Biaffine Dependency and Semantic Graph Parsing for Enhanced Universal Dependencies

Giuseppe Attardi, Daniele Sartiano, Maria Simi
Dipartmento di Informatica,
University of Pisa
{attardi, sartiano, simi}@di.unipi.it

Abstract

This paper presents the system used in our submission to the *IWPT 2021 Shared Task*. This year the official evaluation metrics was ELAS, therefore dependency parsing might have been avoided as well as other pipeline stages like POS tagging and lemmatization. We nevertheless chose to deploy a combination of a dependency parser and a graph parser. The dependency parser is a biaffine parser, that uses transformers for representing input sentences, with no other feature. The graph parser is a semantic parser that exploits a similar architecture except for using a sigmoid crossentropy loss function to return multiple values for the predicted arcs. The final output is obtained by merging the output of the two parsers. The dependency parser achieves top or close to top LAS performance with respect to other systems that report results on such metrics, except on low resource languages (Tamil, Estonian, Latvian).

1 System Overview

The shared task 2021 aims specifically at performing enhanced dependency parsing, starting from raw text, in a multi-language setting consisting of seventeen languages Bouma et al. (2021).

We concentrate on the syntactic parsing and enhancement stages, by exploiting existing tools for tokenization, sentence splitting.

2 Syntactic parsing

State of the art dependency parsers currently often adopt the graph-based model, based on neural networks for the choice of arcs and labels.

In particular the Bi-LSTM-based deep biaffine neural dependency parser by Dozat and Manning (2017) has been quite popular and used in three out of five of the top submissions to the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies (Zeman et al., 2018), in particular in the top non-ensemble submission (Kanerva et al., 2018).

We trained our own models for each language on the shared task treebanks using DiaParser, which uses the Stanza tokenizer and multi-word splitter.

2.1 DiaParser

DiaParser is a dependency parser derived from Supar[1], which exploits transformers to obtain contextualized word representations. Such representations are obtained by first applying the specific transformer tokenizer, splitting them into wordpieces, and then the embeddings for words is obtained as the average of the wordpiece embeddings.

The code for the parser is available on GitHub[2].

We exploit the idea to provide hints to the parser, obtained from structural syntax probes (Hewitt and Manning, 2019). We explored the idea to use a syntax probe to extract hints for the parser to estimate the most likely edges for the parse tree. Eventually a quite simple solution proved effective: to extract values from one of the attention layers of the transformer (typically layer 6) and add them to the score of the biaffine layer with a trainable weight $alpha$.

One may consider a transformer as computing three functions, the outputs $T_o : \mathbb{R}^{n\times d} \rightarrow \mathbb{R}^{n\times d}$, the hidden states $T_h : \mathbb{R}^{n\times d} \rightarrow \mathbb{R}^{L\times n\times d}$, and the attention weights $T_a : \mathbb{R}^{n\times d} \rightarrow \mathbb{R}^{H\times L\times n\times n}$ for H heads and for L layers.

Given a sentence with n words $\mathbf{w} = [w_1, w_2, ..., w_n]$, we feed the parser with $\mathbf{E} = [e_1, ..., e_n]$, where $e_i = mix_l(T_o(\mathbf{w}))_i$ is the scalar mix of the top l layers of the outputs of the transformer T applied to $\mathbf{w}$ (Liu et al., 2019a).

The attentive parser estimates the probability of

[1] https://github.com/yzhangcs/parser
[2] https://github.com/Unipisa/diaparser

Proceedings of the 17th International Conference on Parsing Technologies (IWPT 2021), pages 184–188
Bangkok, Thailand (online), August 6, 2021.

each possible arc for sentence $\mathbf{w}$ as follows:

$$\begin{aligned}\mathbf{v}_i^{(a\text{-}d)}, \mathbf{v}_i^{(a\text{-}h)} &= \mathrm{MLP}^{(a\text{-}d)}(\mathbf{e}_i), \mathrm{MLP}^{(a\text{-}h)}(\mathbf{e}_i), \\ A &= T_a(\mathbf{w})_l^{(h)} \\ S_{ij}^{(arc)} &= [\mathbf{v}_i^{(a\text{-}d)}, 1]^\top U^{arc} \mathbf{v}_i^{(a\text{-}h)} \\ &\quad + \alpha \cdot A_{ij} \\ P(y_{ij}^{(arc)}|\mathbf{w}) &= \mathrm{softmax}_j(S_i^{(arc)})\end{aligned} \tag{1}$$

where α is a learned weight and A are the attention weights of the transformer T for a given layer l and the given head h.

During prediction the syntactic parser applies the Chu-Liu-Edmonds algoritm (Chu, 1965; EDMONDS, 1967) to ensure the well-formedness of the parse tree, but only after a quick check that the arcs contain cycles.

The results we obtained with such an extension on the English development corpus where 92.21 UAS and 90.31 LAS, using Electra (Clark et al., 2020) as transformer as well as for attention, a small improvement with respect to 91.32 UAS and 89.33 LAS without using these features.

2.2 Semantic Graph Parser

The graph parser uses the approach of Dozat and Manning (2018).

The graph parser shares the same architecture as the biaffine dependency parser, except in for using a sigmoid cross entropy loss function instead of a softmax, to allow for multiple results. Those arcs with a logit value greater than zero are retained.

$$\begin{aligned}S_{ij}^{(arc)} &= \{s_{i,j} \geq 0\} \\ P(y_{ij}^{(arc)}|\mathbf{w}) &= \mathrm{argmax}_j(S_i^{(arc)})\end{aligned} \tag{2}$$

The scores of each pair of words in $\mathbf{w}$ can be decoded into a graph by keeping only edges that received a positive score. Labels are assigned to each such predicted edge, choosing the highest-scoring label for that edge.

The two losses of the edge and arc labels predictors are combined through an hyper-parameter $\lambda \in \{0, 1\}$:

$$\ell = \lambda \ell^{(label)} + (1 - \lambda)\ell^{(edge)} \tag{3}$$

The methods does not ensure a fully connected graph, hence we merge it with the tree produced by the syntactic parser.

The final enhanced dependency arcs are obtained as the union of the arcs predicted by the syntactic and semantic parsers, with a check that no extra arcs to the root are introduced.

3 System Description

3.1 Tokenization

DiaParser exploits the Stanza tokenizer and multi-word splitter to perform sentence splitting, tokenization and multi-word splitting. It automatically downloads tokenizer models for each language from the Stanza repository. We trained a specific MWT model for Italian, trained on the Italian UD treebank Italian_ISST, augmented with a special list of sentences, representative of 75 categories of verb conjugations and of articled prepositions, which we contributed back to the official Stanza distribution.

3.2 Experiments

The syntactic and semantic parsers were trained separately on each language corpus, using language specific transformer models, where available. For languages with more than one corpus, they were just concatenated together into a single corpus.

We used the following transformers for sentence representations and attention weights:

Lang.	Model
ar	asafaya/bert-large-arabic
bg	DeepPavlov/bert-base-bg-cs-pl-ru-cased
cs	DeepPavlov/bert-base-bg-cs-pl-ru-cased
en	google/electra-base-discriminator
fi	TurkuNLP/bert-base-finnish-cased-v1
fr	dbmdz/bert-base-french-europeana-cased
it	dbmdz/electra-base-italian-xxl-cased-discriminator
nl	wietsedv/bert-base-dutch-cased
ro	DeepPavlov/rubert-base-cased
sv	KB/bert-base-swedish-cased
uk	dbmdz/electra-base-ukrainian-cased-discriminator

Table 1: Transformer models used for each language.

For all other languages we used `bert-base-multilingual-cased`.

4 Settings and Results

4.1 Experimental Settings

In training, we used the official train and gold development sets. We used the development set to select the model hyper-parameters based on LAS for the dependency parser and labeled F1 on enhanced dependencies for the semantic graph parser.

We use a batch size of 2000 tokens with the AdamW (Loshchilov and Hutter, 2019) optimizer. The hyper-parameters of our system are shown in

Parameter	Value
Arc hidden size	500
Rel hidden size	100
MLP dropout	33%
Transformer layers	4
Optimizer	AdamW
Learning rate	5e-5
Warmup	0.1
Loss interpolation (λ)	0.1
batch size	2000

Table 2: Hyper parameters used in the experiments.

Table 2, which are mostly adopted from previous work on dependency parsing.

4.2 Results

The official results are those labeled unipi-smax in our submission, obtained through merging the outputs of the dependency and semantic graph parser.

Table 3 shows our team official results obtained in tokenization, tagging, parsing and enhancement on the test sets.

5 Pretrained Multilingual Model

After the submission deadline, we experimented building a single model on the concatenation of the training corpora of all languages. The corpora was preprocessed to eliminate empty nodes, which represent implicit nodes, denoted with IDs such as 2.1 in the CoNLLU file format. We used the official script `enhanced_collapse_empty_nodes.pl`, which collapses graphs reducing such empty nodes into non-empty nodes and introducing new dependency labels.

We used the official script to collapse graphs through reducing such empty nodes into non-empty nodes and introducing new dependency labels. In the post-process, we add empty nodes according to the dependency labels. As the official evaluation only score the collapsed graphs, such a process does not impact the system performance.

Then the enhanced dependency labels in the training corpus were de-lexicalized, stripping lexical information from labels, like in (Grünewald and Friedrich, 2020), replacing them with placeholders (e.g. obl:[case]) indicating where in the dependency graph the lexical information is expected to be found. This process allowed us to reduce the total number of enhanced dependency labels from 6125 to 1282.

This also made it possible to fit the model to be trained into the 32GB of memory of our V100 GPU. We run the training in parallel on 4 such GPUs: each epoch took about 45 minutes and run for 29 epochs.

The model was trained using contextualized word embeddings from RoBERTa (Liu et al., 2019b), more precisely `xlm-roberta-large` from HuggingFace[3] using a scalar mixture of the top 4 hidden layers (Liu et al., 2019a).

Then the model was fine tuned on each language with its specific language corpus. The enhanced dependency labels in the output of the parser are converted back to their lexical notation using a heuristic processing similar to the one outlined in (Grünewald and Friedrich, 2020):

Furthermore, for languages that have case morphology, like Czech, the case is added to the label.

The multilingual model does provide significant improvements for languages with smaller corpora, in particular Latvian, Lithuanian and Tamil, as shown in Table 4:

Notably Lithuanian improves on EULAS by 6.75 points. The ELAS scores do not improve as much, possibly due to the ri-lexicalization algorithms that may need tuning to each language.

6 Conclusions

We experimented using two parsers with the same architecture to perform syntactic and semantic parsing. We first trained parser models on the specific corpus for each language. The final output is obtained by merging the outputs of the two parsers. This simple approach works reasonably well for languages with large enough corpora.

To address the difficulty in handling low resource languages, we explored building a single model trained on all corpora and fine tuning it on each specific corpora. Since enhanced dependency labels contain lexical parts and the number of such labels is quite large, we adopted a preprocessing step to de-lexicalize the labels. The approach gave promising results on some languages, but the back-convertion algorithm that introduces the lexical parts in the labels after parsing still needs to be improved.

Given the similarity of the architectures of the syntactic and semantic parsers, the prospect of performing joint training is promising and has been

[3] https://huggingface.co/xlm-roberta-large

Language	Tok	Sent	UAS	LAS	EULAS	ELAS
Arabic	99.96	80.83	86.19	81.97	79.79	77.17
Bulgarian	99.93	97.49	95.29	92.71	91.89	90.84
Czech	99.91	95.05	94.13	92.36	90.14	88.73
Dutch	99.82	70.55	90.12	87.69	84.92	84.14
English	98.36	91.34	90.64	88.47	87.75	87.11
Estonian	99.62	87.44	87.11	84.14	82.66	81.27
Finnish	99.60	91.90	94.25	92.76	90.61	89.62
French	99.78	96.44	93.47	90.30	88.91	87.43
Italian	99.77	98.75	95.03	93.65	92.52	91.81
Latvian	99.82	99.07	89.90	86.63	83.92	83.01
Lithuanian	99.84	88.11	82.75	78.31	74.61	71.31
Polish	99.41	98.35	94.93	92.71	90.94	88.31
Russian	99.59	99.03	94.51	93.32	91.49	90.90
Slovak	99.96	86.00	93.32	91.75	88.77	86.05
Swedish	99.45	93.53	90.86	88.53	86.61	84.91
Tamil	99.01	88.35	63.27	56.04	54.16	51.73
Ukrainian	99.85	96.75	93.68	91.92	89.41	87.51
Average	**99.63**	**91.70**	**89.97**	**87.25**	**85.24**	**83.64**

Table 3: UNIPI Official results on the test set.

Language	Tok	Sent	UAS	LAS	EULAS	ELAS
Latvian	99.82	99.07	89.90	86.63	87.54	84.78
Lithuanian	99.84	88.11	82.75	78.31	81.36	76.62
Slovak	99.96	86.00	93.32	91.75	91.47	81.17
Tamil	99.01	88.35	63.27	56.04	55.90	53.71

Table 4: Preliminary results with multi-language model.

considered but left for further research.

Acknowledgments

The experiments were run on a server with four NVIDIA Tesla V100 GPUs generously offered by prof. Marco Aldinucci of the University of Turin.

References

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

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

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

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

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

J. EDMONDS. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards, B*, 71:233–240.

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.

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.

Jenna Kanerva, Filip Ginter, Niko Miekka, Akseli Leino, and Tapio Salakoski. 2018. Turku neural parser pipeline: An end-to-end system for the CoNLL 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 133–142, Brussels, Belgium. Association for Computational Linguistics.

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

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

Ilya Loshchilov and F. Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

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.