Integrating Label Attention into CRF-based Vietnamese Constituency Parser

Duy Vu-Tran, Phu-Thinh Pham, Duc Do, An-Vinh Luong and **Dien Dinh**
University of Science, Ho Chi Minh city, Vietnam
Vietnam National University, Ho Chi Minh city, Vietnam
{vtduy18, phpthinh18}@apcs.fitus.edu.vn
{dotrananhduc, anvinhluong}@gmail.com
ddien@fit.hcmus.edu.vn

Abstract

Attention mechanisms and linear-chain conditional random field (CRF) have been applied to constituency parsing, and the achieved results are phenomenal. While self-attention and label attention layers (LAL) have been proven to be state-of-the-arts in English constituency parsing for their improvement in the encoding phase, the CRF two-stage technique shows its effectiveness in lowering computational cost. Attention-based architectures allow a word (self-attention) or a label (label-attention) to include its own viewpoint into extracted information. Our system is an extension of the current CRF-based model with additional attention-based methods to improve the quality of the encoding phase. Another crucial factor in our encoder is BERT as the pre-trained model has gained recognition in various natural language processing (NLP) tasks. Taking the advantage of different methods, we implement a model that combines label attention, contextualized encoding, and conditional random field. Furthermore, we adopt the biaffine attention, which is mainly used in the dependency parsing task, in our scoring layer. The architecture performs greatly on the Vietnamese treebank as it gives an over-85 F1-score on the test set and an over-82 F1-score on the dev set. On a larger scale, our idea of integration could be utilized in other language models.

1 Introduction

In the modern era, syntactical parsing has gained remarkable results due to the rise of deep learning and neural networks (Mrini et al., 2020; Zhang et al., 2020; Zhou and Zhao, 2019; Stern et al., 2017; Wang and Tu, 2020; Yang and Deng, 2020). Especially, constant improvements on English and Chinese constituency parsing come from proposals of machine learning techniques (Mrini et al., 2020; Zhang et al., 2020; Zhou and Zhao, 2019; Stern et al., 2017), which encourages us to apply such models to the same task of the Vietnamese language. Since most of the existing constituency parsers are encoder-decoder architectures, the main approach to improving the performance is to upgrade either encoder or decoder or both.

Encoders handle inputs and extract their significance in vector forms so that the model can easily understand them. Particularly, the inputs of a constituency parser are sentences, and the encoders try to learn the information of each word or span of the sentences. Recurrent neural networks (RNN) and Long short-term memory networks (LSTM) are the main tools to extract such features from data for their ability to learn the contextual characteristics, and Zhang et al. (2020); Stern et al. (2017); Gaddy et al. (2018) have benefited from these mechanisms. PHAN et al. (2019), one of the pioneers in Vietnamese constituency parsing, used BiLSTM in their encoder. Despite RNNs and LSTMs' ability to capture contextual information, they are surpassed by the works of self-attention.

In 2017, Vaswani et al. (2017) presented the model of Transformer and the self-attention mechanism, which opened a new chapter for natural language processing (NLP). Self-attention layers are capable to understand the global context of a given input and additionally, an attention-weighted view of the input's words to itself. With self-attention, (Kitaev and Klein, 2018) improved the works of (Stern et al., 2017; Gaddy et al., 2018). Later on, Tran et al. (2020) adopted the model for Vietnamese constituency parsing successfully, which inspires us to combine the architecture with our encoder.

After the emergence of Transformer, BERT (Devlin et al., 2019), which is a Transformer-based model and pre-trained on a large corpus of a target language, was proposed. PHAN et al. (2019); Tran et al. (2020) included PhoBERT (Nguyen and Nguyen, 2020) in their models as PhoBERT is specifically trained on Vietnamese, and impressive results are achieved. Besides that, PhoBERT

also performed astonishingly in other Vietnamese NLP tasks (Nguyen and Nguyen, 2021) which we believe to enhance our encoder with the pre-trained knowledge of Vietnamese.

In 2020, Mrini et al. (2020) further developed the attention mechanism which resulted in Label Attention Layer (LAL). While self-attention (Vaswani et al., 2017) refers to input's views to itself, LAL offers the view of a label to the given sentence. Self-attention, BERT, and LAL are composed to better the encoder. The impact of LAL is further discussed in the Experiment section 3.

Besides dealing with the encoder, we adopt the conditional random field (CRF) concepts and two-stage decoding from Zhang et al. (2020) as they have proposed an efficient inside algorithm and proven the effectiveness of the methods. We also consider the biaffine attention (Dozat and Manning, 2017) for the scorer which is inspired by the task dependency parsing.

In this paper, we make a combination of current mechanisms into one single model: self-attention layers, label attention layers, PhoBERT for encoder; Biaffine Attention for scoring; two-stage CRF method for the decoder. We conduct examine our model on Vietnamese treebank (Nguyen et al., 2009) and achieve the results of over 85 on the test set and over 82 on the dev set (all results are given in F1-score). Section 2 re-describe our parser in the order of encoder, scorer, decoder, loss function. Section 3 presents the experiments' settings and a comparison between our model and other methods, and finally, we give our conclusion in section 4.

2 Model Architecture

2.1 Overview

Our parser includes 3 main components: encoder, scorer, decoder. Before being processed by the parser, the input sentence is transformed into the desired representations for the encoder. The encoder is a combination of the Attention mechanism and BERT, both of which extract essential information of the input sequence. On the Attention branch, the word-level weighted views for the sentence are extracted by k self-attention layers (Vaswani et al., 2017), and the following d_{lal}-head label attention(Mrini et al., 2020) is responsible for enhancing these outcomes with label-level weighted views. Additionally, the BERT(Devlin et al., 2019) branch handles the input independently and provides the features that are learned from the pre-training process. The results of two branches are aggregated into one sentence representation, which goes through the Biaffine scorer (Dozat and Manning, 2017) to compute the span score matrix, and the label score tensor. For the decoder, we apply the theory of Conditional Random Field (CRF) (Zhang et al., 2020) and lower the computational cost with the two-phase strategy. The model is referred to figure 1.

2.2 Encoder

Our encoder includes 2 parts: the Attention part and the BERT part. A visualization of the Attention part is given in figure 2. The idea of the encoder is based on the hypothesis that a constituency tree depends on the set of constituency labels and the context of the given text. To realize the assumption, we use:

- k self-attention layers to extract the information of how each word of a sequence 'see' the sequence's context itself.
- d_{lal}-head label-attention layer on the top of the self-attention layers to retrieve the viewpoint of the constituency labels to the input sentence.
- BERT on the other branch to improve the encoder with pretrained contextual features.

Token representations For the Attention part, we concatenate the content and the position embedding following (Vaswani et al., 2017), where the content is represented by the part-of-speech (POS) embedding [1]. We choose POS embedding instead of the word embedding because we observe that the POS one gives higher performance (which we will discuss in the Experiment section (3)) and because the information of words are extracted by the BERT layer instead.
For the BERT part, we tokenize the sequence into subword representations which are widely used for BERT encoders.

Self-attention Mechanism We follow the encoder implementation of (Mrini et al., 2020) in which the token representations are put through several self-attention layers (Vaswani et al., 2017) before being processed by the LAL.

Self-attention consists of a number of consecutive layers, each of which has identical operations.

[1]The POS tags are in XPOS forms.

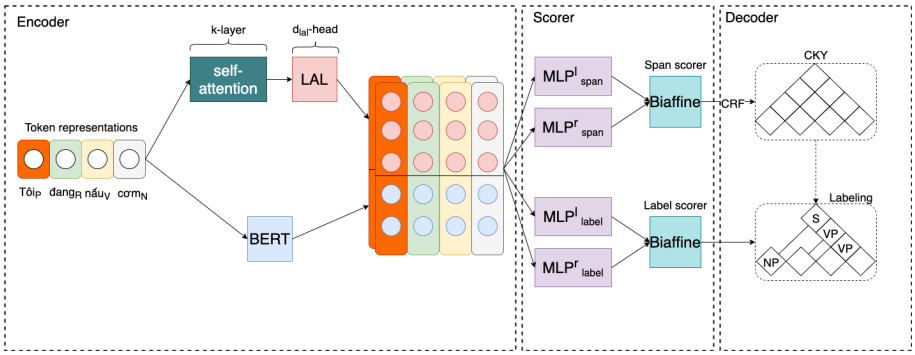

Figure 1: Model architecture. The main flow of the model for the sentence "Tôi đang nấu cơm". The sentence is encoded by Label Attention Layer (Mrini et al., 2020) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019). Afterwards, it is moved through MLP to extract features before being scored by the Biaffine (Dozat and Manning, 2017) mechanism and Conditional Random Field (CRF) (Zhang et al., 2020). The score is decoded using Cocke–Kasami–Younger (CKY) algorithm.

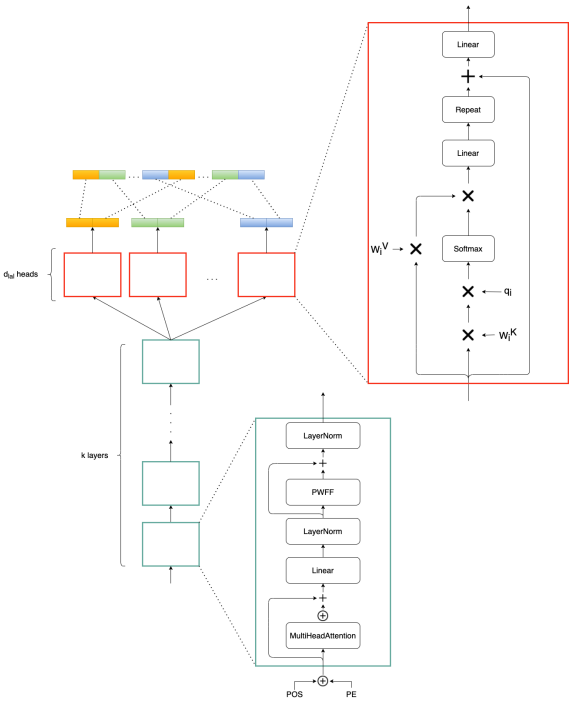

Figure 2: The encoder of attention mechanism includes 2 parts. Firstly, The token representations go through k self-attention layers to extract the attentive-weighted information. Subsequently, the data are fed into d_{lal}-head label attention. The output of each head is divided into 2 halves which act as forward and backward features of the word. (The $\times$ operator refers to the matrix multiplication operation, and the $\oplus$ implies the concatenation. POS and PE stand for part-of-speech embedding and positional embedding).

In a layer k of the self-attention, for a single head j in the self-attention layers, given the input matrix $X^{(k)}$:

$$c'_j = softmax(\frac{(W'^Q_j X^{(k)})(W'^K_j X^{(k)})}{\sqrt{d'}})W'^V_j X^{(k)} \quad (1)$$

where W'^Q_j, W'^K_j, W'^V_j are the learnt matrices of the head j. We used $'$ to distinguish notations between the label attention and the original self-attention. Each attention head j learns from the input individually as they have their own trainable parameters. The chosen number of heads is 8 following (Vaswani et al., 2017).

To aggregate multiple heads of attention, we concatenate the outputs together:

$$output' = Linear(c'_1 \oplus c'_2 \oplus ... \oplus c'_8) \quad (2)$$

where the $Linear$ operation is used to transform the output back to the input X 's dimension. $Multihead$ is followed by a residual connection and a Layer Normalization (Ba et al., 2016) which results in the layer k's output.

$$\begin{aligned} L^{(k)} &\equiv Layer^{(k)}(X^{(k)}) \\ &= LayerNorm(X^{(k)} + output') \end{aligned} \quad (3)$$

Before being fed to the next layer $k+1$, the output goes through a position-wise feed-forward which has the same form as in (Vaswani et al., 2017) and a residual-LayerNorm again:

$$X^{(k+1)} = LayerNorm(L^{(k)} + PWFF(L^{(k)})) \quad (4)$$

where PWFF is the position-wise feed-forward function.

$$PWFF(X) = Linear(ReLU(Linear(X))) \quad (5)$$

where ReLU refers to the Rectified Linear Unit function. The input and output of position-wise feed-forward have the same dimensions (Kitaev and Klein, 2018).

Label Attention Layer For English constituency parsing, the authors of (Mrini et al., 2020) have proposes the Label Attention Layer(LAL), and it has produced a state-of-the-art result. As Label Attention allows the labels to give their attention views on a given sentence, we apply the mechanism to our model as a part of encoder. A comparison between models with and without LAL is made and reported in the Experiment section (3).
The LAL takes a matrix X as input which consists of embedded vectors of words of a given sentence. The attention weight of each head j is represented by an attention-weight vector a_j which is calculated from the query vector q_j.

$$a_j = softmax(\frac{q_j(W_j^K X)}{\sqrt{d}}) \tag{6}$$

where W_j^k is a learnt matrix of head j, and it is used to learn the key factors of X. d is the length of the query vector. Afterwards, the context vectors c_j are computed (Label Attention Layer uses vectors instead of matrices).

$$c_j = a_j(W_j^V X) \tag{7}$$

where W_j^V is the learnt value matrix of head j.

The context vectors are projected to the dimension X's vectors before being repeatedly added to X.

$$c_j = Linear(c_j) \tag{8}$$

$$output_j = \begin{bmatrix} c_j \\ c_j \\ \dots \\ c_j \end{bmatrix} + X \tag{9}$$

The $output_j$ is a matrix representing the label j view of the sequence X, and it is projected to a smaller dimension d_{lal} to prevent overfitting as well as to optimize computational effectiveness. The outcomes of all attention heads j (for $j \in [1; d_{label}]$), given that d_label is the number of heads), are concatenated into the final output of LAL.

$$output_j = Linear(output_j) \tag{10}$$

$$\mathbf{L} = output_1 \oplus output_2 \oplus ... \oplus output_{d_{label}} \tag{11}$$

$\mathbf{L} \in \mathbb{R}^{n \times (d_{label} * d_{lal})}$ where n is the length of the given sequence, and $\mathbf{L}_i$ represents the vector of the given sequence's i^{th} word.

Although the paper (Mrini et al., 2020) states that we can choose as many attention heads as the labels, they also show that the relation between the number of attention heads and one of the labels is not specifically one-to-one. This number is a hyper-parameter that should be chosen via experiments which we present in the Experiment section(3). The LAL is put on top of the self-attention module to provide the labels' weighted views for the words' attentions. In other words, the self-attention layers are not replaced by the LAL but instead, are enhanced by it.

Inside the self-attention and label attention layers, we apply the partition for content embedding and positional embedding as it is shown in (Kitaev and Klein, 2018) to gain better performance.

BERT Fine-tuning BERT (Devlin et al., 2019) is a context-aware embedding encoder, and it is pretrained on a large corpus of the target language. We decide to fine-tune the encoder further which leads the pretrained model to fit the treebank. The authors of (Nguyen and Nguyen, 2020) have introduced a RoBERTA-based (Liu et al., 2019) model: Phobert, and the model is pretrained on a large Vietnamese corpus, which makes it more suitable for the task. PhoBERT (Nguyen and Nguyen, 2020) has shown impressive improvement in many Vietnamese tasks such as dependency parsing, POS tagging, named entity recognition (Nguyen and Nguyen, 2021). Given a sequence of subwords $\{sw_1, sw_2, ..., sw_n\}$, the output of BERT is denoted as:

$$\mathbf{B}_i = BERT(sw_i) \tag{12}$$

While the LAL provides the viewpoint of the labels to a given delexical sequence, BERT supports the contextual attention views of each word to the input sentence.

Input Representation Following (Stern et al., 2017), we combine the forward and backward representations of the output. We split the composition of BERT and LAL in two halves and treat them as the forward and backward representations. For a n-long sequence s, the process is shown in equations

13, 14, 15.

$$\mathbf{for}_i = \mathbf{B}_i[0:n/2] \oplus \mathbf{L}_i[0:n/2] \quad (13)$$

$$\mathbf{back}_i = \mathbf{B}_i[n/2+1:n] \oplus \mathbf{L}_i[n/2+1:n] \quad (14)$$

$$\mathbf{E}_i = \mathbf{for}_i \oplus \mathbf{back}_{i+1} \quad (15)$$

2.3 Scorer

Inspired by the Biaffine Attention (Dozat and Manning, 2017) used for the task of dependency parsing, we apply the mechanism to our scoring architecture. The authors of (Zhang et al., 2020) made a comparison between the Biaffine scoring method with the previous one (Stern et al., 2017), and the Biaffine one gave a consistently higher performance.

Feature Extraction We extract the information of the left and right boundaries using two separate MLP layers as each word w_i acts as either the left boundary (the span endpoint is to the left of w_i) or the right boundary (the span endpoint is to the right of w_i) of a span. Another two MLP layers are used for the labelling task.

$$\mathbf{span}_i^l, \mathbf{span}_i^r = MLP_{span}^l(\mathbf{E}_i), MLP_{span}^r(\mathbf{E}_i) \quad (16)$$

$$\mathbf{label}_i^l, \mathbf{label}_i^r = MLP_{label}^l(\mathbf{E}_i), MLP_{label}^r(\mathbf{E}_i) \quad (17)$$

Biaffine Scorer The Biaffine takes the outputs from the feature extraction process as it inputs. Particularly for a span (i, j), the Biaffine uses the left boundary features of the i^{th} word and the right one of the j^{th} to score the span. Similarly, we gain the label scores of any span (i, j) and label $l \in \ell$.

$$s(i,j) = \begin{bmatrix} \mathbf{span}_i^l \\ 1 \end{bmatrix}^T \mathbf{D} \left[\mathbf{span}_j^r\right] \quad (18)$$

where $\mathbf{D}$ is a trainable parameter, and $\mathbf{D} \in \mathbb{R}^{d \times d}$. d is the output dimension of $MLP_{span}^{l/r}$.

$$s(i,j,l) = \begin{bmatrix} \mathbf{label}_i^l \\ 1 \end{bmatrix}^T \mathbf{D}_l \begin{bmatrix} \mathbf{label}_j^r \\ 1 \end{bmatrix} \quad (19)$$

where $\mathbf{D}_l$ is a trainable parameter, and $\mathbf{D} \in \mathbb{R}^{\hat{d} \times |\ell| \times \hat{d}}$. $\hat{d}$ is the output dimension of $MLP_{label}^{l/r}$.

2.4 CRF Decoder

In (Zhang et al., 2020), the authors proposed a two-stage framework for constituency parsing, which they proved to have a lower computational cost. Given a sentence x, our goal is to find an optimal tree $\hat{\mathbf{Y}}$. While the previous one-stage method (Stern et al., 2017; Gaddy et al., 2018) tries to parse the optimal tree directly, the two-stage method firstly finds the optimal unlabelled tree.

$$S(x,y) = \sum_{(i,j) \in y} s(i,j) \quad (20)$$

where $S(x, y)$ is the total score of the parsed tree (given x) which is calculated by summing all edge's scores of a legal tree y. y is one of the candidate constituency tree. We put the score under CRF to calculate the conditional probability.

$$p(y|x) = \frac{e^{S(x,y)}}{\sum_{y' \in Tr(x)} e^{S(x,y')}} \quad (21)$$

where $Tr(x)$ is the set of legal trees. Under CRF, the constituency tree is optimized by the CKY algorithm.

$$\hat{\mathbf{Y}} = \arg\max_y p(y|x) \quad (22)$$

The next stage is to identify the label of the optimal unlabelled tree. For each constituent (i, j) of a given tree y and n-length sentence x, the label $\hat{\mathbf{L}}$ of (i, j) is:

$$\hat{\mathbf{L}} = \arg\max_{l \in \ell} s(i,j,l) \quad (23)$$

where ℓ is the set of all possible constituency labels.

According to (Zhang et al., 2020), the complexity of the decoding phase is $\mathcal{O}(n^3 + n|\ell|)$ as the CKY algorithm takes $\mathcal{O}(n^3)$ time complexity and the second stage takes $\mathcal{O}(|\ell|)$ for each edge in y (a constituency tree has $2n - 1$ constituents).

2.5 Loss Function

The loss function of our model consists of 2 parts: the span loss and the label loss. The span loss is calculated by the CRF loss as we try to maximize the conditional probability in equation 21. In other words, given a sentence x and its target tree y, we minimize the $-\log p(y|x)$:

$$L_{span}(x,y) = -S(x,y) + \log \sum_{y' \in Tr(x)} e^{S(x,y')} \quad (24)$$

The second term of equation 24 has an efficiency batchified calculation which is detailed in (Zhang et al., 2020). The label loss L_{label} is computed using the cross entropy function, and the total loss of the model is the sum of two parts:

$$L_{total}(x, y, l) = L_{span}(x, y) + L_{label}(x, y, l) \quad (25)$$

where l is the set of possible labels.

3 Experiment

3.1 Experiment Setup

Data We use the Vietnamese treebank (Nguyen et al., 2009) to train and test our models. The dataset is divided into 3 sets: train, dev, test with 8321, 692, 1388 sentences respectively. The data are pre-processed using the code[4] provided by (Kitaev and Klein, 2018). The root constituents are included in the data, and the function tags are removed due to the purpose of the task. The data are biased as the numbers of 'NP' and 'VP' labels dominate others, which results from the high amount of Noun and Verb words in the treebank. This is heavily affected by Vietnamese grammar, where a simple clause is usually formed from a noun phrase and a verb phrase. Figure 3 and 4 visualize the statistic of constituency labels and POS tags in the dataset.

Parameter choice For the self-attention and label attention layers, we adopt directly the setting of (Mrini et al., 2020) without any tuning except for the number of label attention heads. Following (Mrini et al., 2020), we choose the output dimensions for the MLPs in 2.3 to be 1024 for MLP_{span}'s and 250 MLP_{label}'s. For other parameter settings, we directly follow the choice of (Dozat and Manning, 2017). The token-batch size is 1000, and the training process runs for 100 epochs. The model is evaluated on the performance of the dev set.

Measurement We follow the standard measurement precision (P), recall (R), F-score (F) for evaluation with the help of the `EVALB` tool[2].

Models define Our baseline model is the original CRF model using CharLSTM and BiLSTM-based encoder of (Zhang et al., 2020)[3] with the same settings of the authors. Other models to be compared are listed below:

- *CRF model using pre-trained PhoBERT and BiLSTM encoder*[3]: we examine the impact of our encoder with the LSTM-based one.
- *Berkeley Neural Parser (Benepar) using pre-trained PhoBERT*[4]: we compare our model with the previous method used in (Tran et al., 2020).
- *Our model without using Label Attention Layers*[5]: we consider the contribution of label attention to the parser.

For models using PhoBERT, we re-train them on both base and large versions of PhoBERT to evaluate their influences. Furthermore, we try different setups and use two types of token representations for our LAL to find the optimal choice.

3.2 Results

Table 1 compares our model's scores with other models' on the Vietnam treebank's dev and test set. Overall, our parser (using the large version of PhoBERT) obtains the highest score in all scores (precision, recall, F-score) on both dev and test set. With PhoBERT$_{base}$, the model slightly drops by 0.45 in precision, 0.21 in recall, 0.33 in F-score on the dev set while the numbers are 0.78, 0.48, 0.64 respectively on the test set.

3.3 Evaluation

Evaluation on dev set We conduct the experiments on both versions of PhoBERT (Nguyen and Nguyen, 2020), and we observe that the contribution of Phobert$_{large}$ is greater than PhoBERT$_{base}$. The difference in F-score between the 2 versions ranges from 0.3 to 0.48. When using Character-level LSTM instead of PhoBERT, the scores fall sharply, which proves the essence of the Vietnamese pre-trained model.

With the rich information provided by the encoder and the effectiveness of the Biaffine scorer, our model (w PhoBERT$_{large}$) gives better results in all precision, recall, and F-score on the Vietnamese treebank (Nguyen et al., 2009) as it increases by 1.70% in F-score compared to the Benepar model (w PhoBERT$_{large}$) and 1.78% higher compared to the CRF model (w PhoBERT$_{large}$).

Evaluation on test set On the test set, PhoBERT$_{large}$ continues to outperform

[2]https://nlp.cs.nyu.edu/evalb/

[3]https://github.com/yzhangcs/parser

[4]https://github.com/nikitakit/self-attentive-parser

[5]Our model without LAL refers to removing the *self-attention → LAL* block in our encoder.

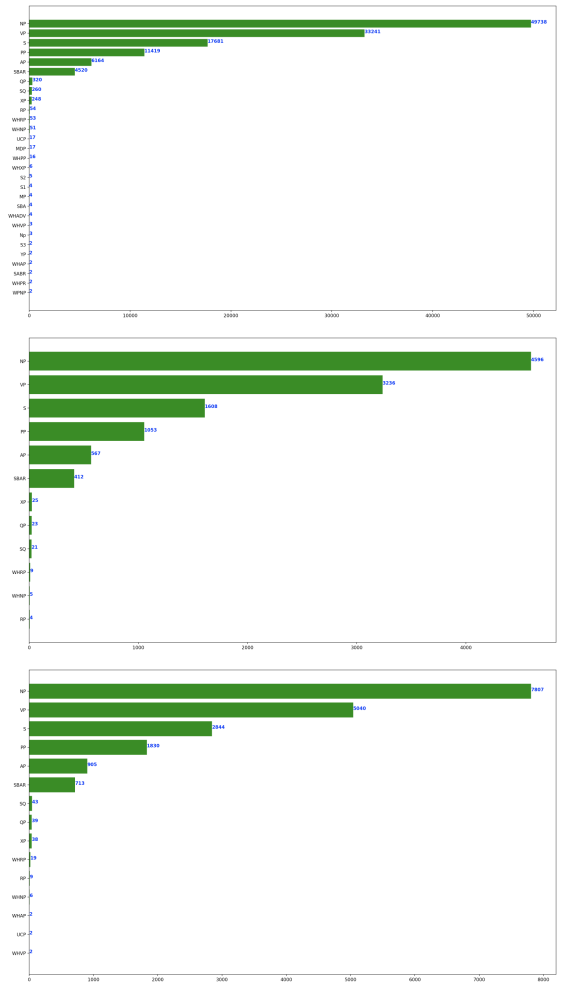

Figure 3: Constituency labels statistic (removed labels whose frequency is 1). Up down: train set, dev set, test set

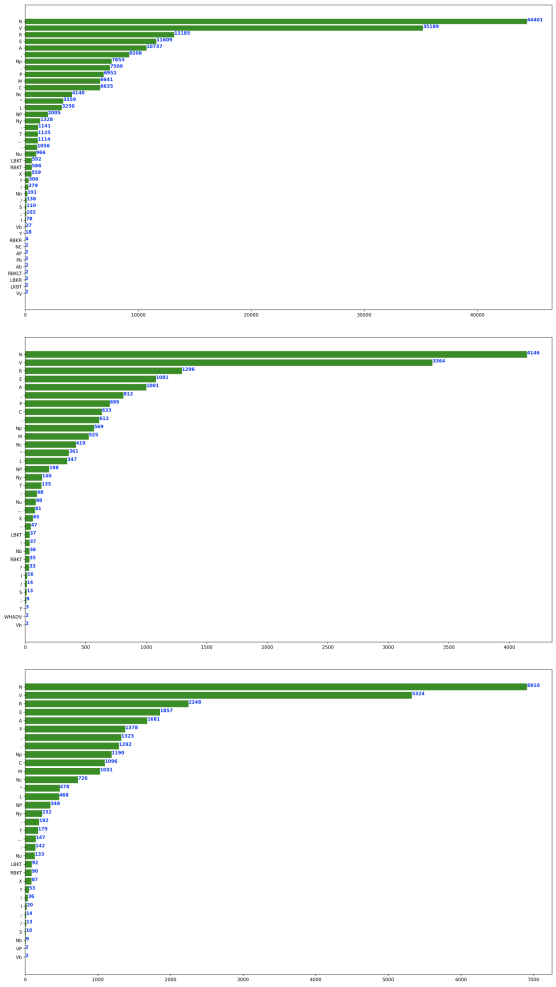

Figure 4: POS tags statistic(removed tags whose frequency is 1). Up down: train set, dev set, test set

PhoBERT$_{base}$ in the Benepar and our models, with 0.51 and 0.64 higher in F-score respectively. However, CRF models give a contrasting result, one of whose explanations can be that the LSTM encoder of the CRF models cannot handle thoroughly massive features gained by PhoBERT$_{large}$.

Our model receives absolute higher results as it achieves a consistent improvement of more than 1 F-score compared to other models. It is clear that the components of our combination benefit from each other and they reconcile to surpass the current architectures.

Impact of Label Attention Layers We examine the importance of the Label Attention mechanism by removing it from the model, and the result drops by 1.42 (the model w PhoBERT$_{large}$) and 0.78 (the model w PhoBERT$_{base}$). Without the LAL, the result is still higher than other tested models.

We make a comparison between different numbers of attention heads in LAL, and table 2 gives the details. In our model defined in 3.1, we use 64 heads for it gives the best F-score on the test set. We test a lower number of heads (32) and surprisingly, the obtained dev set's F-score is better (0.1 higher compared to the 64-head) while the test set's F-score is not significantly lower (0.04 lower compared to the 64-head). Following (Mrini et al., 2020), we choose the last model with 89 heads as there are 89 label heads in Vietnamese treebank and obtain the peak performance on the dev set. Although the 89-head model gives peak performance on the dev set, it requires much more time to train. On the other hand, with the lowest computational cost, the 32-head model still achieves a relatively high F-score on both the dev and test set.

As mentioned in token representations in section 2.2, we use POS embedding instead of word

embedding as the content for the *self-attention* → *LAL* block. To make the decision, we trained our model with either the POS or the word embedding whose performances are shown in table 3. Using the word embedding leads to a steep decrease, with 1.65 and 1.68 F-score drops on the dev and test set respectively, which might result from a large amount of the words in the dictionary (the size of the POS's dictionary is only 63).

4 Conclusion

While English and Chinese constituency parsing has achieved significant improvement with different advanced techniques, the task of the Vietnamese language still lacks approaches. We propose an extension of CRF-based model (Zhang et al., 2020) with label attention mechanism (Mrini et al., 2020) to enhance the performance of constituency parsing. This paper takes the advantages of each mechanism to gain a greater impact: pretrained PhoBERT provides knowledge of Vietnamese; self-attention and label attention retrieve the view of words and labels respectively; biaffine attention enhances the scoring framework, and two-stage decoding lowers computational cost. The outcomes of the model are promising with a high F-score, and the idea of combination can be generalized for other languages.

Acknowledgments

This research is supported by research funding from Advanced Program in Computer Science, University of Science, Vietnam National University - Ho Chi Minh City.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

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.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. 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 999–1010, New Orleans, Louisiana. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. 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. 2019. Roberta: A robustly optimized bert pretraining approach.

Khalil Mrini, Franck Dernoncourt, Quan Hung Tran, Trung Bui, Walter Chang, and Ndapa Nakashole. 2020. Rethinking self-attention: Towards interpretability in neural parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 731–742, Online. Association for Computational Linguistics.

Dat Quoc Nguyen and Anh Tuan Nguyen. 2020. Phobert: Pre-trained language models for vietnamese.

Linh The Nguyen and Dat Quoc Nguyen. 2021. Phonlp: A joint multi-task learning model for vietnamese part-of-speech tagging, named entity recognition and dependency parsing.

Phuong-Thai Nguyen, Xuan-Luong Vu, Thi-Minh-Huyen Nguyen, Van-Hiep Nguyen, and Hong-Phuong Le. 2009. Building a large syntactically-annotated corpus of Vietnamese. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 182–185, Suntec, Singapore. Association for Computational Linguistics.

Thi-Phuong-Uyen PHAN, Ngoc-Thanh-Tung HUYNH, Hung-Thinh TRUONG, Tuan-An DAO, and Dien DINH. 2019. Vietnamese span-based constituency parsing with bert embedding. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–7.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 818–827, Vancouver, Canada. Association for Computational Linguistics.

Tuan-Vi Tran, Xuan-Thien Pham, Duc-Vu Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. 2020. An empirical study for vietnamese constituency parsing with pre-training.

		Dev set			Test set		
		P	R	F	P	R	F
CRF	w CharLSTM	72.41	73.70	73.05	75.78	78.01	76.88
	w pre-trained PhoBERT$_{large}$	80.50	82.10	81.29	82.90	85.19	84.03
	w pre-trained PhoBERT$_{base}$	80.14	81.50	80.81	83.38	85.47	84.41
Benepar	w pre-trained PhoBERT$_{large}$	81.68	81.07	81.37	84.03	85.02	84.52
	w pre-trained PhoBERT$_{base}$	81.04	81.10	81.07	83.65	84.36	84.01
Ours	w PhoBERT$_{large}$	**82.32**	**83.84**	**83.07**	**84.80**	**87.16**	**85.97**
	w PhoBERT$_{base}$	81.87	83.63	82.74	84.02	86.68	85.33
	w PhoBERT$_{large}$ w/o LAL	80.90	82.49	81.69	83.26	85.88	84.55

Table 1: Results table

Number of attention heads	**Dev set**			**Test set**		
	P	R	F	P	R	F
32	82.67	83.70	83.18	84.79	87.09	85.93
64	82.32	**83.84**	83.08	84.80	**87.16**	**85.97**
89	**82.85**	83.65	**83.25**	**85.09**	86.60	85.84

Table 2: Comparison of different number of attention heads of Label Attention

Token representations	**Dev set**			**Test set**		
	P	R	F	P	R	F
word embedding	80.46	82.44	81.43	83.07	85.55	84.29
POS embedding	**82.32**	**83.84**	**83.08**	**84.80**	**87.16**	**85.97**

Table 3: Comparison of different used content embedding

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xinyu Wang and Kewei Tu. 2020. Second-order neural dependency parsing with message passing and end-to-end training. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 93–99, Suzhou, China. Association for Computational Linguistics.

Kaiyu Yang and Jia Deng. 2020. Strongly incremental constituency parsing with graph neural networks.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural crf constituency parsing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4046–4053. International Joint Conferences on Artificial Intelligence Organization. Main track.

Junru Zhou and Hai Zhao. 2019. Head-Driven Phrase Structure Grammar parsing on Penn Treebank. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2396–2408, Florence, Italy. Association for Computational Linguistics.