# Neural Model for Named Entity Recognition Considering Aligned Representation

Hongyang Sun  Taewhan Kim
Dept. of Electrical and Computer Engineering, Seoul National University

**Abstract**

Sequence tagging is an important task in Natural Language Processing (NLP), in which the Named Entity Recognition (NER) is the key issue. So far the most widely adopted model for NER in NLP is that of combining the neural network of bidirectional long short-term memory (BiLSTM) and the statistical sequence prediction method of Conditional Random Field (CRF). In this work, we improve the prediction accuracy of the BiLSTM by supporting an aligned word representation mechanism. We have performed experiments on multilingual (English, Spanish and Dutch) datasets and confirmed that our proposed model outperformed the existing state-of-the-art models.

## 1. Introduction

Named Entity Recognition (NER) (also known as entity chunking, sequence labeling, sequence tagging) is a subtask of Natural Language Processing (NLP) that seeks to locate and classify named entities in text into pre-defined categories such as the names of persons, organizations, locations, etc. Traditional method to handle these problems is done by using dictionary lookups and handcrafting these linguistic data. However, the long consuming time makes it desperate to deal with large data. Recently, recurrent neural network (RNN) such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) was proposed to empower the model.

On the other hand, it has been validated that statistical models such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs), have a huge advantage in solving the problem of NER due to the high effectiveness of capturing the relevance among the data. Consequently, a lot of works combining Neural Networks (NN) with CRF have been proposed to tackle the tagging problem and proved their effectiveness. In particular, some works (eg., Bharadwaj et al., 2016 [1]) utilized the phonological information of languages to deal with NER. Additionally, during the development of NLP, word embedding methods like word2vec [2] and GloVe [3] provide a new perspective of the representation of words, but still cannot handle the out-of-vocabulary (OVV) words. In this work, we propose a character-level embedding rather than the conventional word-level only embedding for NER to improve the prediction accuracy. By concatenating character-level representation and word embedding together, it is expected that the model is able to achieve a much better prediction accuracy. Note that there are a few works that considered the character-level representation (eg., Yang et al., 2016 [4], Peters et al., 2017 [5]), but they rely on pre-training of an external corpus, thus require a bigger model and longer training. For example, one of its language models was trained on 32 GPUs for half a month, which lacks efficiency.

Contrary to the existing character-level based methods, we construct an effective and efficient neural network architecture for NER and chunking task that doesn't need extra training resources. We combine a character-level BiLTSM and word embedding together with an attention alignment to extract the information of the corpus, then input them to a BiLSTM for sequence tagging. Lastly, we applied CRF to predict the tagging results. We implemented our proposed model and compared the results with that produced by the existing models, and confirmed that ours outperformed the others.

We conducted the experiment on the CoNLL 2000 English chunking task, CoNLL 2002 Spanish and Dutch NER task, and CoNLL 2003 English NER task.

In the following section, we review the existing models and processing mechanism, together with our update to support our proposed model.

## 2. The Proposed Model

Our model consists of three parts: (1) character-level BiLSTM, (2) the attention mechanism, and (3) word-level BiLSTM-CRF.

### *A. Character-level BiLSTM*

The first part of our model is based on character-level, which is trained directly with the training dataset to capture the linguistic characteristics of words. To avoid the model being too large, instead of performing prediction for every character in one word, we performed the prediction only for the start and end characters in each word. By concatenating the last output of forward LSTM and the first output of the backward LSTM, we produce a word's character-level representation like the following:

$$\overrightarrow{h_t} = \mathrm{LSTM}\left(x, \overrightarrow{h}_{t-1}\right), \overleftarrow{h_t} = \mathrm{LSTM}(x, \overleftarrow{h}_{t+1}) \quad (1)$$

$$\mathbf{c} = [\overrightarrow{h_t}\,;\, \overleftarrow{h_t}] \quad (2)$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the hidden states of forward and backward LSTMs, respectively. $\mathbf{c}$ is the character-level representation of the word. The corresponding architecture is shown in Fig.1.

### *B. Attention Mechanism*

The concept of attention [6] was first proposed in the field of Neural Machine Translation (NMT). The core thought is to align the hidden states in the encoder and to compute a weighted sum of these hidden states as a context vector. This enables the encoder model to capture more information of the whole original sentence and was validated to be able to achieve a much better performance. In this work we followed

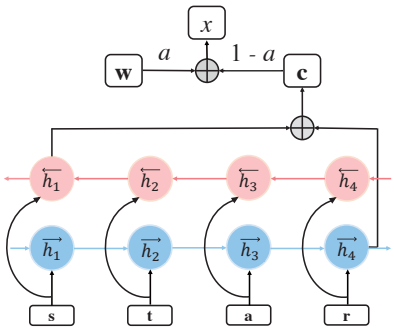


Fig.1: Character-level BiLSTM and attention mechanism. ⊕ represents concatenation. Characters pass through a two-direction LSTM first and then are concatenated as **c**, **c** is the character-level representation of the word "star", **w** is the word embedding of "star". By multiplying the aligning weight, they are combined to produce the final representation.

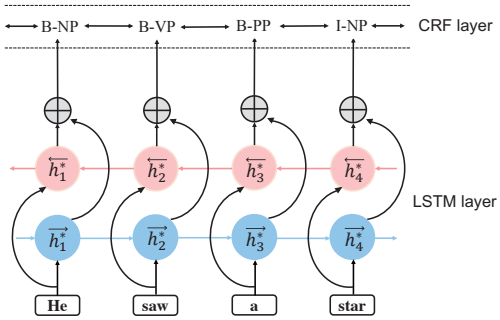


Fig. 2: BiLSTM-CRF architecture. ⊕ represents concatenation. After obtaining the representation of a word, we input it into a word-level BiLSTM, concatenate the output of the two LSTMs together, and then input them to a CRF layer, which is able to predict the label of the word.

the core concept and applied the attention alignment into our model. Instead of directly concatenating the character-level LSTM outputs and word embedding, we align a coefficient $a$ to dynamically decide the proportion of these two representations. For an input sentence, suppose **w** represents the word embedding matrix and **c** represents the character-level embedding matrix, then we calculate the aligning coefficient by:

$$a = \sigma\left(\mathrm{W}_3 g(\mathrm{W}_2\mathbf{w} + \mathrm{W}_1\mathbf{c})\right) \tag{3}$$

where $g$ is a nonlinear function such as *ReLU* or *tanh* and $\sigma$ is a sigmoid function to restrict $a$ between 0 and 1, Three *W*s are weight matrices. And finally the combined input **x** is computed by a concatenation:

$$\mathbf{x} = [a \cdot \mathbf{w}; (1-a) \cdot \mathbf{c}] \tag{4}$$

This enhances the flexibility of word expressions. For example, if the word has a regular prefix and suffix, then it learns more from word-level embeddings. On the contrary, if it has rare prefix or suffix, or is even unknown, it pays attention more to the character level. We also adopted this methodology when processing **c**. That is, rather than computing Eq.(2), **c** is actually calculated by aligning a coefficient $a'$:

$$\mathbf{c} = \left[a' \cdot \overrightarrow{h_t}; (1-a') \cdot \overleftarrow{h_t}\right] \tag{5}$$

The attention mechanism along with character-level BiLSTM is described in Fig.1.

### C. *Word-level BiLSTM-CRF*

After obtaining the representation of words, we input them into a BiLSTM. Although we can train a BiLSTM model simply in an NER task, it is impracticable to use this model to capture the relevance between the outputs. However, there are constraints among the labels such as “B-Person I-Person” is valid, and “B-Person I-Organization” is invalid. A CRF layer could add some constraints to the final predicted labels to ensure their validity. These constraints can be learned by the CRF layer automatically from the training dataset during the training process. Consequently, we model tagging decisions jointly with a CRF [7]. For an input sentence:

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \tag{6}$$

we denote the score matrix produced by the BiLSTM with **E**, which is called *Emission Score Matrix*. **E** is of size $n \times k$, where $n$ is the number of words in this sentence and $k$ is the number of tags. The element $E_{i,j}$ corresponds to the score of the $j^{th}$ tag of the $i^{th}$ word in a sentence. For a sequence of predictions:

$$\mathbf{y} = (y_1, y_2, \ldots, y_n) \tag{7}$$

the score is defined as:

$$s(\mathbf{X}, \mathbf{y}) = \sum_{i=0}^{n} T_{y,y_{i+1}} + \sum_{i=0}^{n} E_{i,y_i} \tag{8}$$

where **T** is called *Transition Score Matrix*, representing the score of transferring from one label to another. The element $T_{i,j}$ represents the score of transition from tag $i$ to tag $j$. we added *start* and *end* tags so that **T** is a square matrix of size $k+2$. We can calculate a probability for a predicted sequence **y** by:

$$p(\mathbf{y}|\mathbf{X}) = \frac{e^{s(\mathbf{X},\mathbf{y})}}{\sum_{\hat{\mathbf{y}}\in\mathbf{Y}} e^{s(\mathbf{X},\hat{\mathbf{y}})}} \tag{9}$$

During training, we minimize the negative log-likelihood:

$$-\sum_{i} \log p(\mathbf{y}|\mathbf{X}) \tag{10}$$

While decoding, we predict the output sequence by the formula:

$$\mathbf{y}^* = \underset{\hat{\mathbf{y}}\in\mathbf{Y}}{\operatorname{argmax}}\, s(\mathbf{X}, \hat{\mathbf{y}}) \tag{11}$$

The Viterbi algorithm is used to efficiently calculate Eq.(10). The architecture of BiLSTM-CRF is shown in Fig. 2.

Lastly, motivated by *Highway Networks* ([8], [9]), we apply highway layer both in character-level and word-level BiLSTM to deal with the effect decline when the depth of the network increases. $\mathbf{H}(x)$ is an affine transform of input $x$ followed by a non-linear activation function. $\mathbf{T}(x)$ is defined as a linear transform of $x$ passing through a sigmoid function. Therefore, the value is in between 0 and 1. Eventually, the output of a network is:

Table 1: Dataset statistics

| Benchmark | Task | Language | Training Tokens | Dev Tokens | Test Tokens |
|---|---|---|---|---|---|
| CoNLL 2000 | chunking | English | 211727 | - | 47377 |
| CoNLL 2002 | NER | Spanish | 207484 | 51645 | 52098 |
| CoNLL 2002 | NER | Dutch | 202931 | 37761 | 68994 |
| CoNLL 2003 | NER | English | 204567 | 51578 | 46666 |

Table 2: Chunking performance on CoNLL 2000 dataset (English)

| Model | F1 Score |
|---|---|
| Collobert et al. (2011) [10] | 94.32 |
| Zhai et al. (2017) [11] | 94.72 |
| Akhundov et al. (2018) [12] | 94.74 |
| Lample et al. (2016) [7] | 94.49 |
| Hashimoto et al. (2016) [13] | 95.02 |
| Yang et al. (2016) [4] | 95.41 |
| Peters et al. (2017) [5] | **96.37** |
| **Ours** | 95.12 |

Table 3: NER performance on CoNLL 2002 dataset (Spanish)

| Model | F1 Score |
|---|---|
| dos Santos et al. (2015) [14] | 82.21 |
| Gillick et al. (2015) [15] | 82.95 |
| Akhundov et al. (2018) [12] | 84.36 |
| Lample et al. (2016) [7] | 85.75 |
| Yang et al. (2016) [4] | 85.77 |
| Bharadwaj et al. (2016) [1] | 85.81 |
| **Ours** | **86.93** |

$$y = \mathbf{H}(x) \cdot \mathbf{T}(x) + x \cdot \big(1 - \mathbf{T}(x)\big) \tag{12}$$

Here, the $\cdot$ represents element-wise multiplication. For the extreme situation ($\mathbf{T}(x) = 0$ or 1, we have:

$$y = \begin{cases} x & \text{if } \mathbf{T}(x) = 0 \\ \mathbf{H}(x) & \text{if } \mathbf{T}(x) = 1 \end{cases} \tag{13}$$

In our experiment, the highway network is applied to every LSTM (forward and backward) to enhance the performance in deep network.

## 3. Experiment

### A. Datasets

The experiment was conducted on multilingual datasets including the CoNLL 2000 chunking task, CoNLL 2002 Spanish and Dutch NER task, and CoNLL 2003 English NER task. The description of the datasets is shown in Table 1.

**CoNLL00** chunking defines 11 syntactic chunking types and contains training and test datasets only. Like the previous works in [5], [9], we randomly sampled 1000 sentences in the training data as a developing dataset.

**CoNLL02** NER task was developed to research on Spanish and Dutch. The tags have the same format as that in the chunking task and there are four types of phrases: person names (PER), organizations (ORG), locations (LOC) and miscellaneous names (MISC). The dataset was separated into training, develop and test sets.

**CoNLL03** NER task was developed for research of English and contains four types of phrases same as CoNLL02. It was separated into training, develop and test sets.

Table 4: NER performance on CoNLL 2002 dataset (Dutch)

| Model | F1 Score |
|---|---|
| Lample et al. (2016) [7] | 81.74 |
| Gillick et al. (2015) [15] | 82.84 |
| Yang et al. (2016) [4] | 85.19 |
| Akhundov et al. (2018) [12] | **85.61** |
| **Ours** | 85.57 |

Table 5: NER performance on CoNLL 2003 dataset (English)

| Model | F1 Score |
|---|---|
| Collobert et al. (2011) [10] | 89.59 |
| Huang et al. (2015) [16] | 90.10 |
| Lample et al. (2016) [7] | 90.94 |
| Akhundov et al. (2018) [12] | 91.11 |
| Yang et al. (2016) [4] | 91.20 |
| Peters et al. (2017) [5] | **91.93** |
| **Ours** | 91.07 |

### B. Training

For each task, we trained our networks using the back-propagation algorithm to update the parameters for every training example. There are several alternative algorithms such as Adadelta [17] and Adam [18] that exhibit different convergence characteristics. From the experiments, we found they had quick convergence speed but the final score was relatively low. On the other hand, SGD (stochastic gradient descent) with momentum performed better. As a result, we adopted SGD for training. For the learning rate, we updated it using the reciprocal decay method:

$$\eta_t = \frac{\eta_0}{1 + \rho t} \tag{14}$$

where $\eta_t$ is the learning rate in the $t$-th epoch. $\eta_0$ is set to 0.015 and the decay factor $\rho$ is set to 0.05.

For each of the character-level and word-level BiLSTMs, we applied 1-layer and 2-layer LSTMs separately and set the hidden dimension to 300. Batch size was set to 10 and total epoch was set to 200. We chose the 100-dimension and 300-dimension GloVe [3] word vector for English word embedding. Due to the scarcity of embeddings of multilingual languages, we applied 300-dimension word embedding only for Spanish and Dutch ([19]).

### C. Performance

The performance is evaluated with the F1 score, which is a combined measurement of the precision and recall rates:

$$F1 = \frac{2 \times precision \times recall}{recall + precision} \tag{15}$$

As shown in Table 2 through Table 5, our model performed

Table 6: Training time on CoNLL03 NER task

| Model | F1 Score | Time | Device |
|---|---|---|---|
| Ours | 91.07 | 6.9h | GTX 1080 |
| Peters et al., 2017 | 91.93 | 1412h | Telsa K40 |

Table 7: Effect of layers and word embedding dimension

| Task | 1layer + 100d | 1layer + 300d | 2layer + 100d |
|---|---|---|---|
| CoNLL00 | 94.99 | 94.59 | **95.12** |
| CoNLL03 | 90.95 | 90.83 | **91.07** |

well on all the datasets, especially in Spanish and Dutch's situation. In CoNLL02 task of Spanish, there are several kinds of research based on BiLSTM-CRF (Huang et al., 2015 [16]) or GRU-CRF (Yang et al., 2016 [4]) architecture, and our model achieved the best F1 score. For CoNLL02 task of Dutch, we also achieved the second best result. We want to point out that due to the scarcity of word embeddings of languages other than English, we applied 300-dimension word embedding only for CoNLL02 Spanish and Dutch tasks, additionally, we reduced the batch size to 5 and only adopted 1-layer LSTM to avoid excessively large model. For CoNLL00 and CoNLL03 English task, there are some existing excellent works (eg., Peters et al., 2017 [5] etc.) and our model's performance is close to these state-of-the-art researches.

We implemented the model using PyTorch library and carried out all the experiment on Nvidia GTX 1080 Ti GPU. The training time is shown in Table 6.

Furthermore, for CoNLL00 chunking and CoNLL03 NER task, we evaluated the model of different layer number and word embedding alignment. The result is summarized in Table 7. We find that for the character-level and word-level BiLSTMs, 2-layer network architecture obtains a better performance than 1-layer's architecture. This indicates that appropriately deeper neural network helps enhance the model. While for the 1 layer's situation, 300-dimension word embedding alignment has a faster convergence speed but the final F1 score is worse than 100-dimension embedding alignment. This reveals that a simple and efficient embedding scheme is more productive in the practical situation.

## 4. Conclusion

Neural Network architectures have been applied to deal with sequence tagging tasks like NER and chunking tasks, and have achieved prominent success. However, the excessive reliance on extra pre-training and the shortage of handling rare words degraded the models' effectiveness. In this work, we leveraged the attention mechanism to combine character-level representation with word embedding, and passed it through a BiLSTM-CRF network to accomplish the sequence tagging task. By this, our proposed model was able to avoid external training corpus and tackle the rare words. The evaluation was conducted on 4 multilingual datasets which covered English, Spanish and Dutch. It was confirmed that our results stood out in the leading researches of the relevant field.

## Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (2017R1E1A1A03070465).

## References

[1]A. Bharadwaj, D. Mortensen, C. Dyer, and J. Carbonell, "Phonologically aware neural model for named entity recognition in low resource transfer settings," in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1462–1472.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111–3119.

[3] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.

[4] Z. Yang, R. Salakhutdinov, and W. Cohen, "Multi-task cross-lingual sequence tagging from scratch," arXiv preprint arXiv:1603.06270, 2016.

[5] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi- supervised sequence tagging with bidirectional language models," arXivpreprint arXiv:1705.00108, 2017.

[6] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[7] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C.Dyer, "Neural architectures for named entity recognition," arXiv preprintarXiv:1603.01360, 2016.

[8] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," arXiv preprint arXiv:1505.00387, 2015.

[9] L. Liu, J. Shang, F. Xu, X. Ren, H. Gui, J. Peng, and J. Han, "Empower sequence labeling with task-aware neural language model," arXiv preprint arXiv:1709.04109, 2017.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," Journal of Machine Learning Research, vol. 12, no. Aug, pp. 2493–2537, 2011.

[11] F. Zhai, S. Potdar, B. Xiang, and B. Zhou, "Neural models for sequence chunking." in AAAI, 2017, pp. 3365–3371.

[12] A. Akhundov, D. Trautmann, and G. Groh, "Sequence labeling: A practical approach," arXiv preprint arXiv:1808.03926, 2018.

[13] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many- task model: Growing a neural network for multiple nlp tasks," arXiv preprint arXiv:1611.01587, 2016.

[14] C. N. d. Santos and V. Guimaraes, "Boosting named entity recognition with neural character embeddings," arXiv preprint arXiv:1505.05008, 2015.

[15] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya, "Multilingual language processing from bytes," arXiv preprint arXiv:1512.00103, 2015.

[16] Z. Huang, W. Xu, and K. Yu, "Bidirectional lstm-crf models for sequence tagging," arXiv preprint arXiv:1508.01991, 2015.

[17] M. D. Zeiler, "Adadelta: an adaptive learning rate method," arXiv preprint arXiv:1212.5701, 2012.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[19] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," arXiv preprint arXiv:1607.04606, 2016.