: we train tree parsers to predict spanning trees extracted from the enhanced UD graphs, instead of basic UD trees, to minimize train-test mismatch. See Figure 1 for an example. Spanning tree extraction is in essence assignment of unique head nodes to all nodes in a graph, subject to tree constraints. For consistent extraction, we apply the following rules:

• If a node has a unique head in the enhanced graph, there is no ambiguity in head assignment.

• If a basic UD edge is present among the set of incoming edges to a given node, include that basic UD edge in the spanning tree.

• Otherwise, there must be multiple incoming edges, none of which are present in the basic UD tree. We pick the parent node that is the "highest", i.e., the closest to the root node, in the basic tree.

The above head assignment steps do not formally guarantee that the extracted structures will be trees, but empirically, we observe that the extraction results are indeed trees for all training sentences.[6]

[6]Dear Reviewer 1: your question here in the submitted paper caused us to uncover a bug! Fixing it rectified the 4 training sentences that weren't originally getting trees.

2.3 Parameterization

Our parser architecture is adapted from that of Dozat and Manning (2017, 2018), which forms the basis for the prior graph-based approaches in the IWPT 2020 shared task. We predict unlabeled edges and labels separately, and for the unlabeled edges, we use a combination of a tree parser and a graph-edge prediction module.

Representation The first step is to extract contextual representations. For this purpose, we use the pre-trained `XLM-R` model (Conneau et al., 2020), which is trained on multilingual CommonCrawl data and supports all 17 languages in the shared task. The `XLM-R` feature extractor is finetuned along with model training. Given a length-n input sentence $x = x_1, \ldots, x_n$ and layer l, we extract

$$[\mathbf{x}_0^l, \mathbf{x}_1^l, \ldots, \mathbf{x}_n^l] = \texttt{XLM-R}^l(\texttt{<s>}, x_1, \ldots, x_n, \texttt{</s>}),$$

where inputs to the `XLM-R` model are a concatenated sequence of word pieces from each UD word, we denote the layer-l vector corresponding to the last word piece in the word x_i as $\mathbf{x}_i^l$, and the dummy root representations $\mathbf{x}_0$s are taken from the special `<s>` token at the beginning of the sequence.

Deep Biaffine Function All our parsing components use deep biaffine functions (DBFs), which score the interactions between pairs of words:

$$\begin{aligned}\text{DBF}(i,j) =& \mathbf{v}_i^{\text{head}\top} U \mathbf{v}_j^{\text{mod}} + \mathbf{b}^{\text{head}} \cdot \mathbf{v}_i^{\text{head}} \\ &+ \mathbf{b}^{\text{mod}} \cdot \mathbf{v}_j^{\text{mod}} + b,\end{aligned}$$

where $\mathbf{v}_i^{\text{head}}$ and $\mathbf{v}_j^{\text{mod}}$ are non-linearly transformed vectors from weighted average `XLM-R` vectors across different layers:

$$\mathbf{v}_i^{\text{head}} = \text{ReLU}\left(W^{\text{head}} \sum_l \frac{e^{\alpha_l^{\text{head}}}}{\sum_{l'} e^{\alpha_{l'}^{\text{head}}}} \mathbf{x}_i^l \right),$$

and $\mathbf{v}_j^{\text{mod}}$ is defined similarly. Each DBF has its own trainable weight matrices U, W^{head}, and W^{mod}, vectors $\mathbf{b}^{\text{head}}$ and $\mathbf{b}^{\text{mod}}$, and scalars b, $\{\alpha_l^{\text{head}}\}$ and $\{\alpha_l^{\text{mod}}\}$.

Tree Parser To estimate the probabilities of head attachment for each token w_j, we define

$$P(\text{head}(w_j) = w_i) = \text{softmax}_i(\text{DBF}^{\text{tree}}(i,j)).$$

The tree parsing models are trained with cross-entropy loss, and we use a non-projective maximum spanning tree algorithm (Chu and Liu, 1965; Edmonds, 1967) for global inference.

Language	Direct Training		Generic	Finetuned
	Graph+Fix	Tree-Graph		
Arabic	80.34	80.30	80.57	**80.63**
Bulgarian	91.81	92.00	91.69	**92.30**
Czech	92.93	**92.98**	92.94	**92.98**
Dutch	92.14	92.13	92.03	**92.21**
English	88.38	88.51	88.44	**88.83**
Estonian	**89.53**	89.42	89.22	89.40
Finnish	91.97	92.10	91.84	**92.48**
French	94.46	94.51	94.26	**95.52**
Italian	93.04	93.24	93.26	**93.41**
Latvian	88.47	88.42	88.38	**89.78**
Lithuanian	90.57	90.63	90.47	**90.85**
Polish	91.28	91.48	91.28	**91.63**
Russian	93.47	**93.50**	93.37	93.47
Slovak	93.70	93.83	94.00	**95.44**
Swedish	90.35	90.48	90.33	**91.57**
Tamil	66.24	66.82	67.35	**68.95**
Ukrainian	92.98	92.94	93.24	**93.89**
Average	89.51	89.61	89.57	**90.20**

Table 1: Dev-set ELAS (%) results, comparing graph parsers with connectivity-fixing postprocessing against tree-graph integrated models (§2) and comparing parsers trained directly on each language, generic-language parsers, and parsers finetuned on individual languages from the generic-language checkpoint (§3).

Graph Parser In addition to the spanning trees, we make independent predictions on the existence of any extra edges in the enhanced UD graphs by

$$P(\exists\text{edge } w_i \to w_j) = \text{sigmoid}(\text{DBF}^{\text{graph}}(i, j)).$$

We train the graph parsing model with a cross entropy objective, and during inference, any edges with probabilities ≥ 0.5 are included in the outputs.

Relation Labeler For each edge in the unlabeled graph, we predict the relation label via

$$P(\text{lbl}(w_i \to w_j) = r) = \text{softmax}_r(\text{DBF}^{\text{rel-}r}(i, j)),$$

where we have as many deep biaffine functions as the number of candidate relation labels in the data. To reduce the large number of potential labels due to lexicalization, the relation labeler operates on a de-lexicalized version of the labels, and then a re-lexicalization step expands the predicted labels into their full forms (§5).

Training The above three components are separately parameterized, and during training, we optimize for the sum of their corresponding cross-entropy loss functions.

2.4 Empirical Comparisons

In Table 1, we compare our tree-graph integrated-format parser with a fully graph-based approach. The graph-based baseline uses the same feature extractor, graph parser, and relation labeler modules, but it omits the tree parser for producing spanning trees, and we apply post-processing steps to ensure connectivity of the output graphs. Our tree-graph integrated-format parser outperforms the graph-based baseline on 12 out of the 17 test languages (binomial test, $p = 0.07$).

3 TGIF: Two-Stage Generic- to Individual-Language Finetuning

In addition to the tree-graph integration approach, our system submission also features a two-stage finetuning strategy. We first train a language-generic model on the concatenation of all available training treebanks in the shared task data regardless of their source languages, and then finetune on each individual language in a second step.

This two-stage finetuning strategy is designed to encourage knowledge sharing across different languages, especially from high-resource languages to lower-resource ones. In our experiment results as reported in Table 1, we find that this strategy is indeed beneficial for the majority of languages, especially those with small training corpora (e.g., 2.13 and 1.01 absolute ELAS improvements on Tamil and French respectively), though this comes at the price of slightly decreased accuracies on high-resource languages (e.g., -0.02 on Estonian and -0.03 on Russian). Additionally, we find that the language-generic model achieves reasonably competitive performance when compared with the set of models directly trained on each individual language. This suggests that practitioners may opt to use a single model for parsing all languages if there is a need to lower disk and memory footprints, without much loss in accuracy.

4 Pre-TGIF: Pre-Training Grants Improvements Full-Stack

Inspired by the recent success of pre-trained language models on a wide range of NLP tasks (Peters et al., 2018; Devlin et al., 2019; Conneau et al., 2020, *inter alia*), we build our own text processing pipeline based on pre-trained language models. Due to limited time and resources, we only focus on components relevant to the shared task, which include tokenization, sentence splitting, and multi-word token (MWT) expansion.

4.1 Tokenizers with Character-Level Masked Language Model Pre-Training

We follow state-of-the-art strategies (Qi et al., 2020; Nguyen et al., 2021) for tokenization and model the task as a tagging problem on sequences of characters. But in contrast to prior methods where tokenization and sentence segmentation are bundled into the same prediction stage, we tackle tokenization in isolation, and for each character, we make a binary prediction as to whether a token ends at the current character position or not.

An innovation in our tokenization is that we fine-tune character-based language models trained on Wikipedia data. In contrast, existing approaches typically use randomly-initialized models (Qi et al., 2020) or use pre-trained models on subword units instead of characters (Nguyen et al., 2021).

We follow Devlin et al. (2019) and pre-train our character-level sequence models using a masked language modeling objective: during training, we randomly replace 15% of the characters with a special mask symbol and the models are trained to predict the identity of those characters in the original texts. Due to computational resource constraints, we adopt a small-sized architecture based on simple recurrent units (Lei et al., 2018).[7] We pre-train our models on Wikipedia data[8] and each model takes roughly 2 days to complete 500k optimization steps on a single GTX 2080Ti GPU.

4.2 Sentence Splitters

We split texts into sentences from sequences of tokens instead of characters (Qi et al., 2020). Our approach resembles that of Nguyen et al. (2021).[9] This allows our models to condense information from a wider range of contexts while still reading the same number of input symbols. The sentence splitters are trained to make binary predictions at each token position on whether a sentence ends there. We adopt the same two-stage finetuning strategy as for our parsing modules based on pre-trained `XLM-R` feature extractors (§3).

[7]Simple recurrent units are a fast variant of recurrent neural networks. In our preliminary experiments, they result in lower accuracies than long-short term memory networks (LSTMs), but are 2-5 times faster, depending on sequence lengths.

[8]We extract Wikipedia texts using WikiExtractor (Attardi, 2015) from Wikipedia dumps dated 2021-04-01.

[9]An important difference is that our sentence splitters are aware of token boundaries and the models are restricted from making token-internal sentence splitting decisions.

4.3 Multi-Word Token (MWT) Expanders

The UD annotations distinguish between tokens and words. A word corresponds to a consecutive sequence of characters in the surface raw text and may contain one or more syntactically-functioning words. We break down the MWT expansion task into first deciding whether or not to expand a given token and then performing the actual expansion. For the former, we train models to make a binary prediction on each token, and we use pre-trained `XLM-R` models as our feature extractors.

For the MWT expansion step once the tokens are identified through our classifiers, we use a combination of lexicon- and rule-based approaches. If the token form is seen in the training data, we adopt the most frequently used way to split it into multiple words. Otherwise, we invoke a set of language-specific handwritten rules developed from and tuned on the training data; a typical rule iteratively splits off an identified prefix or suffix from the remainder of the token.

4.4 Lemmatizers

While the shared task requires lemmatized forms for constructing the lexicalized enhanced UD labels, we only need to predict lemmas for a small percentage of words. Empirically, these words tend to be function words and have a unique lemma per word type. Thus, we use a full lexicon-based approach to (incomplete) lemmatization. Whenever a lemma is needed during the label re-lexicalization step, we look the word up in a dictionary extracted from the training data.

4.5 Evaluation

We compare our text-processing pipeline components with two state-of-the-art toolkits, Stanza (Qi et al., 2020) and Trankit (Nguyen et al., 2021) in Table 2. We train our models per-language instead of per-treebank to accommodate the shared task setting, so our models are at a disadvantage when there are multiple training treebanks for a language that have different tokenization/sentence splitting conventions (e.g., English-EWT and English-GUM handle word contractions differently). Despite this, our models are highly competitive in terms of tokenization and MWT expansion, and we achieve significantly better sentence segmentation results across most treebanks. We hypothesize that a sequence-to-sequence MWT expansion approach, similar to the ones underlying Stanza and Trankit,

Treebank		Token			Sentence			Word	
	Stanza	Trankit	Ours	Stanza	Trankit	Ours	Stanza	Trankit	Ours
Arabic-PADT	99.98	99.95	**99.99**	80.43	96.79	**96.87**	97.88	**99.39**	98.70
Bulgarian-BTB	99.93	99.78	**99.95**	97.27	98.79	**99.06**	99.93	99.78	**99.95**
Czech-FicTree	99.97	**99.98**	99.97	98.60	99.50	**99.54**	99.96	**99.98**	99.96
Czech-CAC	**99.99**	**99.99**	**99.99**	**100.00**	**100.00**	**100.00**	99.97	99.98	**99.99**
Dutch-Alpino	**99.96**	99.43	99.86	89.98	90.65	**94.45**	**99.96**	99.43	99.86
Dutch-LassySmall	99.90	99.36	**99.94**	77.95	92.60	**94.23**	99.90	99.36	**99.94**
English-EWT	**99.01**	98.67	98.79	81.13	90.49	**92.70**	**99.01**	98.67	98.79
English-GUM	**99.82**	99.52	98.88	86.35	91.60	**95.11**	**99.82**	99.52	99.18
Estonian-EDT	**99.96**	99.75	99.95	93.32	96.58	**96.60**	**99.96**	99.75	99.95
Estonian-EWT	**99.20**	97.76	98.72	67.14	82.58	**89.37**	**99.20**	97.76	98.72
Finnish-TDT	**99.77**	99.71	99.76	93.05	97.22	**98.26**	**99.73**	99.72	**99.73**
French-Sequoia	**99.90**	99.81	99.88	88.79	94.07	**96.82**	99.58	99.78	**99.84**
Italian-ISDT	**99.91**	99.88	99.90	98.76	**99.07**	**99.07**	99.76	**99.86**	99.83
Latvian-LVTB	**99.82**	99.73	99.80	99.01	98.69	**99.26**	**99.82**	99.73	99.80
Lithuanian-ALKSNIS	99.87	99.84	**99.99**	88.79	95.72	**96.22**	99.87	99.84	**99.99**
Polish-LFG	**99.95**	98.34	99.84	99.83	99.57	**99.88**	**99.95**	98.34	99.89
Polish-PDB	99.87	**99.93**	99.49	98.39	98.71	**99.66**	99.83	**99.92**	99.84
Russian-SynTagRus	99.57	99.71	**99.73**	98.86	99.45	**99.54**	99.57	99.71	**99.73**
Slovak-SNK	**99.97**	99.94	99.95	90.93	**98.49**	96.72	**99.97**	99.94	99.94
Swedish-Talbanken	**99.97**	99.91	**99.97**	98.85	99.26	**99.34**	**99.97**	99.91	**99.97**
Tamil-TTB	99.58	98.33	**99.63**	95.08	**100.00**	**100.00**	91.42	94.44	**95.34**
Ukrainian-IU	99.81	99.77	**99.86**	96.65	97.55	**98.38**	99.79	99.76	**99.84**

Table 2: Test-set F1 scores for tokenization, sentence segmentation, and MWT expansion, comparing Stanza (Qi et al., 2020), Trankit (Nguyen et al., 2021), and our system submission. Our system results are from the shared task official evaluations; Stanza and Trankit results are reported in the Trankit documentation with models trained on UD 2.5. *Caveat*: the results may not be strictly comparable due to treebank version mismatch.

may provide further gains to morphologically-rich languages that cannot be sufficiently modeled via handwritten rules, notably Arabic.

5 Other Technical Notes

Hyperparameters We report our hyperparameters in the Appendix.

Empty nodes Enhanced UD graphs may contain empty nodes in addition to the words in the surface form. Our parser does not support empty nodes, so we follow the official evaluation practice and collapse relation paths with empty nodes into composite relations during training and inference.

Multiple relations In some cases, there can be multiple relations between the same pair of words. We follow Wang et al. (2020) and merge all these relations into a composite label, and re-expand them during inference.

De-lexicalization and re-lexicalization Certain types of relation labels include lexicalized information, resulting in a large relation label set. For example, nmod:in contains a lemma "in" that is taken from the modifier with a case relation. To combat this, we follow Grünewald and Friedrich's (2020) strategy and replace the lemmas[10] with placeholders consisting of their corresponding relation labels. The previous example would result in a delexicalized label of nmod:[case]. During inference, we apply a re-lexicalization step to reconstruct the original full relation labels given our predicted graphs. We discard the lexicalized portions of the relation labels when errors occur either in de-lexicalization (unable to locate the source child labels to match the lemmas) or re-lexicalization (unable to find corresponding placeholder relations).

Sequence length limit Pre-trained language models typically have a limit on their input sequence lengths. The XLM-R model has a limit of 512 word pieces. For a small number of sentences longer than that, we discard word-internal word pieces, i.e., keep a prefix and a suffix of word pieces, of the longest words to fit within limit.

Multiple Treebanks Per Language Each language in the shared task can have one or more treebanks for training and/or testing. During evaluation, there is no explicit information regarding the source treebank of the piece of input text. Instead of handpicking a training treebank for each

[10]We find that using lemmas instead of word forms significantly improves coverage of the lexicalized labels.

Language	combo	dcu_epfl	fastparse	grew	nuig	robertnlp	shanghaitech	tgif (Ours)	unipi
Arabic	76.39	71.01	53.74	71.13	–	81.58	**82.26**	81.23	77.13
Bulgarian	86.67	92.44	78.73	88.83	78.45	93.16	92.52	**93.63**	90.84
Czech	89.08	89.93	72.85	87.66	–	90.21	91.78	**92.24**	88.73
Dutch	87.07	81.89	68.89	84.09	–	88.37	88.64	**91.78**	84.14
English	84.09	85.70	73.00	85.49	65.40	87.88	87.27	**88.19**	87.11
Estonian	84.02	84.35	60.05	78.19	54.03	86.55	86.66	**88.38**	81.27
Finnish	87.28	89.02	57.71	85.20	–	91.01	90.81	**91.75**	89.62
French	87.32	86.68	73.18	83.33	–	88.51	88.40	**91.63**	87.43
Italian	90.40	92.41	78.32	90.98	–	93.28	92.88	**93.31**	91.81
Latvian	84.57	86.96	66.43	77.45	56.67	88.82	89.17	**90.23**	83.01
Lithuanian	79.75	78.04	48.27	74.62	59.13	80.76	80.87	**86.06**	71.31
Polish	87.65	89.17	71.52	78.20	–	89.78	90.66	**91.46**	88.31
Russian	90.73	92.83	78.56	90.56	66.33	92.64	93.59	**94.01**	90.90
Slovak	87.04	89.59	64.28	86.92	67.45	89.66	90.25	**94.96**	86.05
Swedish	83.20	85.20	67.26	81.54	63.12	88.03	86.62	**89.90**	84.91
Tamil	52.27	39.32	42.53	58.69	–	59.33	58.94	**65.58**	51.73
Ukrainian	86.92	86.09	63.42	83.90	–	88.86	88.94	**92.78**	87.51
Average	83.79	83.57	65.81	81.58	–	86.97	87.07	**89.24**	83.64
Rank	4	6	8	7	9	3	2	1	5

Table 3: Official ELAS (%) evaluation results. Our submission ranks first on 16 out of the 17 languages.

language, we simple train and validate on the concatenation of all available data for each language.

Training on a single GPU The XLM-R model has large number of parameters, which makes it challenging to finetune on a single GPU. We use a batch size of 1 and accumulate gradients across multiple batches to lower the usage of GPU RAM. When this strategy alone is insufficient, e.g., when training the language-generic model, we additionally freeze the initial embedding layer of the model.

6 Official Evaluation

The shared task performs evaluation on UD treebanks that have enhanced UD annotations across 17 languages: Arabic (Hajič et al., 2009), Bulgarian (Simov et al., 2004), Czech (Hladká et al., 2010; Bejček et al., 2013; Jelínek, 2017), Dutch (van der Beek et al., 2002; Bouma and van Noord, 2017), English (Silveira et al., 2014; Zeldes, 2017), Estonian (Muischnek et al., 2014, 2019), Finnish (Haverinen et al., 2014; Pyysalo et al., 2015), French (Candito et al., 2014; Seddah and Candito, 2016), Italian (Bosco et al., 2013), Latvian (Pretkalniņa et al., 2018), Lithuanian (Bielinskienė et al., 2016), Polish (Patejuk and Przepiórkowski, 2018; Wróblewska, 2018), Russian (Droganova et al., 2018), Slovak (Zeman, 2018), Swedish (Nivre and Megyesi, 2007), Tamil (Ramasamy and Žabokrtský, 2012), Ukrainian (Kotsyba et al., 2016), and multilingual parallel treebanks (Zeman et al., 2017).

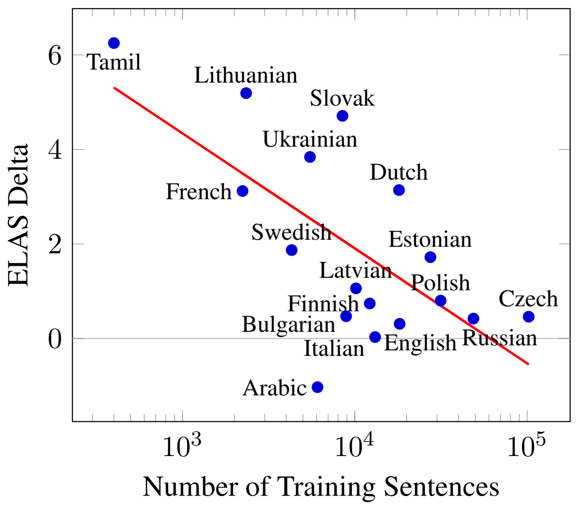

Figure 2: The per-language delta ELAS between our submission and the best performing system other than ours, as a function of (the log of the) number of training sentences. (For Italian, the difference is quite small but still positive.) Our models achieve larger improvements on lower-resource languages.

Table 3 shows the official ELAS evaluation results of all 9 participating systems in the shared task.[11] Our system has the top performance on 16 out of 17 languages, and it is also the best in terms of macro-average across all languages. On average, we outperform the second best system by a margin of more than 2 ELAS points in absolute terms, or more than 15% in relative error reduction.

Figure 2 visualizes the "delta ELAS" between

[11]Reproduced from https://universaldependencies.org/iwpt21/results.html.

our submission and the best result other than ours on a per-language basis, plotted against the training data size for each language. Our system sees larger improvements on lower-resource languages, where we have more than 5-point leads on Tamil and Lithuanian, two languages among those with the smallest number of training sentences.

7 Closing Remarks

Our submission to the IWPT 2021 shared task combines three main techniques: (1) tree-graph integrated-format parsing (graph $\rightarrow$ spanning tree $\rightarrow$ additional edges) (2) two-stage generic-to individual-language finetuning, and (3) preprocessing pipelines powered by language model pre-training. Each of the above contributes to our system performance positively,[12] and by combining all three techniques, our system achieves the best ELAS results on 16 out of 17 languages, as well as top macro-average across all languages, among all system submissions. Additionally, our system shows more relative strengths on lower-resource languages.

Due to time and resource constraints, our system adopts the same set of techniques across all languages and we train a single set of models for our primary submission. We leave it to future work to explore language-specific methods and/or model combination and ensemble techniques to further enhance model accuracies.

Acknowledgements We thank the anonymous reviewers for their constructive and detailed comments, and the task organizers for their flexibility regarding page limits. This work was supported in part by a Bloomberg Data Science Ph.D. Fellowship to Tianze Shi and a gift from Bloomberg to Lillian Lee.

References

Giuseppe Attardi, Daniele Sartiano, and Maria Simi. 2020. Linear neural parsing and hybrid enhancement for 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 206–214, Online. Association for Computational Linguistics.

Giusepppe Attardi. 2015. WikiExtractor. `https://github.com/attardi/wikiextractor`.

James Barry, Joachim Wagner, and Jennifer Foster. 2020. The ADAPT enhanced dependency parser at the IWPT 2020 shared task. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 227–235, Online. Association for Computational Linguistics.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague dependency treebank 3.0.

Agnė Bielinskienė, Loïc Boizou, Jolanta Kovalevskaitė, and Erika Rimkutė. 2016. Lithuanian dependency treebank ALKSNIS. *Human Language Technologies – The Baltic Perspective*, pages 107–114.

Cristina Bosco, Simonetta Montemagni, and Maria Simi. 2013. Converting Italian treebanks: Towards an Italian Stanford dependency treebank. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 61–69, Sofia, Bulgaria. 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.

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, Online. Association for Computational Linguistics.

Gosse Bouma and Gertjan van Noord. 2017. Increasing return on annotation investment: The automatic construction of a Universal Dependency treebank for Dutch. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 19–26, Gothenburg, Sweden. Association for Computational Linguistics.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric de la Clergerie. 2014. Deep Syntax Annotation of the Sequoia French treebank. In *Proceedings of*

[12]Comparing the 3 components: multilingual pre-training has a greater effect than the tree-graph parsing design. Sentence segmentation performance (SSP) doesn't necessarily translate to ELAS, so our SSP's large relative improvement at SS doesn't imply that SS is the biggest contributor to our system.

the Ninth International Conference on Language Resources and Evaluation (LREC-2014), pages 2298–2305, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Mathieu Dehouck, Mark Anderson, and Carlos Gómez-Rodríguez. 2020. Efficient EUD parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 192–205, Online. 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 *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France. 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.

Kira Droganova, Olga Lyashevskaya, and Daniel Zeman. 2018. Data conversion and consistency of monolingual corpora: Russian UD treebanks. In *Proceedings of the 17th International Workshop on Treebanks and Linguistic Theories (TLT 2018)*, Oslo, Norway. Linköping University Electronic Press.

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

Adam Ek and Jean-Philippe Bernardy. 2020. How much of enhanced UD is contained in UD? In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 221–226, Online. 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, Prisca Piccirilli, and Annemarie Friedrich. 2021. Coordinate constructions in English enhanced Universal Dependencies: Analysis and computational modeling. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 795–809, Online. Association for Computational Linguistics.

Jan Hajič, Otakar Smrž, Petr Zemánek, Petr Pajas, Jan Šnaidauf, Emanuel Beška, Jakub Kracmar, and Kamila Hassanová. 2009. Prague Arabic dependency treebank 1.0.

Katri Haverinen, Jenna Nyblom, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Anna Missilä, Stina Ojala, Tapio Salakoski, and Filip Ginter. 2014. Building the essential resources for Finnish: The Turku Dependency Treebank. *Language Resources and Evaluation*, 48(3):493–531.

Han He and Jinho D. Choi. 2020. Adaptation of multilingual transformer encoder for robust enhanced universal dependency parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 181–191, Online. Association for Computational Linguistics.

Johannes Heinecke. 2020. Hybrid enhanced Universal Dependencies parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 174–180, Online. Association for Computational Linguistics.

Daniel Hershcovich, Miryam de Lhoneux, Artur Kulmizev, Elham Pejhan, and Joakim Nivre. 2020. Køpsala: Transition-based graph parsing via efficient training and effective encoding. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 236–244, Online. Association for Computational Linguistics.

Barbora Vidová Hladká, Jan Hajič, Jiří Hana, Jaroslava Hlaváčová, Jiří Mírovský, and Jan Raab. 2010. The Czech academic corpus 2.0 guide. *The Prague Bulletin of Mathematical Linguistics*, 89(2008):41–96.

Tomáš Jelínek. 2017. FicTree: A manually annotated treebank of Czech fiction. In *Proceedings of the 17th Conference on Information Technologies - Applications and Theory*, pages 181–185, Martinské Hole, Slovakia.

Jenna Kanerva, Filip Ginter, and Sampo Pyysalo. 2020. Turku enhanced parser pipeline: From raw text to enhanced graphs in the IWPT 2020 shared task. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 162–173, Online. Association for Computational Linguistics.

Natalia Kotsyba, Bohdan Moskalevskyi, and Mykhailo Romanenko. 2016. Gold standard Universal Dependencies corpus for Ukrainian. https://github.com/UniversalDependencies/UD_Ukrainian-IU.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4470–4481, Brussels, Belgium. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *Proceedings of the Eighth International Conference on Learning Representations*, Online. OpenReview.net.

Kadri Muischnek, Kaili Müürisep, Tiina Puolakainen, Eleri Aedmaa, Riin Kirt, and Dage Särg. 2014. Estonian dependency treebank and its annotation scheme. In *Proceedings of the 13th Workshop on Treebanks and Linguistic Theories (TLT13)*, pages 285–291, Tübingen, Germany.

Kadri Muischnek, Kaili Müürisep, and Dage Särg. 2019. CG roots of UD treebank of Estonian web language. In *Proceedings of the NoDaLiDa 2019 Workshop on Constraint Grammar-Methods, Tools and Applications*, pages 23–26, Turku, Finland.

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

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

Joakim Nivre and Beata Megyesi. 2007. Bootstrapping a Swedish treebank using cross-corpus harmonization and annotation projection. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories*, pages 97–102, Bergen, Norway.

Agnieszka Patejuk and Adam Przepiórkowski. 2018. *From Lexical Functional Grammar to Enhanced Universal Dependencies: Linguistically Informed Treebanks of Polish*. Institute of Computer Science, Polish Academy of Sciences, Warsaw.

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

Lauma Pretkalniņa, Laura Rituma, and Baiba Saulīte. 2018. Deriving enhanced universal dependencies from a hybrid dependency-constituency treebank. In *Proceedings of the 21sh International Conference on Text, Speech, and Dialogue*, pages 95–105, Brno, Czech Republic. Springer.

Sampo Pyysalo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. Universal Dependencies for Finnish. In *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, pages 163–172, Vilnius, Lithuania. Linköping University Electronic Press, Sweden.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Loganathan Ramasamy and Zdeněk Žabokrtský. 2012. Prague dependency style treebank for Tamil. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 1888–1894, Istanbul, Turkey. European Language Resources Association.

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 2016)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association.

Djamé Seddah and Marie Candito. 2016. Hard time parsing questions: Building a QuestionBank for French. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2366–2370, Portorož, Slovenia. European Language Resources Association (ELRA).

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 2897–2904, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Kiril Simov, Petya Osenova, Alexander Simov, and Milen Kouylekov. 2004. Design and implementation of the Bulgarian HPSG-based treebank. *Research on Language and Computation*, 2(4):495–522.

Leonoor van der Beek, Gosse Bouma, Rob Malouf, and Gertjan Van Noord. 2002. The Alpino dependency treebank. In *Proceedings of Computational Linguistics in the Netherlands*, Twente, Netherlands.

Xinyu Wang, Yong Jiang, and Kewei Tu. 2020. Enhanced universal dependency parsing with second-order inference and mixture of training data. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 215–220, Online. Association for Computational Linguistics.

Alina Wróblewska. 2018. Extended and enhanced Polish dependency bank in Universal Dependencies format. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 173–182, Brussels, Belgium. Association for Computational Linguistics.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Daniel Zeman. 2018. Slovak dependency treebank in Universal Dependencies. *Jazykovedný casopis/Journal of Linguistics*, 68(2):385–395.

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

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj, and Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19, Vancouver, Canada. Association for Computational Linguistics.

A Hyperparameters

Character-level Language Model Pre-training	
Optimization:	
Optimizer	RAdam (Liu et al., 2020)
Batch size	128
Number of steps	500,000
Initial learning rate	3×10^{-4}
Weight decay	0.1
Gradient clipping	1.0
Simple Recurrent Units:	
Sequence length limit	512
Vocab size	512
Embedding size	256
Hidden size	256
Numer of layers	8
Dropout	0.3
Tokenizer	
Optimization:	
Optimizer	RAdam
Batch size	32
Initial learning rate	5×10^{-5}
Weight decay	0
Gradient clipping	1.0
Multi-layer Perceptrons (MLPs):	
Number of layers	1
Hidden size	500
Dropout	0.5
Sentence Splitter, MWT Expander, and Parser	
Pre-trained model	`XLM-R` (Large)
Optimization:	
Optimizer	RAdam
Batch size	8
Initial learning rate	1×10^{-5}
Second-stage learning rate	1×10^{-6}
Weight decay	0
Gradient clipping	1.0
Tagger MLPs (Sentence Splitter, MWT Expander):	
Number of layers	1
Hidden size	400
Dropout	0.5
Parser MLPs (Unlabeled Tree and Graph Parsers):	
Number of layers	1
Hidden size	383
Dropout	0.33
Parser MLPs (Relation Labeler):	
Number of layers	1
Hidden size	255
Dropout	0.33