Implementing Bi-LSTM-based deep biaffine neural dependency parser for Vietnamese Universal Dependency Parsing

Nguyen Thi Thuy Lien
FPT University
Hanoi, Vietnam
liennttmse0096@fpt.edu.vn

Abstract

This paper presents our approach to resolve the Vietnamese Universal Dependency Parsing task in VLSP 2020 Evaluation Campaign. On the basis of Deep Biaffine Attention for Neural Dependency Parsing(Dozat and Manning, 2017), we adapted the dependency parser for Vietnamese. Our best model obtained a pretty good performance on the test datasets, achieving 84.08% UAS score and 75.64% LAS on average for the ConLL-U dataset. On the raw text data-set, the results we reached still quite limited, on average 74.47% of UAS and 65.3% of LAS.

1 Introduction

Dependency grammars is a family of grammar formalisms that are quite important in contemporary speech and language processing systems (Daniel Jurafsky, 2019). The dependency parsing task is to identify pairs of a dependent token and a head token that have dependency relation and their dependency relation labels in a given sentence. For decades, researchers have applied dependency parsing in many tasks of natural language processing such as information extraction, coreference resolution, question-answering, semantic parsing, etc.

Universal dependency parsing shared-task was proposed in VLSP 2020 evaluation campaign to promote the development of dependency parsers for Vietnamese(HA My Linh, 2020). The shared-task published a training corpus of approximately 10,000 dependency-annotated sentences. There are two parts of testing, the first one requires the participant to parse from the input as raw texts where no linguistic information is available. And the second, participant systems will have to parse dependencies information from linguistics annotated sentences. On the CoNLL-U formated test dataset, with the best model, we reached 84.08% UAS score and 75.64% LAS score (averaged on seven test sets). With the raw text dataset, we obtained 74.47% UAS score and 65.30% LAS score.

2 Related Works

Dependency parsing consists of transition-based, graph-based, and grammar-based parser (Nivre and Kübler, 2009). A graph-based algorithm finds the highest scoring parse tree from all possible outputs of an input sentence, scoring each complete tree, while a transition-based algorithm builds a parse by a sequence of actions and scoring each action individually (Zhang and Clark, 2008).

In 2016, Kiperwasser & Goldberg presented a scheme for dependency parsing which is based on bidirectional-LSTMs. The BiLSTM is trained jointly with the parser objective(Kiperwasser and Goldberg, 2016). The effectiveness was demonstrated in two ways by integrating it into a greedy transition-based parser and a globally optimized first-order graph-based parser. In both cases, this approach yields extremely competitive parsing accuracies.

In 2017, Dozat & Manning build off recent work from Kiperwasser & Goldberg, they use a larger but more thoroughly regularized parser than other recent BiLSTM-based approaches, with biaffine classifiers to predict arcs and labels (Dozat and Manning, 2017). Their parser gained state of the art or near state of the art performance on standard treebanks for six different languages.

3 Methodology

3.1 Data preprocessing

Training data includes 6 files, including 8150 sentences. In which, the number of different UPOS labels assigned is 30 and the number of XPOS labels is 56, these labels are unevenly distributed across the dataset.

Realizing that the appearance of some labels with a low sample count may negatively interfere with the results, we converted the group POS tag to accordingly non-group label, such as 'ADV:G' to 'ADV'. Simultaneously, we merge the labels with the same meanings but the different writing styles, such as Adv and ADV. The histogram of UPOS tag labels and XPOS tag labels after handling are shown in Figure 1 and Figure 2.

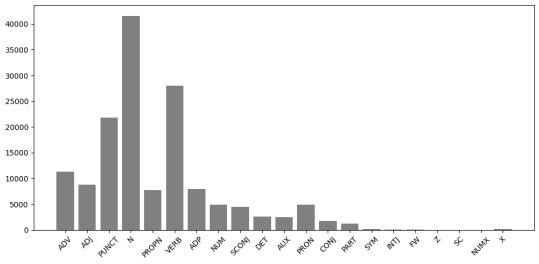

Figure 1: Histogram of processed Upos

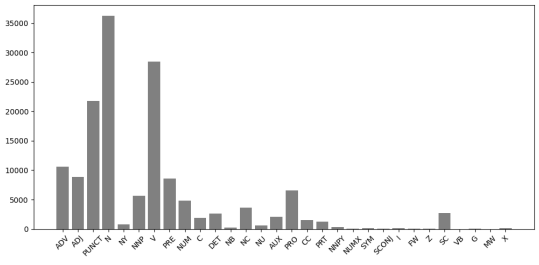

Figure 2: Histogram of processed Xpos

3.2 Proposed System

Tokenization and Sentence Splitting The first step of processing is tokenizing the raw text sentences. We used the VNCoreNLP toolkit to deal with this stage. In Vietnamese, lemmas are the same as the word forms.

POS Tagging To predict POS, we build a BERT-based (Devlin et al., 2019) classifier using bert-base-multilingual-cased pretrained-model available in HuggingFace (Wolf et al., 2020). The bert-base-multilingual-cased includes 12-layer, 768-hidden, 12-heads, 179M parameters, trained on cased text in the top 104 languages with the largest Wikipedia. This model was fine-tuned on the training data on total of 8 epochs using the hyper-parameters shown in Table 1.

Dependency Parsing We implemented a BiLSTM-based deep biaffine neural dependency parser (Dozat and Manning, 2017).

Hyper-parameters	Value
`lr`	2e-5
`eps`	1e-8
Optimizer	AdamW

Table 1: Hyper-parameters of the BERT classifier for pos.

Hyper-parameters	Value
Embedding size	100
Word embedding	fastText
`lr`	3e-3
Optimizer	Adam
LSTM size	400
Deep biaffine size	400
LSTM dropout	0.5
LSTM depth	3

Table 2: Hyper-parameters of the dependency parse.

We used Adam optimizer (Kingma and Ba, 2014) to optimize the network with the learning rate of 0.003 and fastText for word representations (Joulin et al., 2016). With fastText pre-trained word vectors, each word vector has 300 dimensions.

We set the max-steps to 50,000. However, after 3,000 steps without improvement in the validation accuracy, the training process is terminated instead of running through the whole 50,000 steps. After every 100 steps, a model checkpoint will be saved if there is an increase in validation accuracy. Table 2 summarises the hyper-parameters of the dependency parser we used in the parser.

In our experiments, we built two different models for dependency parsing, the first model uses both UPOS and XPOS information as training and predict data and the second model only uses UPOS information during the entire process.

4 Experiments & Results

The VLSP 2020 workshop provides two dependency parsing test datasets. The first one includes data files in raw text format and the other contains data files in which the sentences have been tokenized and stored in the CoNLL-U format.

Table 3 shows the evaluation results on the raw text dataset. Our system achieves 65.30% of LAS and 74.47% of UAS on average. The best result is obtained on the vn3 set which was crawled from VnExpress, with 69.29% of LAS and 77.09% of UAS. In contrast, the result recorded on the vn8 set is the lowest, just 59.35% of LAS and 69.61% of

Model		VTB	vn1	vn3	vn7	vn8	vn10	vn14	Avg. Score
First	UAS(%)	74.55	71.7	**77.09**	70.56	*69.61*	76.41	71.62	**74.47**
Model	LAS(%)	65.34	58.91	**69.29**	65.56	*59.35*	68.22	63.69	**65.30**

Table 3: Results on the raw text dataset.

Model		VTB	vn1	vn3	vn7	vn8	vn10	vn14	Avg. Score
First	UAS(%)	84.41	*74.34*	84.42	**85.56**	82.88	83.55	82.79	**84.08**
Model	LAS(%)	75.94	*63.37*	76.48	**78.89**	74.84	75.48	76.31	**75.64**
Second	UAS(%)	**83.20**	*68.32*	76.60	71.11	70.56	73.13	75.56	**81.58**
Model	LAS(%)	**75.14**	*55.95*	68.48	61.11	61.09	63.29	68.08	**73.32**

Table 4: Results on the CoNLL-U formated dataset.

UAS. One of the reasons that can be mentioned is that the subject of vn8 is somewhat different from the other data sets.

Table 4 presents the evaluation results on the CoNLL-U datasets. The model using both UPOS and XPOS information for training gives better results, 84.08% of UAS and 75.64% of LAS on average of seven datasets. This model works best on the vn7 dataset, reaching 85.56% of UAS and 78.89% of LAS. However, it performs worse on the vn1 set, obtains only 74.34% of UAS and 63.37% of LAS. The second model which uses only UPOS and tokens as input on the training process achieves a bit lower performance, with 81.58% averaged UAS score and 73.32% averaged LAS score. The result obtained when adding xpos feature are higher than using only upos feature. It proves that xpos feature has a relatively vital meaning in universal dependency parsing.

Experimental results indicate that the results obtained on raw text dataset is substantially worse than those obtained on data in CoNLL-U format. UAS decreased 9,61% and LAS reduced even more, up to 10.34% on average. A plausible explanation is that the raw data processing is not done effectively enough. On the other hand, the results that we achieved are relatively low compared to the evaluation on English data (Wilie et al., 2020). However, it implies that there will probably still be room for improvement.

5 Conclusion

In this paper, we present our experiments for the Vietnamese universal dependency parsing task at VLSP 2020 Evaluation Campaign. For raw text processing, we combine several toolkits and models. At the first step, we choose the VNCoreNLP toolkit as a tokenizer. Then a BERT classifier is used to detect the universal part-of-speech tags and Vietnamese part-of-speech tags. At the end, a Bi-LSTM-based deep biaffine neural dependency parser is implemented to produce dependency parsing results. We have obtained promising results on the test dataset, although the results are still lower than results on English datasets. It indicates that our approach probably still has space for growth. Our experiment includes separate modules, which are not inextricably linked. In the future works, we plan to continue doing experiments and improving the dependency parsing model. Next, we plan to build a comprehensive and unified pipeline system which processes raw text and generates dependencies information. In addition, we will also analyze more carefully the pre-processing and processing stages to give a convincing explanation for the difference between the results on the CoNLL-U formatted dataset and raw-text dataset, as well as the difference between files in these datasets.

References

James H. Martin Daniel Jurafsky. 2019. Chapter 15 Dependency Parsing. In *Speech and Language Processing*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing.

VU Xuan Luong NGUYEN Thi Luong PHAN Thi Hue LE Van Cuong HA My Linh, NGUYEN Thi Minh Huyen. 2020. Vlsp 2020 shared task: Universal dependency parsing for vietnamese. In *Proceedings of The seventh international workshop on*

Vietnamese Language and Speech Processing (VLSP 2020), Hanoi, Vietnam.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Joakim Nivre and Sandra Kübler. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2.

Bryan Wilie, Karissa Vincentio, Genta Indra Winata, Samuel Cahyawijaya, Xiaohong Li, Zhi Yuan Lim, Sidik Soleman, Rahmad Mahendra, Pascale Fung, Syafri Bahar, and Ayu Purwarianti. 2020. Indonlu: Benchmark and resources for evaluating indonesian natural language understanding.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii. Association for Computational Linguistics.