A Joint Deep Contextualized Word Representation for Deep Biaffine Dependency Parsing

Xuan-Dung Doan
Viettel Cyberspace Center, Viettel Group
Hanoi, Vietnam
dungdx4@viettel.com.vn

Abstract

We propose a joint deep contextualized word representation for dependency parsing. Our joint representation consists of five components: word representations from ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) language models for Vietnamese (Che et al., 2018; Nguyen and Nguyen, 2020), Word2Vec (Mikolov et al., 2013) embeddings trained on baomoi dataset (Xuan-Son Vu, 2019), character embeddings (Kim, 2014), and part-of-speech tag embeddings. When using the joint representation with a deep biaffine dependency parser (Dozat and Manning, 2016), our model ranks 2nd in Vietnamese Universal Dependency Parsing Shared-Task at VLSP 2020 (Linh et al., 2020).

1 Introduction

Dependency parsing is the task of automatically identifying binary grammatical relations between tokens in a sentence. There are two common approaches to dependency parsing: transition-based (Nivre, 2003; McDonald and Pereira, 2006), and graph-based (Eisner, 1996; McDonald et al., 2005a).

Recently, there has been a surge in the use of deep learning approaches to dependency parsing (Chen and Manning, 2014; Dyer et al., 2015; Kiperwasser and Goldberg, 2016; Dozat and Manning, 2016; Ma et al., 2018; Fernández-González and Gómez-Rodríguez, 2019; Zhang et al., 2020), which help alleviate the need for hand-crafted features, take advantage of the vast amount of raw data through word embeddings, and achieve state-of-the-art results.

Contextualized word representations, such as ELMo and BERT, have shown to be extremely helpful in a variety of NLP tasks. The contextualized model is used as a feature extractor, which is able to encode semantic and syntactic information of the input into a vector.

In this work, we further improve dependency parsing performance by making good use of external contextualized word representations.

2 Related works

Che et al. (2018) incorporated ELMo into both dependency parser and ensemble parser training with different initialization. Their system achieved the best result in CoNLL 2018 shared task.

Li et al. (2019) captured contextual information by combining the power of both BiLSTM and self-attention via model ensembles. The results led to a new state-of-the-art parsing performance.

Nguyen and Nguyen (2020) replaced the pre-trained word embedding of each word in an input sentence by corresponding contextualized embedding computed for the first subword token of the word. They achieve the state-of-the-art performance on VnDT dependency treebank v1.1 (Nguyen et al., 2014).

3 Methodology

In our model, an input sentence of n words $w = w_1, w_2, ..., w_n$ is fed to each of the component networks to learn separate token embeddings. We describe the learning process below.

3.1 Graph-based Dependency Parsing

Graph-based Dependency Parsing follows the common structured prediction paradigm (McDonald et al., 2005a; Taskar et al., 2005):

$$predict(w) = \underset{y \in \mathcal{Y}(w)}{\text{argmax}}\, score_{global}(w, y) \quad (1)$$

$$score_{global}(w, y) = \sum_{part \in y} score_{local}(w, part) \quad (2)$$

Given an input sentence w (and the corresponding sequence of the vectors $w_{1:n}$), we look the highest-score parse tree y in the space $\mathcal{Y}$(w) of valid dependency trees over w. In order to make the search tractable, the scoring function is decomposed to the sum of local scores for each part independently.

3.2 Word Embedding

The input layer maps each input word w_i into a dense vector representation x_i. We use word2vec (Mikolov et al., 2013) embeddings trained on baomoi dataset (Xuan-Son Vu, 2019) $emb^{word}_{w_i}$, a CNN-encoder character representation (Kim, 2014) $emb^{char}_{\hat{w}_i}$, and POS-tag embedding is created randomize to enrich each word's representation $emb^{tag}_{t_i}$ further.

$$x_i = emb^{word}_{w_i} \oplus emb^{char}_{\hat{w}_i} \oplus emb^{tag}_{t_i} \qquad (3)$$

3.3 Deep Contextualized Word Representations

3.3.1 ELMO

ELMO uses an LSTM (Hochreiter and Schmidhuber, 1997) network to encode words in a sentence and training the LSTM network with language modeling objective on large-scale raw text. $ELMo_i$ calculates the hidden representation $h_i^{(LM)}$ as

$$h_i^{(LM)} = BiLSTM^{(LM)}(h_0^{(LM)}, (\hat{w}_1, ..., \hat{w}_n))_i \qquad (4)$$

where $\hat{w}_i$ is the output of a CNN over characters. ELMo representational power is computed by a linear combination of BiLSTM layers:

$$ELMo_i = \gamma \sum_{j=0}^{L} s_j h_{i,j}^{(LM)} \qquad (5)$$

where s_j is a softmax-normalized task-specific parameter and γ is a task-specific scalar. We use the Vietnamese ELMo model released by Che et al. (2018).

3.3.2 BERT

BERT introduced an alternative language modeling objective to be used during training of the model. Instead of predicting the next token, the model is expected to guess a masked token. BERT is based on the Transformer architecture (Vaswani et al., 2017), which carries the benefit of learning potential dependencies between words directly. For use in downstream tasks, BERT extract the Transformer's encoding of each token at the last layer, which effectively produces $BERT_i$.

PhoBERT (Nguyen and Nguyen, 2020) was introduced for the Vietnamese NLP community as a Roberta-based model (Liu et al., 2019). PhoBERT achieves the state-of-the-art in Vietnamese POS-tag and Named Entity Recognition. Therefore, we use PhoBERT to produce $BERT_i$.

After getting $ELMo_i$ and $BERT_i$, we use them as an additional word embedding. The calculation of x_i becomes:

$$\begin{aligned} x_i = emb^{word}_{w_i} \oplus emb^{char}_{\hat{w}_i} \oplus emb^{tag}_{t_i} \\ \oplus ELMo_i \oplus BERT_i \end{aligned} \qquad (6)$$

The BiLSTM is used to capture the context information of each word. Finally, the encoder outputs a sequence of hidden states s_i.

3.4 Biaffine Attention Mechanism

We use the Biaffine attention mechanism described in (Dozat and Manning, 2016) for our dependency parser. The task is posed as a classification problem, where given a dependent word, the goal is to predict the head word (or the incoming arc). Formally, let s_i and h_t be the BiLSTM output states for the dependent word and a candidate head word respectively, the score for the arc between s_i and h_t is calculated as:

$$e_i^t = h_t^T W s_i + U^T h_t + V^T s_i + b \qquad (7)$$

Where W, U, V, b are parameters, denoting the weight matrix of the bi-linear term, the two weight vectors of the linear terms, and the bias vector.

Similarly, the dependency label classifier also uses a biaffine function to score each label, given the head word vector h_t and child vector s_i as inputs.

3.5 Training Loss

The parser defines a local cross-entropy loss for each position i. Assuming w_j is the gold-standard head of w_i, the corresponding loss is

$$loss(s, i) = -log \frac{e^{score(i \leftarrow j)}}{\sum_{0 \leq k \leq n, k \neq i} e^{score(i \leftarrow k)}} \qquad (8)$$

3.6 Dependency Parsing Decoding

The decoding problem of this parsing model is solved by using the Maximum Spanning Tree (MST) algorithm (McDonald et al., 2005b).

4 Experiments

4.1 Dataset

The VLSP organizers released the datasets in two phases. We split the first dataset into training, development, and test data, according to the 7:1:2 ratio. We then merge the second dataset into the first training data. The final statistics are summarized in Table 1.

Table 1: Statistics of the public dataset

	Number of sentences
Train set	6626
Develop set	507
Test set	1010

4.2 Setup

Table 2 summarizes the hyper-parameters that we use in our experiments. We implement an additional model that trains on lowercased input data, since the dataset also includes text from social media, which contains many word-form errors. We compare our results with the graph-based Deep Biaffine (BiAF) (Dozat and Manning, 2016) parser. Since the private test set of the VLSP Shared Task contains raw text only, we use VncoreNLP (Vu et al., 2018) to segment and POS-tag the raw data. Parsing performance is measured using UAS metric (Unlabeled Attachment Score) and LAS metric (Labeled Attachment Score) by comparing the gold relations of the test set and relations returned by the system. We use the evaluation script published at CoNLL 2018 [1].

Table 2: Hyper-parameters in our experiments

	Layer	Hyper-Parameter	Value
Input	Word	dimension	300
	POS	dimension	50
	Char	dimension	50
LSTM	Encoder	encoder layer	6
		encoder size	500
	MLP	arc MLP size	512
		label MLP size	128
	Training	Dropout	0.33
		optimizer	Adam
		learning rate	0.001
		batch size	80
ELMo		dimension	1024
BERT		dimension	768

4.3 Main Results

The results on the test set are shown in Table 3.

Table 3: The results (UAS%/LAS%) on the test set

	UAS/LAS
BiAF	80.83/69.40
Our model	82.86/**71.16**
Our lowercase model	**83.02**/71.05

The raw private test set after segmentation and POS tagging by VncoreNLP is the input to our model. The results on the raw private test set are shown in Table 4.

Table 4: The results (UAS%/LAS%) on each file of the raw private test set

	Our model	Our lowercase model
VTB	**76.33/67.46**	75.68/66.59
vn1	**74.79/65.38**	72.17/62.61
vn3	74.22/66.73	**74.95/67.28**
vn7	**68.33/61.67**	66.11/61.11
vn8	**74.81**/65.71	74.29/**65.97**
vn10	**80.64/72.46**	78.45/69.98
vn14	72.61/62.45	**73.36/63.69**
Total	**76.12/67.32**	75.48/66.53

Beside providing the private raw data set, VLSP organizers also provide the data in CoNLL-U (Ginter et al., 2017) format. The results on the private CoNLL-U format test set are shown in Table 5.

Table 5: The results (UAS%/LAS%) on each file of the private CoNLL-U format test set

	Our model	Our lowercase model
VTB	**84.81/76.44**	84.58/76.29
vn1	**78.98/70.94**	77.43/70.17
vn3	**85.89**/76.97	85.46/**77.58**
vn7	**82.22/75.56**	80.00/73.89
vn8	**82.49/73.93**	81.32/73.8
vn10	**85.46/77.53**	81.20/72.69
vn14	**84.04**/75.31	83.54/**76.81**
Total	**84.65/76.27**	84.23/76.05

The final result is calculated by averaging UAS and LAS scores on the raw private data and the private CoNLL-U format data. The official rank

[1] https://universaldependencies.org/conll18/conll18_ud_eval.py

is based on average the final UAS and LAS score. The final result of all teams is shown in Table 6.

Table 6: The final results (UAS%/LAS%/Average%) of all teams

	UAS	LAS	Aver.	Rank
Our model	80.39	71.80	76.09	2
DP2	80.89	71.36	76.12	1
DP3	78.58	70.04	74.31	4
DP4	79.28	70.47	74.87	3
DP5	77.28	68.77	73.03	5

Our model ranks 1st in LAS and 2nd in UAS. Finally, we rank 2nd on average UAS and LAS, officially.

5 Conclusion

We present joint ELMO and BERT as features for dependency parsing. In the future, we plan to analyze the effectiveness of our model when ELMO and/or BERT are excluded. We also plan to improve our model by using the self-attention mechanism as a replacement for the BiLSTM-based encoder in our current model.

References

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. *CoRR*, abs/1807.03121.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. 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. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2019. Left-to-right dependency parsing with pointer networks. *CoRR*, abs/1903.08445.

Filip Ginter, Jan Hajič, Juhani Luotolahti, Milan Straka, and Daniel Zeman. 2017. CoNLL 2017 shared task - automatically annotated raw texts and word embeddings. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

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.

Ying Li, Zhenghua Li, Min Zhang, Rui Wang, Sheng Li, and Luo Si. 2019. Self-attentive biaffine dependency parsing. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5067–5073. International Joint Conferences on Artificial Intelligence Organization.

HA My Linh, NGUYEN Thi Minh Huyen, VU Xuan Luong, NGUYEN Thi Luong, PHAN Thi Hue, and LE Van Cuong. 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.

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

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard H. Hovy. 2018. Stack-pointer networks for dependency parsing. *CoRR*, abs/1805.01087.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd*

Annual Meeting of the Association for Computational Linguistics (ACL'05), pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*.

Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. PhoBERT: Pre-trained language models for Vietnamese. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1037–1042.

Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, Phuong-Thai Nguyen, and Minh Le Nguyen. 2014. From treebank conversion to automatic dependency parsing for vietnamese. In *Natural Language Processing and Information Systems*, pages 196–207, Cham. Springer International Publishing.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 896–903, New York, NY, USA. Association for Computing Machinery.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Thanh Vu, Dat Quoc Nguyen, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. 2018. VnCoreNLP: A Vietnamese natural language processing toolkit. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 56–60, New Orleans, Louisiana. Association for Computational Linguistics.

Son N. Tran Lili Jiang Xuan-Son Vu, Thanh Vu. 2019. Etnlp: A visual-aided systematic approach to select pre-trained embeddings for a downstream task. *In: Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP)*.

Yu Zhang, Zhenghua Li, and Min Zhang. 2020. Efficient second-order TreeCRF for neural dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3295–3305, Online. Association for Computational Linguistics.