ICIC Express Letters
Part B: Applications
Volume **9**, Number **5**, May **2018**

ICIC International ©2018 ISSN 2185-2766
pp. 375–380

IMPROVING A NEURAL NAMED ENTITY MODEL WITH LINGUISTIC FACTORS

Hai Nguyen[1], Long Nguyen[2], Tan Le[3], Phuoc Tran[4,*] and Huu Nguyen[5]

[1]Faculty of Physics
Ho Chi Minh City University of Pedagogy
No. 280, An Duong Vuong Street, Ward 4, District 5, Ho Chi Minh City, Vietnam
hainm@hcmup.edu.vn

[2]CLC Lab
Faculty of Information Technology
VNU-HCM University of Science
No. 227, Nguyen Van Cu Street, Ward 4, District 5, Ho Chi Minh City, Vietnam
long.hb.nguyen@gmail.com

[3]Faculty of Computer Science
Université du Québec à Montréal
201, Avenue du President-Kennedy, Local PK 4150, H2X 3Y7 Montreal, Quebec, Canada
le.ngoc_tan@courrier.uqam.ca

[4]NLP-KD Lab
Faculty of Information Technology
Ton Duc Thang University
No. 19, Nguyen Huu Tho Street, Tan Phong Ward, District 7, Ho Chi Minh City, Vietnam
*Corresponding author: tranthanhphuoc@tdt.edu.vn

[5]Faculty of Information Technology
Ho Chi Minh City University of Food Industry
No. 140, Le Trong Tan Street, Tay Thanh Ward, Tan Phu District, Ho Chi Minh City, Vietnam
huunt@cntp.edu.vn

Received October 2017; accepted January 2018

Abstract. *This paper reports the use of word-level linguistic factors to a neural model for named entity recognition tasks. Several word-level linguistic factors, such as lemmas, word clusters, part-of-speech tags, and syntactic chunk tags, were combined in embedding layers. These factors were incorporated into a bidirectional recurrent neural model. The experiments showed that our proposed method obtained a better performance with a significant gain compared to a baseline model. Adding these linguistic factors allowed the neural model work better in case of data sparseness or language ambiguity problems.*
Keywords: Named entity recognition, Neural networks, Linguistic factors

1. Introduction.

Named entities (NEs), especially person names (PER), location names (LOC), and organization names (ORG), have a very important role in many natural language processing (NLP) applications (e.g., information extraction, information retrieval, and machine translation). NE recognition (NER) is not only a fundamental task but also one of the most challenging tasks in NLP research area. Since 1995, there have been many NER systems developed for many languages [1] with the domination of supervised learning methods such as support vector machine (SVM) [2] and conditional random field (CRF) [3].

NER by using neural network models has received much attention in recent years because these models have shown remarkable results and improvements over conventional NER models [4]. These models usually take advantages of using long short term memory (LSTM) [5] models which improve recurrent neural network (RNN) [6] models by their ability to capture long sequence context of sentences. Recent neural NER methods [4, 7, 8] captured the context of a sentence in both directions of the sentence using forward and backward LSTM models. Their methods are called bidirectional LSTM (B-LSTM) and have shown significant improvements in NER tasks. There were methods which tried to adapt convolutional neural network models [9] or CRF algorithms [4] to B-LSTM models. Collobert et al. proposed a CNN model [10] which can learn proper characteristics of words to automatically extract features with a CRF layer on the top. A similar architecture was introduced in [11] by adding more spelling features. It was later improved with character-level embeddings [4]. An architecture that used a stack of LSTM layers was also presented in [4]. The above methods used B-LSTM layers as a key part in their architecture augmented with some features such as pre-trained word embedding layers and dropout layers. Even these B-LSTM models have shown promising results, they are lack of ability of using linguistic information for surface words. The additional linguistic information can help to resolve language ambiguity or data sparseness problems which usually occur when training data are not large enough. This idea is inspired by one of previous successful methods in machine translation [12], where the model incorporates various linguistic annotations for the surface level words.

This paper presents a novel method to incorporate additional linguistic factors at word level to a B-LSTM neural network model for the English NER task. We experimented with four word-level factors including lemmas, word clusters, part-of-speech (POS) tags, and syntactic chunk tags. The experiments illustrated that not only the combination of these factor but also each individual factor significantly improves the B-LSTM model performance with impressive results on an English data set. Our main contribution is to find a best method that is able to represent the linguistic factors in word representation as well as to input the factors into the B-LSTM neural network model to resolve language ambiguity or data sparseness problems.

This method shows several advantages. Firstly, these linguistic factors are easy to achieve for different languages. For languages that POS tagging or syntactic chunking toolkits does not exist, word clusters or word patterns can be used. Then, these factors can resolve language ambiguity or data sparseness problems. Moreover, these factors are at word-level and are easy to combine into common word representation.

The remainder of this paper is organized as follows. The proposed method is presented in Section 2. Then the experiments and the evaluation on the data are reported in Section 3. Finally, Section 4 presents some conclusions and future work.

2. Proposed Method.

In this study, we investigate the possibility to integrate linguistic factors to a neural network model for NER tasks. As an initial work, our purpose is to find how additional linguistic factors can resolve data sparseness and language ambiguity problems in a specific B-LSTM model. Assume that we have a sentence containing N words; each word is augmented with L linguistic factors. Then, we can express the sentence as $\left\{\left(\left\{x^{(n,l)}\right\}_{l=0}^{L}, y^{(n)}\right)\right\}_{n=1}^{N}$.

LSTMs take an input of a sequence of vectors $(x_1, x_2, \ldots, x_n)$ and produce an output of a sequence of vectors $(h_1, h_2, \ldots, h_n)$ to represent the information at each input step. LSTMs have been designed to incorporate a memory cell which can protect and control the cell state. They use several gates to control the amount of information from the previous states which should be forgotten and the information from the inputs which should be updated to the memory cell [13]. There are many variants of LSTM implementations. In this paper, we reuse the implementation of Lample et al. [4]:

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \tag{1}$$

$$c_t = (1 - i_t) \odot c_{t-1} + i_t \odot \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{2}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \tag{3}$$

$$h_t = o_t \odot \tanh(c_t) \tag{4}$$

where σ is the element-wise sigmoid function, and $\odot$ is the element-wise product. c_t and o_t are the cell state and the output at the step t, respectively.

Figure 1 illustrates an LSTM cell in an LSTM neural network. In Equation (1), i_t is a sigmoid layer to decide which values are updated. x_t is the input value of the cell at an input step. h_{t-1} and c_{t-1} are the hidden state and the cell state of the previous step. In Equation (2), c_t is where the LSTM cell decides what previous information to forget and what new information should be added to. In Equation (3), o_t is a sigmoid layer which decides what parts of the cell state should be outputted. Finally, in Equation (4), h_t is the decided output which is taken from the multiplication between o_t and the tanh function of the cell state. W_{xx} and b_x in the equations are parameters of the LSTM network, which are needed to learn from training data.

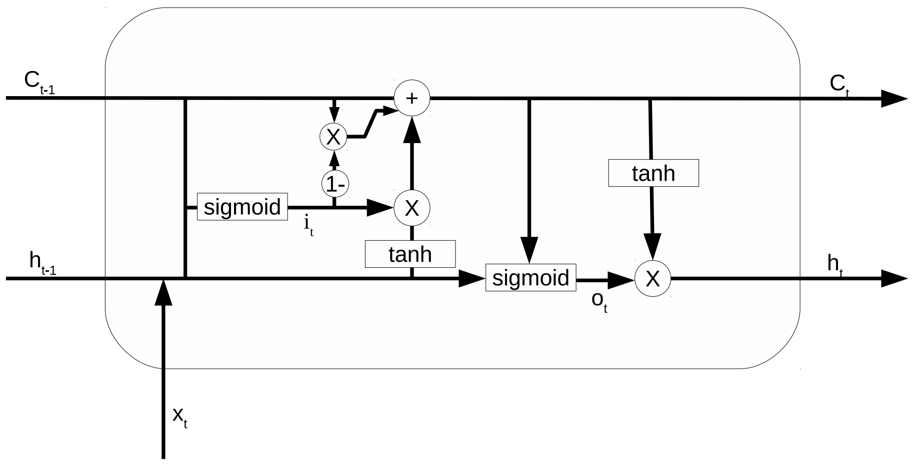

Figure 1. An LSTM cell

A B-LSTM which contains a forward LSTM and a backward LSTM operates on a sequence in forward and backward directions, respectively. If $\overrightarrow{h_j}$ is the summary representation of a word at position j-th from the left of a sequence, $\overleftarrow{h_j}$ will be the summary representation of that word starting from the right of the sequence. It means the combination of $h_t = \left[\overrightarrow{h_j}, \overleftarrow{h_j}\right]$ summarizes the representation of the whole sequence.

Figure 2 shows the architecture of our proposed model with integrating linguistic factors to the B-LSTM model. At the beginning, the input layer is a concatenation of several embedding layers (i.e., L layers) encoding the linguistic factors to word vectors. It means that these embedding layers are word, lemma, word cluster, pos, chunk embedding layer which are trained separately on each factor. This can help to improve our NER model performance when these embedding layers are pre-trained with large corpora. At each input step, word vector, lemma vector, cluster vector, pos vector and chunk vector are concatenated to establish a common vector (i.e., an input vector). Secondly, these input vectors are inputted into the forward LSTM layer and the backward LSTM layer. Because an LSTM summarizes a sequence by reading one symbol at a time, $\overrightarrow{h_j}$ of the forward LSTM summarizes the sequence up to the j-th word beginning from the first word and $\overleftarrow{h_j}$ of the backward LSTM summarizes the sequence up to the j-th word beginning from the last word. In other words, a context dependent vector $h_t = \left[\overrightarrow{h_j}, \overleftarrow{h_j}\right]$ together summarizes the whole input sequence. As a result, the prediction of an NE label has the context information of the whole input sequence instead of having only a small context information due to the limit of context window size in traditional classifiers (e.g., SVM, and CRF).

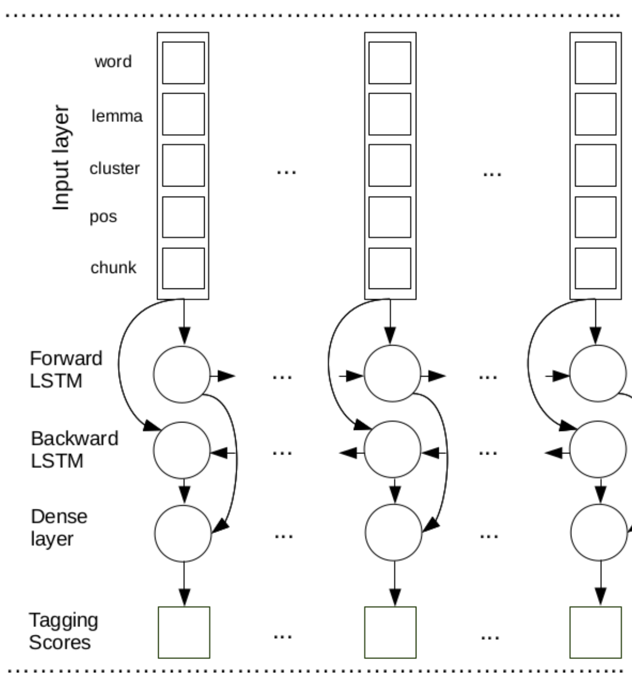

FIGURE 2. Our neural network based named entity architecture

Eventually, these context vectors are inputted to a dense layer which is a regular densely-connected neural network layer with a softmax activation function to produce tagging scores. Because the output of the softmax function represents a categorical probability distribution, the label which has the highest tagging score is assigned to the word.

3. **Experiments.**

3.1. **Training configuration.** We used Keras[1] library to implement all our experiments. The baseline was a simple B-LSTM model with an embedding layer. All neural models were configured with 128 embedding layer dimensions and 100 hidden layer dimensions. For the linguistic factors, we used 64 embedding layer dimensions to encode each linguistic factor including word lemmas, word clusters, POS tags, and syntactic chunk tags. The word lemmas and word clusters were collected by NLTK toolkit[2] and Brown Cluster[3], respectively.

To train our neural networks, we used stochastic gradient descent (SGD) with a fixed learning rate of 0.01. The batch size was 32 and the number of epochs was 200. The brown cluster size was 100.

For evaluation of our neural networks, we used the CoNLL script[4]. To test the statistical significance, we used the bootstrapping resampling method [14] to measure the significant level ($p < 0.01$) of F-score differences between the neural models.

3.2. **Data sets.** We conducted our experiments on CoNLL-2003 English dataset[5]. Because the dataset already contains POS tags and syntactic chunk tags, we only need to add word lemmas and word clusters factors. For the training step, we used the training set containing 14,985 sentences. We evaluated on the *testa* and *testb* testing set. Table 1 provides statistics on the dataset.

[1]https://keras.io/
[2]http://www.nltk.org/
[3]https://github.com/percyliang/brown-cluster
[4]http://www.cnts.ua.ac.be/conll2002/ner/bin/conlleval.txt
[5]http://www.cnts.ua.ac.be/conll2003/ner/

Table 1. Statistics of the English dataset

Dataset	#tokens	#types	#sentences
train	204,562	23,624	14,985
testa	51,573	9,966	3,464
testb	46,624	9,485	3,682

The dataset contains columns separated by a single space. Table 2 illustrates linguistic factors in an English sentence. The first item on each line is a word, the second a lemma, the third a word cluster, the fourth a POS tag, the fifth a syntactic chunk tag and the sixth the NE tag. The NE tag is one of classes including PER, LOC, ORG, or MISC (Miscellaneous).

Table 2. Example of linguistic factors

word	lemma	word cluster	pos	chunk	NE
Only	Only	011000111	RB	I-NP	O
France	France	111001	NNP	I-NP	I-LOC
and	and	0011110	CC	I-NP	O
Britain	Britain	111001	NNP	I-NP	I-LOC
backed	back	0110011011	VBD	I-VP	O
Fischler	Fischler	0110100	NNP	I-NP	I-PER
’s	’s	001110	POS	B-NP	O
proposal	proposal	01010111	NN	I-NP	O
.	.	0010	.	O	O

3.3. **Results.** As mentioned previously, the aim of the tests was to evaluate the effectiveness of integrating linguistic factors to a B-LSTM model. For that reason, we used a pure B-LSTM model as a baseline. This section presents our comparisons for different experiments on English NER.

Table 3 shows the scores reported in F_1 score. It is clear to realize that either the individual factor or the combination of factors will improve the performance of the B-LSTM model. The word lemma factor reduced the data sparseness problem in an inflectional language such as English. Other remaining factors resolved the language ambiguity problems. We observed that the combination of all the factors gave us a biggest improvement in overall performance of $(+7.49, +10.74)$. For each factor, the POS factor gave us a particular increase of $(+4.46, +8.05)$; the word cluster factor marked an increase of $(+2.80, +5.60)$, while the chunk factor resulted in a difference of $(+1.72, +1.36)$ and finally the lemma factor resulted in an increase of $(+1.13, +1.38)$.

Even these factors helped to overcome the data sparseness and language ambiguity problems, they could not resolve the difficulty in classifying NEs to their right classes. For

Table 3. NER results on the English dataset

Models	testa	testb
B-LSTM (baseline)	74.41	64.92
B-LSTM + lemma	75.54	66.30
B-LSTM + word cluster	77.21	70.52
B-LSTM + pos	78.87	72.97
B-LSTM + chunk	76.13	66.28
B-LSTM + all factors	81.90	75.66

example, sometimes, a person name can be used to indicate a location or an organization. In the test set, *Charles de Gaulle* was manually tagged as an LOC name but the model predicted *Charles* in that location phrase as a PER name.

4. Conclusion and Future Work.

Prior work has indicated the effectiveness of using B-LSTM models in NER tasks. However, these studies have not focused on the advantages of linguistic factors which can help to solve the language ambiguity and data sparseness problems. In this paper, we studied the importance of additional linguistic factors in a B-LSTM model. We found that in all cases, adding either a linguistic factor or a combination of linguistic factors substantially increases the performance of a B-LSTM model. This study, therefore, proves the benefits of adding linguistic information to a neural network model in NLP research area. In addition, these factors used in our study are easy to establish via existing toolkits. Most notably, this is the first study, according to our knowledge, to investigate the effectiveness of incorporating linguistic factors to a neural network for the English NER tasks.

However, in this paper, we only integrated the linguistic factors to embedding layers by concatenating linguistic vectors. In the future work, we will investigate other possible methods to integrate more effectively these factors. Besides that, we will also examine the possibility of applying additional layers such as CNN layer, CRF layer, or dropout layer into our proposed method.

REFERENCES

[1] D. D. Palmer and D. S. Day, A statistical profile of the named entity task, *Proc. of the 5th Conference on Applied Natural Language Processing (ANLC'97)*, Stroudsburg, PA, USA, pp.190-193, 1997.

[2] C. Cortes and V. Vapnik, Support-vector networks, *Machine Learning*, vol.20, no.3, pp.273-297, 1995.

[3] J. D. Lafferty, A. McCallum and F. C. N. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *Proc. of the 18th International Conference on Machine Learning (ICML'01)*, San Francisco, CA, USA, pp.282-289, 2001.

[4] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami and C. Dyer, Neural architectures for named entity recognition, *HLT-NAACL*, 2016.

[5] F. A. Gers, J. A. Schmidhuber and F. A. Cummins, Learning to forget: Continual prediction with LSTM, *Neural Comput.*, vol.12, no.10, pp.2451-2471, 2000.

[6] C. Goller and A. Kchler, Learning task-dependent distributed representations by backpropagation through structure, *Proc. of IEEE International Conference on Neural Networks*, pp.347-352, 1996.

[7] J. P. C. Chiu and E. Nichols, Named entity recognition with bidirectional LSTM-CNNs, *CoRR*, http://arxiv.org/abs/1511.08308, 2015.

[8] Y. Yao and Z. Huang, Bi-directional LSTM recurrent neural network for Chinese word segmentation, *CoRR*, http://arxiv.org/abs/1602.04874, 2016.

[9] O. Kuru, O. A. Can and D. Yuret, Charner: Character-level named entity recognition, *The 26th International Conference on Computational Linguistics, Proc. of the Conference: Technical Papers*, Osaka, Japan, pp.911-921, 2016.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa, Natural language processing (almost) from scratch, *J. Mach. Learn. Res.*, vol.12, pp.2493-2537, 2011.

[11] Z. Huang, W. Xu and K. Yu, Bidirectional LSTM-CRF models for sequence tagging, *CoRR*, http://arxiv.org/abs/1508.01991, 2015.

[12] P. Koehn and H. Hoang, Factored translation models, *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic, 2007.

[13] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, vol.9, no.8, pp.1735-1780, 1997.

[14] P. Koehn, Statistical significance tests for machine translation evaluation, *Proc. of EMNLP 2004*, Barcelona, Spain, pp.388-395, 2004.