PhoLSTM: A Joint Multi-task Model in Vietnamese Natural Language Processing

Simon Park and **Jason Tan** and **Kawin Tiyawattanaroj**
Princeton University
`{juhyunp, jztan, kawint}@princeton.edu`

Abstract

We present PhoLSTM, a joint multi-task language model that can simultaneously learn and perform part-of-speech tagging, named-entity recognition, and dependency parsing in Vietnamese. By replacing a feed-forward neural network in the dependency parsing layer of PhoNLP (Nguyen and Nguyen, 2021) with a LSTM, PhoLSTM outperforms PhoNLP on benchmark datasets. Although PhoLSTM is currently only trained with a small dataset in Vietnamese, we hope that the model can be applied in larger datasets or in other languages.

1 Introduction

With the rise of Vietnamese NLP research in recent years, part-of-speech (POS) tagging, named-entity recognition (NER), and dependency parsing are among the tasks that have been studied extensively. Since they are all fundamental tasks, they can be embedded in other NLP tasks, such as machine translation (Tran et al., 2016), semantic parsing, open information extraction, and question answering. Therefore, developing accurate, efficient models for POS tagging, NER, and dependency parsing is essential in the advancement of the field of Vietnamese NLP as a whole.

PhoBERT (Nguyen and Tuan Nguyen, 2020) is the current state-of-the-art monolingual pre-trained language model for Vietnamese, and since then, many models have attempted to fine-tune PhoBERT to perform POS tagging, NER, and dependency parsing *independently*. However, having three fine-tuned models, one for each task, not only requires large storage space (three times the size of a single model), but it also erases the interdependence of these three tasks. In reality, the three tasks are closely related: POS tags are essential for dependency parsing, and they can also enhance the NER task.

PhoNLP (Nguyen and Nguyen, 2021) introduces a joint multi-task model which trains on and performs the three tasks *simultaneously*. It reports that the model outperforms previous state-of-the-art models for each of the single tasks. In this project, we attempt to improve the performance of the PhoNLP model by modifying the architecture. Specifically, we introduce PhoLSTM, where the feed-forward neural network (FFNN) in the dependency parsing layer is replaced with a LSTM, which is generally proved to be superior to a FFNN. We then analyze the performance and the robustness against random seeds of our model.

2 Background

2.1 PhoNLP

Proposed by Nguyen and Nguyen (2021), PhoNLP is the first model to jointly learn and perform POS tagging, NER, and dependency parsing for Vietnamese.

In particular, given an input sentence of words, the encoding layer generates contextualized word embeddings that represent the input words. These embedding vectors are fed into a FFNN to predict POS tags for corresponding input words. Each predicted POS tag is then represented by two "soft" embeddings that are separately fed into NER and dependency parsing layers. The NER and the dependency parsing layers each concatenate the original word embedding vectors and their corresponding soft embedding vectors. Then the resulting vectors are fed into a FFNN. Figure 1 from the paper illustrates the structure of the model.

The authors report that PhoNLP produces state-of-the-art results (POS tagging accuracy 93.88%, NER F_1-score 94.51%, dependency parsing LAS 78.17%, dependency parsing UAS 84.95%), outperforming the single task models that perform each task independently (93.68%, 93.69%, 77.89%, 84.78% on POS, NER, LAS, and UAS respectively).

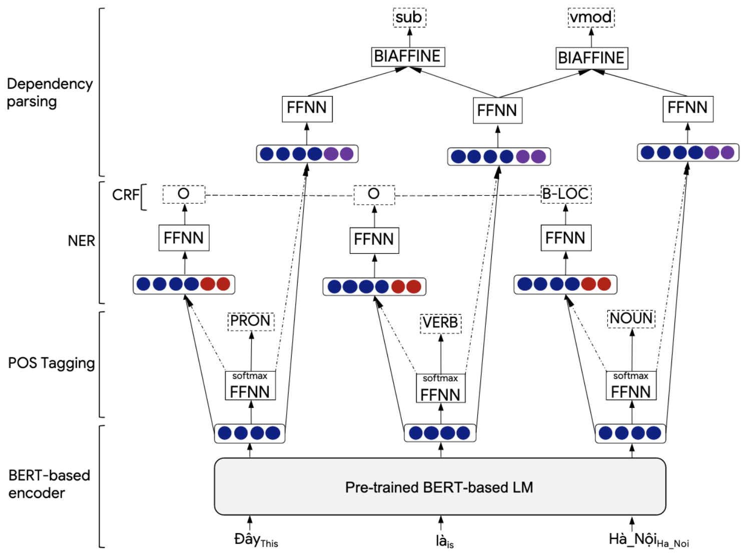

Figure 1: PhoNLP Model

2.2 Motivation

While PhoNLP is able to achieve high accuracy on POS tagging and NER at more than 90% accuracy on both tasks, it does not perform as well on dependency parsing in which it achieves relatively lower accuracy scores at around 80%. Our primary motivation is to modify the PhoNLP model to improve the performance on dependency parsing.

The original model uses a FFNN in the dependency parsing layer. We propose a new model, called PhoLSTM, where the FFNN in the dependency parsing layer is replaced with a bidirectional LSTM as shown in Figure 2.

2.3 LSTMs

Popularized by (Gers et al., 2000), Long Short-Term Memory RNN (LSTM) is a type of RNN that is superior to pre-existing neural networks, especially non-recurrent neural networks like FFNN. One major advantage of LSTM is that it mitigates one of FFNN's major drawbacks: "vanishing gradients," which occur when neural networks are unable to propagate useful gradient information as we process longer sequences. LSTMs partially solve such problem by introducing features such as gates and cells to maintain information from the past (Narasimhan, 2022). In fact, LSTM is proven to be effective in many Vietnamese NLP tasks, including sentiment analysis (Vo et al. (2017), Nguyen et al. (2018b)), and speech detection (Van Huynh et al., 2019). Based on these results, we hypothesize that by replacing a FFNN layer with a LSTM, our modified model would perform better than pre-existing results in Nguyen and Nguyen (2021).

3 Model Description

The parts of this section that overlap with PhoNLP are adapted from Nguyen and Nguyen (2021) with slight modifications for clarity. The part on the dependency parsing layer illustrates the major change we made to the model. We additionally provide a detailed description of the FFNN layer in the POS tagging and NER layers, which was omitted in Nguyen and Nguyen (2021).

3.1 Encoder & Contextualized Embeddings

Given an input sentence consisting of n words $w_1, w_2, \cdots, w_n$, the encoding layer PhoBERT uses BPE (Sennrich et al., 2016) to segment it into subword units and to generate contextualized latent feature embeddings $\mathbf{e}_i$ each representing the first subword of the i^{th} word w_i:

$$\mathbf{e}_i = \text{PhoBERT}_{\text{base}}(w_{1:n}, i) \in \mathbb{R}^d \qquad (1)$$

where d is the size of the embedding layer.

3.2 POS tagging

Following Devlin et al. (2019), the POS tagging layer feeds the contextualized word embeddings $\mathbf{e}_i$ into a feed-forward network (FFNN$_{\text{POS}}$) followed by a $\texttt{softmax}$ predictor for POS tag prediction:

$$\mathbf{p}_i = \texttt{softmax}(\text{FFNN}_{\text{POS}}(\mathbf{e}_i)) \in \mathbb{R}^{k_{pos}} \qquad (2)$$

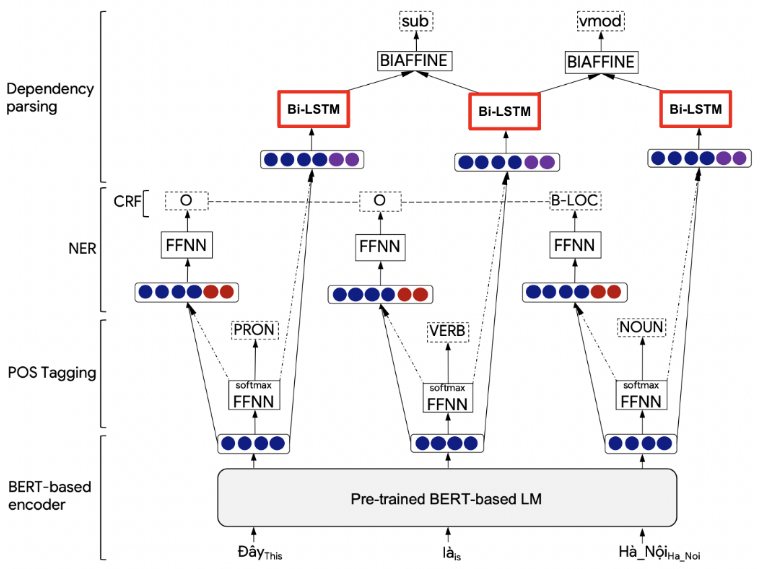

Figure 2: Revised Model with Bi-LSTMs

where k_{pos} is the number of POS tags. Based on probability vectors $\mathbf{p}_i$, a cross-entropy objective loss $\mathcal{L}_{\mathbf{POS}}$ is calculated for POS tagging during training.

3.3 NER

Following Hashimoto et al. (2017), the soft POS tag embedding $\mathbf{t}_i^{(1)}$ is computed by multiplying a learnable weight matrix $\mathbf{W}^{(1)} \in \mathbb{R}^{d_{soft} \times k_{pos}}$ with the corresponding probability vector $\mathbf{p}_i$, where d_{soft} is a hyperparameter that denotes the size of the soft embedding layer:

$$\mathbf{t}_i^{(1)} = \mathbf{W}^{(1)} \mathbf{p}_i \in \mathbb{R}^{d_{soft}} \tag{3}$$

Then, the NER layer creates a sequence of vectors $\mathbf{v}_{1:n}^{(1)}$ where each $\mathbf{v}_i^{(1)}$ is resulted in by concatenating the contextualized word embedding $\mathbf{e}_i$ and the soft POS tag embedding $\mathbf{t}_i^{(1)}$:

$$\mathbf{v}_i^{(1)} = \mathbf{e}_i \circ \mathbf{t}_i^{(1)} \in \mathbb{R}^{d+d_{soft}} \tag{4}$$

The NER layer then passes each vector $\mathbf{v}_i$ into a feed-forward neural network (FFNN_{NER}):

$$\mathbf{h}_i^{(1)} = \text{FFNN}_{\text{NER}}(\mathbf{v}_i^{(1)}) \in \mathbb{R}^{k_{ner}} \tag{5}$$

where k_{ner} is the number of NER labels.

The NER layer feeds the output vectors $\mathbf{h}_i$ into a linear-chain CRF predictor for NER label prediction (Lafferty et al., 2001). A cross-entropy loss $\mathcal{L}_{\mathbf{NER}}$ is calculated for NER during training. For inference, the Viterbi algorithm is used.

3.4 FFNN Layer

The POS tagging layer and the NER layer both use a feed-forward neural network. Here we describe its structure. Given an input vector $\mathbf{u} = (u_1, \cdots, u_n) \in \mathbb{R}^n$, it is first fed through a replacement layer, where each coordinate of the vector is independently and with a fixed probability, replaced with a random value drawn from a normal distribution:

$$u_i \leftarrow \begin{cases} u_i' \sim \mathcal{N}\left(0, \frac{1}{n}\right) & \text{w.p. } p_{rep} \\ u_i & \text{otherwise} \end{cases} \tag{6}$$

where p_{rep} is the probability that the values are replaced. Then the resulting vector $\mathbf{u}$ is multiplied by a learnable weight matrix $\mathbf{W} \in \mathbb{R}^{n \times h}$ and is fed through a ReLU gate:

$$\mathbf{v} = \text{ReLU}(\mathbf{W}\mathbf{u}) \in \mathbb{R}^h \tag{7}$$

where h is the hidden size of this FFNN. The resulting vector $\mathbf{v}$ is then fed through a standard dropout layer, where each coordinate of the vector is independently either set to zero or scaled up:

$$v_i \leftarrow \begin{cases} 0 & \text{w.p. } p_{drop} \\ \frac{1}{1-p_{drop}} v_i & \text{otherwise} \end{cases} \tag{8}$$

where p_{drop} is the probability that the values are dropped.

3.5 Dependency parsing

Similarly to the NER layer, the soft POS tag embedding $\mathbf{t}_i^{(2)}$ is computed by multiplying a learnable weight matrix $\mathbf{W}^{(2)} \in \mathbb{R}^{d_{soft} \times k_{pos}}$ with the corresponding probability vector $\mathbf{p}_i$:

$$\mathbf{t}_i^{(2)} = \mathbf{W}^{(2)}\mathbf{p}_i \in \mathbb{R}^{d_{soft}} \tag{9}$$

Then the dependency parsing layer creates a sequence of vectors $\mathbf{v}_{1:n}^{(2)}$ where each $\mathbf{v}_i^{(2)}$ is resulted in by concatenating the contextualized word embedding $\mathbf{e}_i$ and the soft POS tag embedding $\mathbf{t}_i^{(2)}$

$$\mathbf{v}_i^{(2)} = \mathbf{e}_i \circ \mathbf{t}_i^{(2)} \in \mathbb{R}^{d+d_{soft}} \tag{10}$$

The dependency parsing layer then passes each vector $\mathbf{v}_i^{(2)}$ into a bi-directional LSTM$_{\text{DEP}}$:

$$\mathbf{h}_i^{(2)} = \text{LSTM}_{\text{DEP}}(\mathbf{v}_i^{(2)}) \in \mathbb{R}^{2h_{dep}} \tag{11}$$

where h_{dep} is the size of the hidden layer of the LSTM. Following Dozat and Manning (2017), each $\mathbf{h}_i^{(2)}$ is split into four vectors $\mathbf{h}_i^{(AH)}$, $\mathbf{h}_i^{(AD)}$, $\mathbf{h}_i^{(LH)}$, $\mathbf{h}_i^{(LD)} \in \mathbb{R}^{h_{dep}/2}$ of equal length such that

$$\mathbf{h}_i^{(2)} = \mathbf{h}_i^{(AH)} \circ \mathbf{h}_i^{(AD)} \circ \mathbf{h}_i^{(LH)} \circ \mathbf{h}_i^{(LD)} \tag{12}$$

To predict potential dependency arcs, the parsing layer feeds $\mathbf{h}_i^{(AH)}$, $\mathbf{h}_i^{(AD)}$ into a variant of a Biaffine classifier (Qi et al., 2018) that additionally takes into account the distance and relative ordering between two words to produce a probability distribution of arc heads for each word. For inference, the Chu-Liu/Edmonds' algorithm is used to find a maximum spanning tree (Chu and Liu, 1965; Edmonds, 1967).

The parsing layer also applies the Biaffine classifier to $\mathbf{h}_i^{(LH)}$, $\mathbf{h}_i^{(LD)}$ to label the predicted arcs. An objective loss $\mathcal{L}_{\mathbf{DEP}}$ is computed by summing a cross entropy loss for unlabeled dependency parsing and another cross entropy loss for dependency label prediction during training based on gold arcs and arc labels.

3.6 Joint multi-task learning

The final training objective loss $\mathcal{L}$ of our model is the weighted sum of the POS tagging loss $\mathcal{L}_{\mathbf{POS}}$, the NER loss $\mathcal{L}_{\mathbf{NER}}$, and the dependency parsing loss $\mathcal{L}_{\mathbf{DEP}}$:

$$\mathcal{L} = \lambda_{pos}\mathcal{L}_{\mathbf{POS}} + \lambda_{ner}\mathcal{L}_{\mathbf{NER}} + \lambda_{dep}\mathcal{L}_{\mathbf{DEP}} \tag{13}$$

where $\lambda_{pos}, \lambda_{ner}, \lambda_{dep} \in [0, 1]$ are hyperparameters that satisfy the following relationship:

$$\lambda_{pos} + \lambda_{ner} + \lambda_{dep} = 1$$

Model	**Task**	**#train**	**#valid**	**#test**
PhoLSTM	POS	2400	300	300
	NER	2400	300	300
	DEP	2400	300	300
PhoNLP	POS	23906	2009	3481
	NER	14861	2000	2831
	DEP	8977	200	1020

Table 1: Dataset statistics. #train, #valid, and #test each refer to the number of training, validation, and test sentences

4 Experiments

4.1 Setup

4.1.1 Datasets

We use a portion of the VLSP 2016 NER dataset (Nguyen et al., 2018a) for the NER task and the Vietnamese UD treebank [1] converted from the Vietnamese constituent treebank (Nguyen et al., 2009) for POS tagging and dependency parsing. Due to limitations in computing resources, we extract 3000 sentences from each dataset and split them 8:1:1 into train, validation, and test data.

This differs from the training setting of the PhoNLP model, which used the VLSP 2013 POS dataset [2] for POS tagging and the VnDT dependency treebank v1.1 (Nguyen et al., 2014) converted from the same Vietnamese constituent treebank for the dependency parsing. Table 1 summarizes the statistics of the datasets used for training our model, compared to that of PhoNLP.

4.1.2 Implementation

Our model is implemented based on PyTorch (Paszke et al., 2019), employing the PhoBERT encoder implementation available from the transformers library (Wolf et al., 2020), the Biaffine classifier implementation from Qi et al. (2020), and the AdamW optimizer (Loshchilov and Hutter, 2019). Table 2 summarizes the list of hyperparameters and the values we used in our implementation.

We perform a grid search to select the best values of $\lambda_{pos}, \lambda_{ner}, \lambda_{dep}$. Each experiment was run with a different random seed. Throughout the experiments, we compare models based on the average of the POS tagging accuracy, the NER F$_1$-score and the dependency parsing LAS. We select the model

[1] https://github.com/UniversalDependencies/UD_Vietnamese-VTB

[2] https://vlsp.org.vn/vlsp2013/eval

Layer	Parameter	Value
Embedding	d	768
FFNN_{POS}	p_{rep}	0.33
	h	400
	p_{drop}	0.5
NER	d_{soft}	100
FFNN_{NER}	p_{rep}	0.33
	h	400
	p_{drop}	0.5
Dependency	d_{soft}	100
	h_{dep}	256
Training	Batch Size	8
	Epoch	100
	Learning Rate	1e-5

Table 2: Full list of hyperparameters

		λ_{pos}		
		0.2	0.4	0.6
λ_{ner}	0.2	84.53	84.33	79.62
	0.4	**84.74**	83.87	X
	0.6	84.46	X	X

Table 3: Results of the grid search for optimal values of $\lambda_{pos}, \lambda_{ner}, \lambda_{dep}$. The reported score is the average of the POS tagging accuracy, the NER F_1-score and the dependency parsing LAS

checkpoint that generates the highest average score over the validation set.

4.1.3 Training Environment and Runtime

Training was performed on Google Colab, with 25GB of RAM and Tesla T4 GPU. Training a single model took approximately 6 hours (5 min / epoch) and 500MB of memory.

4.2 Results

Table 3 presents the results from the grid search to select the values of $\lambda_{pos}, \lambda_{ner}, \lambda_{dep}$. The value of λ_{dep} can be inferred from each cell by calculating

$$\lambda_{dep} = 1 - \lambda_{pos} - \lambda_{ner}$$

We report that the choice of $\lambda_{pos} = 0.2$, $\lambda_{ner} = 0.4$, $\lambda_{dep} = 0.4$ produced the best results. Note that the performance of the model is almost the same across the different experiments, each run with a different random seed. We infer from this that the PhoLSTM model is resilient to the choice of random seeds.

We then report a more detailed performance results of our model (with the best model checkpoint from Table 3), compared to the baseline PhoNLP model in Table 4. For the PhoNLP model, the POS tagging accuracy, the NER F_1-score and the dependency parsing LAS were largely dependent on the choice of random seeds. The reported scores are the result of being averaged over three random seeds.

Model	POS	NER	LAS	Score
PhoLSTM	90.86	**93.47**	69.88	**84.74**
PhoNLP	**91.44**	83.43	**71.70**	82.19

Table 4: Performance of PhoLSTM compared to PhoNLP

4.3 Discussion

When the PhoNLP model is trained on our sampled dataset, the performance of each task declines from what was originally reported in Nguyen and Nguyen (2021). Out of the three, NER is the task that is impacted most heavily, with the F_1 score dropping from 94.51% from to 83.43%. To identify the cause for this drop, we analyzed the confusion matrix from the worst-performing seed. Out of 138 words with the 'B-PER' tag, the model is only able to correctly identify 31 of them, and it mislabels 101 of them as 'O', the null tag. This suggests that the PhoNLP model requires more data to correctly identify the 'B-PER' tag. In comparison, our PhoLSTM model is able to correctly identify 121 of them, and the overall NER F_1 score is almost on the same level as that of PhoNLP on the full dataset. This shows that PhoLSTM requires less data to learn to distinguish the 'B-PER' tag from the null tag. This will be most useful if our model is trained on other languages with less training data.

5 Conclusion and Future Work

We have presented PhoLSTM, a joint multi-task model for POS tagging, NER, and dependency parsing in Vietnamese. Experiments on a subset of benchmark Vietnamese datasets show that PhoLSTM outperforms the PhoNLP baseline. More specifically, our model is shown to learn better on a smaller dataset and is shown to be more robust to the choice of random seeds. In future works, we hope to train our model on larger datasets or in other languages and verify if PhoLSTM generalizes better in other training settings.

References

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference of Learning Representations*.

Jack Edmonds. 1967. Optimal branching. *Journal of Research of the National Bureau of Standards*, pages 233–240.

Felix Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–2471.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of EMNLP*, pages 1923–1933.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Karthik Narasimhan. 2022. Lecture notes in COS484: Natural language processing.

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.

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.

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. In *Proceedings of NAACL-HLT 2021: Demonstrations*, pages 1–7.

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.

Thi Minh Huyen Nguyen, The Quyen Ngo, Xuan Luong Vu, Mai Vu Tran, and Thi Thu Hien Nguyen. 2018a. Vlsp shared task: Named entity recognition. *Journal of Computer Science and Cybernetics*, 34(4):283–294.

Vu Duc Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. 2018b. Variants of long short-term memory for sentiment analysis on vietnamese students' feedback corpus. In *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*, pages 306–311.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal Dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task*, pages 160–170.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Viet Hong Tran, Huyen Thuong Vu, Thu Hoai Pham, Vinh Van Nguyen, and Minh Le Nguyen. 2016. A reordering model for vietnamese-english statistical machine translation using dependency information. In *2016 IEEE International Conference on Computing Communication Technologies, Research, Innovation, and Vision for the Future*, pages 125–130.

Tin Van Huynh, Vu Duc Nguyen, Kiet Van Nguyen, Ngan Luu-Thuy Nguyen, and Anh Gia-Tuan Nguyen. 2019. Hate speech detection on vietnamese social media text using the bi-gru-lstm-cnn model.

Quan-Hoang Vo, Huy-Tien Nguyen, Bac Le, and Minh-Le Nguyen. 2017. Multi-channel lstm-cnn model for vietnamese sentiment analysis. In *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, pages 24–29.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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.