ICIC Express Letters
Part B: Applications
Volume **12**, Number **3**, March **2021**

ICIC International ©2021 ISSN 2185-2766
pp. **291–299**

IMPROVING MULTI-GRAINED NAMED ENTITY RECOGNITION WITH BERT AND FOCAL LOSS

Tu Dinh Tran[1,2], Minh Nhat Ha[1,2], Long Hong Buu Nguyen[1,2,*] and Dien Dinh[1,2]

[1]Faculty of Information Technology
University of Science, Ho Chi Minh City
227 Nguyen Van Cu Street, Ward 4, District 5, Ho Chi Minh City, Vietnam
{ tdtu; hnminh }@apcs.vn; ddien@fit.hcmus.edu.vn
*Corresponding author: long.hb.nguyen@gmail.com

[2]Vietnam National University, Ho Chi Minh City
Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam

Received July 2020; accepted October 2020

Abstract. *This paper presents an improved approach to the novel multi-grained named entity recognition model. Unlike conventional approaches which consider named entity recognition as sequential labeling, our model examines all possible named entity candidates with different length in Detector component and classifies them into predefined categories in Classifier component. To boost our performance, we improve the model by applying pretrained BERT embeddings, which is the state-of-the-art context representation method. Realizing the imbalance of positive and negative samples of named entity in the English and Vietnamese dataset, we replace the normal Cross Entropy loss with Focal Loss to solve this issue. Our experiments show that our proposed method is effective in multi-grained as well as single-grained entities.*
Keywords: Nested named entity, BERT, Focal Loss, Multi-grained named entity

1. **Introduction.** Named Entity Recognition (NER) is the task of identifying and labeling chunks of text into categories such as organization, person, and time. This task can be divided into 2 types: non-overlapped (single-grained) NER and nested (multi-grained) NER. While flat NER gets attention from the beginning, nested NER task (illustrated in Figure 1), whose entities are encapsulated over each other, has been focused on recently.

The conventional approach to flat NER is sequential tagging, which includes various approaches like Conditional Random Field (CRF) by Ratinov and Roth [1] or LSTM-based approach by Lample et al. [2]. These methods are not suitable with nested named entity due to the assumption that entity mentions do not overlap each other.

There are various approaches in attempt to solve the nested NER task. One of the initial approaches was proposed by Zhou et al. [3] that relied on rule-based method to detect overlapping mentions. Alex et al. [4] introduced several methods that incorporate Conditional Random Fields (CRFs) into NER task. However, these methods cannot handle overlapping entities of the same type. Discriminative constituency parser method proposed by Finkel and Manning [5] transforms each sentence into a tree, with constituents for each named entity. Wang and Lu [6] proposed a hypergraph-based approach to discover all possible entities. Wang et al. [7] introduced a transition-based approach, which parses entities into trees. Those methods which only aim to handle nested named entity perform inefficiently in flat NER task.

For Vietnamese, most of the previous works modify sequential tagging to be suitable with NER task. In [8], the authors proposed a deep learning model with a stack of multiple

DOI: 10.24507/icicelb.12.03.291

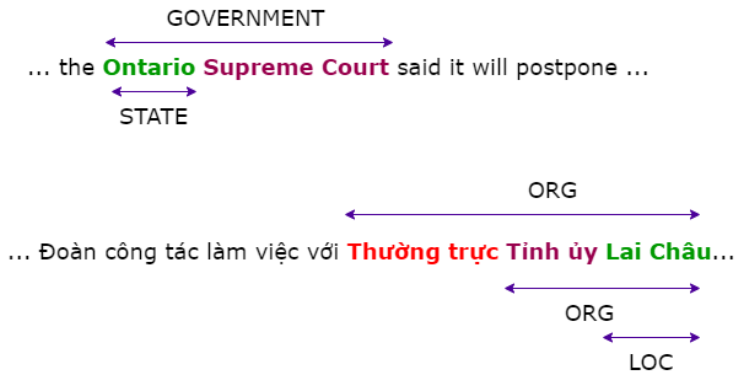

FIGURE 1. Examples of nested named entity in English and Vietnamese

BI-LSTM-CRF components. Additionally, they add linguistic information such as POS tag, chunk tag to enhance the performance. In [9], Pham compared among CRF-based NER models which are trained on each nested level and all nested levels.

Xia et al. [10] proposed an integrated framework named Multi-Grain Named Entity Recognition (MGNER), which detects all possible entities in various granularities and attaches attention mechanism for classification. This approach performs well not only on nested named entity but also on non-overlapping named entities. However, the dominance of negative samples over positive samples in dataset results in the bias of model over this class and the difficulty of classifying hard samples. To solve this drawback, we propose to use Focal Loss by Lin et al. [11] instead of Cross Entropy in MGNER. This loss function is initially utilized to solve the foreground-background class imbalance encountered in object detection, which is similar to our issue. In addition, we apply pretrained BERT embeddings in Devlin et al. [12] instead of pretrained ELMo embeddings in Peters et al. [13] as a result of truly bidirectional contextual information. Adapting the advantages of MGNER model, we also improve it and achieve better results in both English and Vietnamese datasets. Besides, our approach outperforms latest approaches for nested named entity recognition on Vietnamese dataset. Our contribution can be summarized as follows.

- We replace ELMo in the MGNER model by Xia et al. [10] with BERT to take advantages of the better contextual representations.
- We enhance MGNER model by Xia et al. [10] by applying the Focal Loss instead of Cross Entropy to solve the imbalance between positive and negative samples.
- We prove the effectiveness of our proposed method on English and Vietnamese data.

The rest of this paper is organized as follows. We present our model (Section 2), give information about our datasets, implementation and report the results (Section 3) and finally conclude and propose future works (Section 4).

2. **Proposed Model.**

Our model consists of 2 main components: the Detector and the Classifier. The Detector detects all possible named entities and passes these entities as input of the Classifier. Then, they are classified into predefined categories in the Classifier. The Detector is composed of 3 sub-components: Word Processor attaining word level information, Sentence Processor getting contextual information of sentence, Detection Network considering the entity proposals to be positive or negative. The Classifier also contains 3 sub-components: Word Processor being the same as one in the Detector, Entity Processor learning entity features, Classification Network labeling proposals into predefined categories. In addition, we apply pretrained BERT embeddings by Devlin et al. [12] in both the Detector and the Classifier to obtaining contextual representations. Besides, Focal Loss by Lin et al. [11] is utilized in Detection Network to solve the imbalance of positive and negative samples and to enable model to concentrate on hard-classified

entities. Self attention method is adopted in Entity Processor to let the entities learn the related context. To enhance the learning speed, some features are shared between the Detector and the Classifier: the pretrained word embeddings, the language model and the Sentence Processor's output in the Entity Processor.

2.1. **The Detector.** The Detector outputs a set of potential entities for every input sentence. As seen in Figure 2, the Detector has three modules: the Word Processor, the Sentence Processor and the Detection Network. In general, the Word Processor generates a word representation for each token in an input. Word representations are fed into the Sentence Processor to produce contextual sentence presentation. Then, for every input sentence, the Detection Network examines word segments surrounding every token for the possibility thereof to be an entity, and outputs those that qualify.

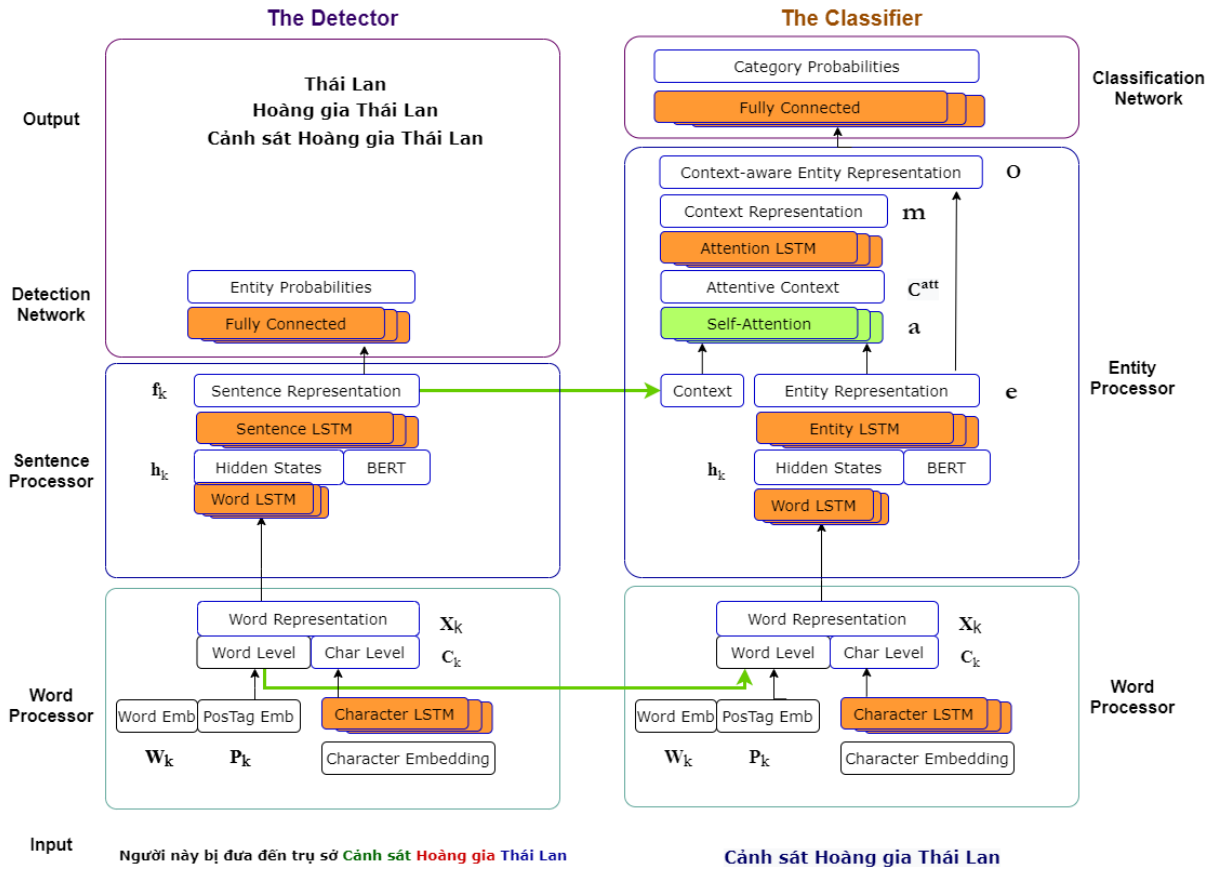

FIGURE 2. Proposed model

2.1.1. *Word Processor.* The Word Processor constructs the word representation for every token in an input sentence. Each token k yields a representation as:

$$x_k = [w_k; p_k; c_k] \tag{1}$$

w_k is the pretrained word embeddings obtained from language model with a dimension D_w. p_k is the POS tag if it exists and c_k is the character information. Character embeddings are input of a bidirectional LSTM with a hidden size D_{cl}, whose final hidden states backward and forward are concatenated as the character information c_k.

2.1.2. *Sentence Processor.* The word representation is fetched through a bi-LSTM layer to get the contextual information of sentence. For each token, the hidden states $\overrightarrow{h}$ of forward LSTM layer are concatenated with the hidden states $\overleftarrow{h}$ of backward LSTM to build the hidden states h_k. The dimension of word LSTM hidden states is D_{wl}.

$$\begin{aligned}\overrightarrow{h_k} &= LSTM_{fw}\left(x_k, \overrightarrow{h_{k-1}}\right)\\ \overleftarrow{h_k} &= LSTM_{fw}\left(x_k, \overleftarrow{h_{k-1}}\right)\\ h_k &= \left[\overrightarrow{h_k}; \overleftarrow{h_k}\right]\end{aligned} \tag{2}$$

As the emergence of pretrained BERT embeddings by Devlin et al. [12] and their effectiveness, we concatenate pretrained BERT embedding of token t_k with hidden states h_k.

$$h_k = \left[\overrightarrow{h_k}; \overleftarrow{h_k}; BERT_k\right] \tag{3}$$

The pretrained BERT embeddings by Devlin et al. [12] use the Transformer framework by Vaswani et al. [14] which contains multiple encoders. Because each encoder attains different levels of contextual information, we decide to utilize stack of 3 hidden states of 3 encoders to generate $BERT_k$.

$$BERT_k = \theta \sum_{l=1}^{l=L-1} u_j h_{k,l}^{L,M} \tag{4}$$

θ is a scale parameter showing how important pretrained BERT embeddings are. L is the number of utilized encoders. Vector $u = [u_0; \ldots; u_{L-1}]$ represents softmax-normalized weights that are suitable for each hidden state. $h_{k,l}^{L,M}$ is the hidden state of encoder l at time k. At last, we apply a bi-LSTM layer to attaining contextual information of sentence. The dimension of this layer is D_{sl} and its forward and backward hidden states of each word are concatenated as $f_k \in R^{2D_{sl}}$.

2.1.3. *Detection Network.* Surrounding each token position in an input sentence, the Detection Network generates at most R word segments, called proposals. Every input sentence is considered to be a sequence of tokens, starting from index 1. Proposals with invalid token index, like 0 or larger than the length of input sentence, will be deleted. For example, in Figure 3 if we choose R = 6, token t_3 will generate a set of 6 proposals: (t_3), (t_3, t_4), (t_2, t_3, t_4), (t_2, t_3, t_4, t_5), $(t_1, t_2, t_3, t_4, t_5)$, $(t_1, t_2, t_3, t_4, t_5, t_6)$.

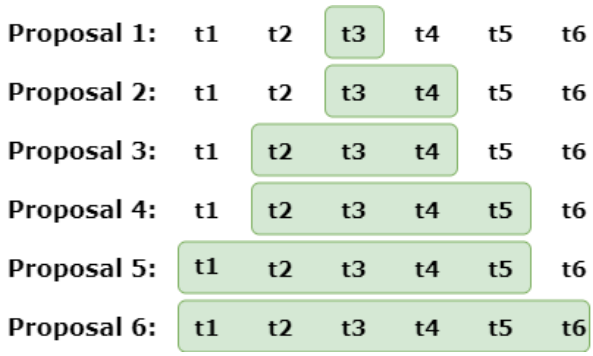

FIGURE 3. All possible entity proposals with R = 6 and position at t_3

Every proposal in the proposal set of each token is simultaneously calculated both the probability of it to be an entity and a non-entity. This calculation is accomplished by a fully connected layer with a two-class softmax function:

$$s_k = softmax(f_k W_p + b_p) \tag{5}$$

where $W_p \in \mathbb{R}^{2D_{sl} \times 2R}$ and $b_p \in \mathbb{R}^{2R}$ are weights and the bias for the entity proposal layer; s_k contains $2R$ scores including R scores for being an entity and R scores for not being

an entity at position k. The Focal Loss is implemented as follows:

$$L_p = -\sum_{k=1}^{K}\sum_{r=1}^{R} \alpha_t y_k^r (1 - s_k^r)^\gamma \log(s_k^r) \tag{6}$$

y_k^r presents proposal type r at position k. s_k^r is the predicted probability of being type r of proposal at position k. α_t is the scale parameter for different types:

$$\alpha_t = \begin{cases} \alpha, & \text{if the proposal is entity} \\ 1-\alpha, & \text{if the proposal is not entity} \end{cases} \tag{7}$$

With $\alpha \in [0, 1]$, α_t helps us solve the imbalance between positive samples and negative samples. In reality, the number of negative entities is much larger than positive entities, which motivates us to choose $\alpha < 0.5$. Additionally, $(1 - s_k^r)^\gamma$ is small with easy-classified samples which have high probability. Therefore, our model can concentrate on analyzing hard-classified samples. The Focal Loss solves issues that Cross Entropy cannot.

2.2. **The Classifier.** The Classifier inspects the outputs of the Detector to place them in predefined categories. For non-overlapping entities, we utilize Non-Maximum Suppression (NMS) algorithm in Neubeck and Gool [15] to filter. In general, NMS picks out proposals with the highest probability and deletes conflicting ones. The result of NMS is fed into the Classifier as input.

We enhance the significance of context using a self-attention mechanism to properly categorize every entity proposal. The framework of the Classifier consists of three components: Word Processor, Entity Processor and Classification Network. The Word Processor, similar to the Detector's, generates word representations for the proposals. Next, the Entity Processor transforms the word representation and sentence representation from the Sentence Processor of the Detector to produce entity representation. Finally, the Classification Network pigeonholes proposals into predefined categories.

2.2.1. *Word Processor.* The architecture of the Word Processor is precisely the same as that of the Detector. The word-level embedding is also the concatenation of pretrained word embeddings and POS tag if it exists. Therefore, this embedding is directly transferred from the Detector to optimize computational resources. Character-level information is obtained the same way as it is in the Detector. However, the training process for character-level information completed separately. Ultimately, word representation for each proposal, likewise, is the concatenation of the word-level embedding and character-level information.

2.2.2. *Entity Processor.* The first two layers of the Entity Processor is similar to those of the Sentence Processor in the Detector. The obtained word representation is fed into a bidirectional LSTM with hidden size D_{wl} and the hidden states are concatenated with BERT embeddings as entity features. Subsequently, another bidirectional LSTM called Entity LSTM is applied to capturing sequence information among the entity words. The last hidden states are concatenated to create the entity representation $e \in \mathbb{R}^{2D_{el}}$. In addition, the sentence representation in the Sentence Processor of the Detector is utilized and transferred directly to the Classifier as context feature C.

To improve performance, we always consider context information in the learning process. As an attempt to emphasize the context surrounding entity proposals, we propose a self-attention mechanism to highlight relevant context words and learn more accurate context information. Both the entity representation e and context feature C are input for the self-attention layer to yield a vector of attention weights a as output:

$$a = softmax\left(CWe^T\right) \tag{8}$$

where $W \in \mathbb{R}^{2D_{sl} \times 2D_{el}}$ is a weight matrix for the self-attention layer, and a is the self-attention weight on different context words. To strengthen the focus on entity-related context, the attentive vector C^{att} is calculated as the attention-weighted context:

$$C^{att} = a * C \tag{9}$$

In addition, we must realign the shape of the output as the length of C^{att} varies for different inputs. To this end, we apply an Attention LSTM with the hidden dimension D_{ml}. The concatenation of the last hidden states in the forward and backward LSTM layer as the context representation $m \in \mathbb{R}^{2D_{ml}}$. The context representation and the entity representation are concatenated together as a context-aware entity representation $o = [m; e]$.

2.2.3. *Classification Network.* To classify obtained entity representation o into predefined categories, we use a two-layer fully connected neural network:

$$p = softmax(W_{c_2}(\sigma(oW_{c_1} + b_{c_1})) + b_{c_2}) \tag{10}$$

in which, $W_{c_1} \in \mathbb{R}^{(2D_{ml}+2D_{el}) \times D_h}$, $b_{c_1} \in \mathbb{R}^{D_h}$, $W_{c_2} \in \mathbb{R}^{2D_{c_1} \times (D_t+1)}$, $b_{c_2} \in \mathbb{R}^{D_t+1}$ are the weights for this fully connected neural network, and D_t is the number of entity types. We add a type for non-entity; hence there are $(D_t + 1)$ types. The square hinge-ranking loss is used:

$$L_c = \sum_{y_w \in Y_w} \max(0, \Delta + p_{y_w} - p_{y_r})^2 \tag{11}$$

where p_w is the probability for the wrong labels y_w, p_r is the probability for the right label y_r, and Δ is a margin. The square hinge-loss accentuates the probability for the right label over the wrong labels, thus improving the accuracy.

3. **Experiments.**

3.1. **Datasets.** For English, we use the fine-grained nested dataset NNE by Ringland et al. [16] over the full Wall Street Journal portion of the Penn Treebank. This dataset has 260,386 mentions of 114 entity types, which include named entities, all time and date and numerical entities with up to 6 layers of nesting. In NNE dataset, for 118,525 top-level entity mentions, 47,020 (39.6%) do not have any nested structure embedded. It is worth noting that one entity can be labeled with multiple types.

To evaluate our model in Vietnamese, we use the dataset VLSP-2018 by Huyen et al. [17], which has both nested and non-overlapping entities. There are four types of entities: person, organization, location and miscellaneous, nested in at most 3 layers. Due to the nature of Vietnamese, the same word can be labeled differently or not labeled as an entity in different context. In addition, to simplify the task, we segmented and tokenized the dataset, using Vietnamese toolkit[1], to make sure each word only takes up on token position. This dataset does not contain POS tags, so we have to add POS tags to it.

The corpora statistics for both datasets can be seen in Table 1. As presented, most entites have the length less than or equal to 6; thus we choose R as 6.

3.2. **Implementation.** For the English dataset, we keep all hyper-parameters the same as that of Xia et al. [10] except: $BERT_k$ is generated by a stack of 3 hidden states of the 9^{th} encoder, the 10^{th} encoder and the 11^{th} encoder of pretrained BERT embeddings from Wolf et al. [18], $\alpha = 0.65$ and $\gamma = 2$.

For the Vietnamese dataset, we have a few modifications of hyper-parameters: $BERT_k$ is generated by the same ordered hidden states but trained on multi-lingual language including Vietnamese by Wolf et al. [18] and its dimension is 768, the pretrained word embeddings are from fastText in Grave et al. [19] with dimension 300, $\alpha = 0.65$, $\gamma = 2$. To compare the effectiveness between pretrained BERT embeddings and pretrained ELMo

[1]http://www.clc.hcmus.edu.vn/?page_id=471&lang=en

Table 1. Datasets statistics

Dataset		NNE			VLSP-2018		
		train	dev	test	train	dev	test
Sentences with NE	Total	35279	1603	3050	8980	2968	3886
	overlap	24402	1043	2074	1318	558	443
Entities	Total	248136	10463	21196	20440	7221	8131
	Overlap	116606	4584	9765	1657	700	510
	Length ≤ 6	247155	10429	21075	20046	7062	8036
	Max length	15	14	14	15	11	11

embeddings, we also use the pretrained ELMo embeddings trained on Vietnamese in Che et al. [20].

3.3. **Evaluation metrics.** We evaluate named entity recognition with 3 metrics: Recall (R), Precision (P), and F1-score (F1):

$$P = \frac{\text{number of correctly predicted named entities}}{\text{number of predicted named entities}} \tag{12}$$

$$R = \frac{\text{number of correctly predicted named entities}}{\text{number of reference named entities}} \tag{13}$$

$$F_1 = \frac{2 \times P \times R}{P + R} \tag{14}$$

3.4. **Result and analysis.** For English, because the MGNER model is state-of-the-art in NER task and its performance was compared with other old models by Xia et al. [10], we just show our improvement for the MGNER model. We can see that our model with BERT and Focal Loss gets the better result of F1 score compared to that of our model with ELMo and Focal Loss by 1.1% in Table 2. Pretrained BERT embedding shows its effectiveness compared to pretrained ELMo embedding as a result of truly bidirectional contextual information and the solution of out-of-vocabulary by subwords. Besides, we also conduct experiments on the VLSP-2018 dataset and our model with BERT and Focal Loss outperforms the latest feature-based model by Pham [9]. Additionally, we also compare the effectiveness between Focal Loss and Cross Entropy in the Detector. The Detector with Focal Loss gets the better result of F1 score with 97.31% compared to F1 score 95.62% of Cross Entropy on NNE dataset in Table 3. The same result appears in VLSP-2018 dataset. It proves that Focal Loss is effective in solving imbalance data between negative and positive entities.

Table 2. Results on NNE and VLSP-2018 datasets

Model	NNE			VLSP-2018		
	P	R	F1	P	R	F1
Feature-based model	–	–	–	78.0	71.69	74.70
MGNER (ELMo + Focal Loss)	89.56	78.23	83.51	72.2	68.9	70.51
MGNER (BERT + Focal Loss)	90.26	79.62	84.61	79.95	78.35	79.14

Table 3. Results of the Detector on NNE and VLSP-2018 datasets

Model	NNE			VLSP-2018		
	P	R	F1	P	R	F1
MGNER (BERT + Cross Entropy)	92.63	98.81	95.62	69.54	92.80	79.51
MGNER (BERT + Focal Loss)	97.29	97.33	97.31	90.79	88.47	89.61

In Table 4, we show the performance of our model and feature-based model in non-overlapping entities and nested named entities. Our model gets better results in both non-overlapping and nested entities proving the effectiveness of our proposed method thanks to BERT and Focal Loss.

TABLE 4. Results on non-overlapping and overlapping entities in VLSP-2018

Model	Overlapping			Non-overlapping		
	P	R	F1	P	R	F1
Feature-based model	44.56	82.51	57.87	73.95	79.33	76.55
MGNER (BERT + Focal Loss)	95.51	45.69	61.72	89.36	80.54	84.72

4. **Conclusions.** This paper presents improvements for the MGNER model, which leverages BERT and Focal Loss to enhance performance. The replacement of BERT over ELMo enables the model to get the truly bidirectional contextual information and to solve out-of-vocabulary issue with subwords. Focal Loss solves the imbalance between positive and negative samples, which allows the model to focus on hard-classified samples. Our model shows improvement compared to the MGNER model in NNE dataset and outperforms other models in the VLSP-2018 dataset.

In the future, we will try to streamline our model in order to decrease the time training and find effective methods to improve it.

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

REFERENCES

[1] L. Ratinov and D. Roth, Design challenges and misconceptions in named entity recognition, *Proc. of the 13th Conference on Computational Natural Language Learning*, pp.147-155, 2009.
[2] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami and C. Dyer, Neural architectures for named entity recognition, *Proc. of NAACL-HLT*, pp.260-270, 2016.
[3] G. Zhou, J. Zhang, J. Su, D. Shen and C. Tan, Recognizing names in biomedical texts: A machine learning approach, *Bioinformatics*, pp.1178-1190, 2004.
[4] B. Alex, B. Haddow and C. Grover, Recognising nested named entities in biomedical text, *Proc. of BioNLP*, 2007.
[5] J. R. Finkel and C. D. Manning, Nested named entity recognition, *Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing*, 2009.
[6] B. Wang and W. Lu, Neural segmental hypergraphs for overlapping mention recognition, *EMNLP*, pp.204-214, 2018.
[7] B. Wang, W. Lu, Y. Wang and H. Jin, A neural transition-based model for nested mention recognition, *EMNLP*, pp.1011-1017, 2018.
[8] T. S. Nguyen and L. M. Nguyen, Nested named entity recognition using multilayer recurrent neural networks, in *Computational Linguistics. PACLING 2017. Communications in Computer and Information Science*, K. Hasida and W. Pa (eds.), Singapore, Springer, 2017.
[9] M. Q. N. Pham, A feature-based model for nested named-entity recognition at VLSP-2018 NER evaluation campaign, *Journal of Computer Science and Cybernetics*, vol.34, no.4, pp.311-321, 2018.
[10] C. Xia, C. Zhang, T. Yang, Y. Li, N. Du, X. Wu, W. Fan, F. Ma and P. S. Yu, Multi-grained named entity recognition, *ACL*, pp.1430-1440, 2019.
[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollr, Focal loss for dense object detection, *IEEE International Conference on Computer Vision (ICCV)*, 2017.
[12] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, *arXiv Preprint arXiv:1810.04805*, 2018.
[13] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, Deep contextualized word representations, *NAACL*, 2018.
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, Attention is all you need, *Advances in Neural Information Processing Systems*, pp.6000-6010, 2017.

[15] A. Neubeck and L. Van Gool, Efficient non-maximum suppression, *The 18th International Conference on Pattern Recognition (ICPR2006)*, vol.3, pp.850-855, 2006.

[16] N. Ringland, X. Dai, B. Hachey, S. Karimi, C. Paris and J. R. Curran, NNE: A dataset for nested named entity recognition in English newswire, *ACL*, pp.5176-5181, 2019.

[17] N. T. M. Huyen, N. T. Quyen, V. X. Luong, T. M. Vu and N. T. T. Hien, VLSP shared task: Named entity recognition, *Journal of Computer Science and Cybernetics*, vol.34, no.4, pp.283-294, 2018.

[18] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest and A. Rush, Transformers: State-of-the-art natural language processing, *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp.38-45, 2020.

[19] E. Grave, P. Bojanowski, P. Gupta, A. Joulin and T. Mikolov, Learning word vectors for 157 languages, *Proc. of the 11th International Conference on Language Resources and Evaluation*, 2018.

[20] W. Che, Y. Liu, Y. Wang, B. Zheng and T. Liu, Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation, *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 2018.