Segment-Level Neural Conditional Random Fields for Named Entity Recognition

Motoki Sato[1,2] **Hiroyuki Shindo**[1,2] **Ikuya Yamada**[3] **Yuji Matsumoto**[1,2]
[1] Nara Institute of Science and Technology
[2] RIKEN Center for Advanced Intelligence Project (AIP)
[3] Studio Ousia
{sato.motoki.sa7, shindo, matsu }@is.naist.jp, ikuya@ousia.jp

Abstract

We present *Segment-level Neural CRF*, which combines neural networks with a linear chain CRF for *segment-level* sequence modeling tasks such as named entity recognition (NER) and syntactic chunking. Our segment-level CRF can consider higher-order label dependencies compared with conventional word-level CRF. Since it is difficult to consider all possible variable length segments, our method uses *segment lattice* constructed from the word-level tagging model to reduce the search space. Performing experiments on NER and chunking, we demonstrate that our method outperforms conventional word-level CRF with neural networks.

1 Introduction

Named entity recognition (NER) and syntactic chunking are segment-level sequence modeling tasks, which require to recognize a *segment* from a sequence of words. A segment means a sequence of words that may compose an expression as shown in Figure 1. Current high performance NER systems use the *word-level* linear chain Conditional Random Fields (CRF) (Lafferty et al., 2001) with neural networks. Especially, it has been shown that the combination of LSTMs (Hochreiter and Schmidhuber, 1997; Gers et al., 2000), convolutional neural networks (CNNs) (LeCun et al., 1989), and word-level CRF achieves the state-of-the-art performance (Ma and Hovy, 2016). Figure 1 shows an overview of the word-level CRF for NER.

However, the word-level neural CRF has two main limitations: (1) it captures only first-order word label dependencies thus it cannot capture segment-level information; (2) it is not easy to incorporate dictionary features directly into a word-level model since named entities and syntactic chunks consist of multiple words rather than a single word. To overcome the limitation of first-order label dependencies, previous work propose the higher-order CRF, which outperforms first-order CRF on NER task (Sarawagi and Cohen, 2005) and morphological tagging task (Mueller et al., 2013).

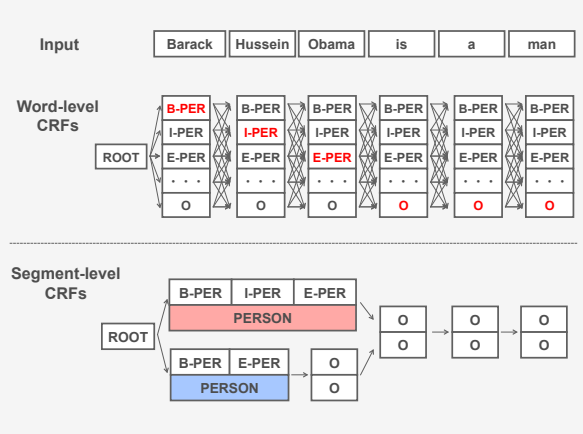

Figure 1: The difference between word-level CRF and segment-level CRF. The segment-level CRF can consider higher-order label dependencies.

In this paper, we extend a neural CRF from word-level to segment-level and propose *Segment-level Neural CRF*. Our method has two main advantages: (1) *segment-level* linear chain CRF can consider higher-order word label dependencies (e.g., the relations between named entities and the other words); (2) it is easy to incorporate dictionary features into the model directly since a dictionary entry and a segment (e.g., a named entity) are in one-to-one correspondence.

Our experiments on chunking and NER demonstrate that our method outperforms conventional word-level neural CRF.

Proceedings of the The 8th International Joint Conference on Natural Language Processing, pages 97–102,
Taipei, Taiwan, November 27 – December 1, 2017

2 Word-level Neural CRF

As a baseline method, we use word-level neural CRF proposed by (Ma and Hovy, 2016) since their method achieves state-of-the-art performance on NER. Specifically, they propose Bi-directional LSTM-CNN CRF (BLSTM-CNN-CRF) for sequential tagging. Here, we briefly review their BLSTM-CNN-CRF model.

Let w_t be the t-th word in an input sentence and $C_t = c_t^{(1)}, \ldots, c_t^{(k)}$ be the character sequence of w_t. BLSTM-CNN-CRF uses both word-level embedding $\mathbf{w}_t \in \mathbb{R}^{d_{word}}$ and character-level embedding $\mathbf{c}_t \in \mathbb{R}^{d_{char}}$. Given a word sequence $X = w_1, \ldots, w_n$, the model outputs a score vector $\mathbf{o}_t$ as follows.

$$\begin{aligned}
\mathbf{c}_t &= \text{CNN}_{char}(C_t), \\
\mathbf{x}_t &= \mathbf{w}_t \oplus \mathbf{c}_t, \\
\mathbf{h}_t &= \text{Bi-LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{h}_{t+1}) \qquad (1) \\
&= \text{LSTM}_f(\mathbf{x}_t, \mathbf{h}_{t-1}) \oplus \text{LSTM}_b(\mathbf{x}_t, \mathbf{h}_{t+1}), \\
\mathbf{o}_t &= \text{softmax}(\mathbf{W}_{TG}\mathbf{h}_t + \mathbf{b}_{TG}),
\end{aligned}$$

where CNN_{char} is the character-level CNN function, $\oplus$ is the concatenation of two vectors, LSTM_f is the forward LSTM function, LSTM_b is the backward LSTM function, Bi-LSTM is the Bi-LSTM function, respectively. Then, $\mathbf{W}_{TG} \in \mathbb{R}^{|\mathcal{T}| \times d_{hidden}}$ is the weight matrix to learn, $\mathbf{b}_{TG} \in \mathbb{R}^{|\mathcal{T}|}$ is the bias vector to learn, $|\mathcal{T}|$ is the size of tag set $\mathcal{T}$, d_{hidden} is the size of hidden layer of Bi-LSTM, and $\mathbf{o}_t \in \mathbb{R}^{|\mathcal{T}|}$ is the score vector in which each element is the probability of a possible tag.

In BLSTM-CNN-CRF, CRF is applied to the output layer. The conditional probability of CRF is defined as follows:

$$\begin{aligned}
\phi(y_{i-1}, y_i, o_i^{(y_i)}) &= \exp(o_i^{(y_i)} + A_{y_{i-1}, y_i}), \\
p(\mathbf{y}|\mathbf{o}; \mathbf{A}) &= \frac{\prod_{i=1}^{n} \phi(y_{i-1}, y_i, o_i^{(y_i)})}{\sum_{y' \in \mathcal{Y}} \prod_{i=1}^{n} \phi(y'_{i-1}, y'_i, o_i^{(y'_i)})},
\end{aligned}$$

where $\phi(y_{i-1}, y_i, o_i^{(y_i)})$ is the potential function[1], $y_i \in \{0, \ldots, |\mathcal{T}|-1\}$ is the index of tag, $o_i^{(j)}$ is the j-th element of the vector $\mathbf{o}_i$. Then, $\mathbf{A} \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{T}|}$ is a transition score matrix, A_{y_{i-1}, y_i} is a transition score for jumping from tag y_{i-1} to y_i, and $\mathcal{Y}$ indicates all possible paths.

At test time, the predicted sequence is obtained by finding the highest score in a all possible paths using Viterbi algorithm as follows:

$$\tilde{y} = \underset{\mathbf{y} \in \mathcal{Y}}{\text{argmax}}\, p(\mathbf{y}|\mathbf{o}_i; \mathbf{A}).$$

3 Segment-level Neural CRF

In this section, we describe our proposed method. Our segment-level neural CRF consists of the following two steps:

(i) A *segment lattice* is constructed from a sequence of words by pruning unlikely BIO tags to reduce a search space. This is because it is difficult to consider all possible variable length segments in practice.

(ii) We use a linear chain CRF to find the highest score path on the segment lattice.

3.1 Constructing Segment Lattice

A *segment lattice* is a graph structure where each path corresponds to a candidate segmentation path as shown in the lower part of Figure 1. The segment lattice is a kind of semi-Markov model (Sarawagi and Cohen, 2005). To construct the segment lattice, we firstly give an input sentence to the word-level tagging model, then obtain the score vector $\mathbf{o}_t$ for each word that gives the probabilities of possible BIO tags. Then, we generate the candidate BIO tags whose scores are greater than the threshold T. After that, we construct the segment lattice by generating admissible segments from the candidate BIO tags. For example, we generate the *PERSON* segment from the candidate BIO tags {*B-PER*, *I-PER*, *E-PER*}.

The threshold T is a hyper-parameter for our model. We describe how to choose the threshold T in Section 4.3. While it has been shown that the CRF layer is required to achieve the state-of-the-art performance in Ma and Hovy (2016), we observe that the CRF has no significant effect on the final performance for the lattice construction. Therefore, we use BLSTM-CNN (without CRF) as the word-level tagging model in this paper.

3.2 Segment-level Vector Representation

To find the highest score path in the segment lattice, we use a standard linear chain CRF at segment-level. Since each segment has variable length, we need to obtain fixed-dimensional

[1]While (Ma and Hovy, 2016) define $\phi(y_{i-1}, y_i, o_i) = \exp(W_{y_{i-1}, y_i} o_i + A_{y_{i-1}, y_i})$ as the potential function where W is the weight vector corresponding to label pair (y_{i-1}, y_i), we use the simple potential function here.

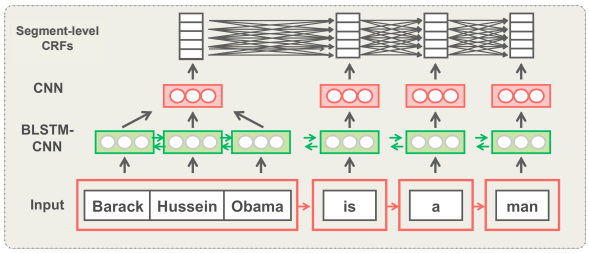

Figure 2: Details of the Segment-level Neural CRF model.

segment-level vector representation for neural networks.

Figure 2 shows the details of the segment-level neural CRF model. Let $u_i = w_b, w_{b+1}, \ldots, w_e$ be the i-th segment in a segment lattice, b is the starting word index, and e is the ending word index. To obtain the fixed-dimensional vector $\mathbf{u}_i \in \mathbb{R}^{d_{node}}$ for the segment u_i, we apply a CNN to the hidden vector sequence $\mathbf{h}_{b:e} = \mathbf{h}_b, \mathbf{h}_{b+1}, \ldots, \mathbf{h}_e$ by Eq. (1), and compute the score vector $\mathbf{z}_i$ as follows:

$$\begin{aligned} \mathbf{r}_i &= \text{CNN}_{node}(\mathbf{h}_{b:e}), \\ \mathbf{z}_i &= \text{softmax}(\mathbf{W}_{LS}\mathbf{r}_i + \mathbf{b}_{LS}), \end{aligned}$$

where CNN_{node} is the CNN function for the segment vector, $\mathbf{W}_{LS} \in \mathbb{R}^{|\mathcal{N}| \times d_{node}}$ is the weight matrix to learn, $\mathbf{b}_{LS} \in \mathbb{R}^{d_{node}}$ is the bias vector to learn, $|\mathcal{N}|$ is the size of named entity type set $\mathcal{N}$, d_{node} is the size of the segment vector, and $\mathbf{z}_i \in \mathbb{R}^{|\mathcal{N}|}$ is the score vector in which each element is the probability of a possible NE type.

Finally, we apply a linear chain CRF to find the highest score path in the segment lattice as we describe in Section 2.

3.3 Dictionary Features for NER

In this subsection, we describe the use of two additional dictionary features for NER. Since an entry of named entity dictionary and the segment in our model are in one-to-one correspondence, it is easy to directly incorporate the dictionary features into our model. We use following two dictionary features on NER task.

Binary feature The binary feature $\mathbf{e}_i \in \mathbb{R}^{d_{dict}}$ indicates whether the i-th segment (e.g., a named entity) exists in the dictionary or not. We use the embedding matrix $\mathbf{W}_{dict} \in \mathbb{R}^{2 \times d_{dict}}$, where d_{dict} is the size of the feature embedding. $e \in \{0, 1\}$ is the binary index which indicates whether the segment exists in the dictionary or not. Using the index e, we extract the column vector $\mathbf{e}_i \in \mathbb{R}^{d_{dict}}$ from $\mathbf{W}_{dict}$ and concatenate the segment vector $\mathbf{r}_i$ and $\mathbf{e}_i$. The concatenated segment vector $\mathbf{r}'_i$ is defined as $\mathbf{r}'_i = \mathbf{r}_i \oplus \mathbf{e}_i$. $\mathbf{W}_{dict}$ is a randomly initialized matrix and updated in the training time. To incorporate the popularity of the Wikipedia entity into our method, we also concatenate one-dimensional vector constructed from the page view count for one month period into $\mathbf{e}_i$. The page view count is normalized by the number of candidate segments in the segment lattice. The Wikipedia dictionary is constructed by extracting the titles of all Wikipedia pages and the titles of all redirect pages from the Wikipedia Dump Data[2].

Wikipedia embedding feature Another additional feature is the Wikipedia embeddings proposed by Yamada et al. (2016). Their method maps words and entities (i.e., Wikipedia entities) into the same continuous vector space using the skip-gram model (Mikolov et al., 2013). We use only the 300 dimensional entity embeddings in this paper. Please refer to Yamada et al. (2016) for more detail.

4 Experiments

4.1 Datasets

We evaluate our method on two segment-level sequence tagging tasks: NER and text chunking[3].

For NER, we use CoNLL 2003 English NER shared task (Tjong Kim Sang and De Meulder, 2003). Following previous work (Ma and Hovy, 2016), we use BIOES tagging scheme in the word-level tagging model.

For text chunking, we use the CoNLL 2000 English text chunking shared task (Tjong Kim Sang and Buchholz, 2000). Following previous work (Søgaard and Goldberg, 2016), the section 19 of WSJ corpus is used as the development set. We use BIOES tagging scheme in the word-level tagging model and measure performance using F1 score in all experiments.

4.2 Model Settings

To generate a segment lattice, we train word-level BLSTM-CNN with the same hyper-parameters used in Ma and Hovy (2016): one layer 200 dimentional Bi-directional LSTMs for each direction, 30 filters with window size 3 in character-

[2]The dump data of Wikipedia is available in Wikimedia http://dumps.wikimedia.org/. We use the dump data at 2016-09-20.

[3]Our code will be available from http://xxxx

Threshold	Oracle Train	Dev	Test
T=0.05	99.93	99.71	99.27
T=0.0005	99.99	99.96	99.71
T=0.00005	100.0	99.98	99.83

Table 1: Threshold T and Oracle score on NER.

	Test Prec.	Recall	F1
BLSTM-CNN	89.04	90.40	89.72
BLSTM-CNN-CRF[3]	90.82	91.11	90.96
Our method	91.07	91.50	91.28
+ Binary Dict	91.05	91.69	91.37
+ WikiEmb Dict	91.29	91.58	91.44
+ Binary + WikiEmb	91.47	91.62	**91.55**
Ma and Hovy (2016)	91.35	91.06	**91.21**

Table 2: Result of CoNLL 2003 English NER.

level CNN, and 100 dimentional pre-trained word embedding of GloVe (Pennington et al., 2014). At input layer and output layer, we apply dropout (Srivastava et al., 2014) with rate at 0.5. In our model, we set 400 filters with window size 3 in CNN for segment vector. To optimize our model, we use AdaDelta (Zeiler, 2012) with batch size 10 and gradient clipping 5. We use early stopping (Caruana et al., 2001) based on performance on development sets.

4.3 How to choose threshold

The threshold T is a hyper-parameter for our model. We choose the threshold T based on how a segment lattice maintains the gold segments in the training and development sets. The threshold T and the oracle score are shown in Table 1. In our experiment, the T = 0.00005 is used in NER task and T = 0.0005 is used in chunking task.

4.4 Results and Discussions

The results of CoNLL 2003 NER is shown in Table 2. By adding a CRF layer to BLSTM-CNN, it improves the F1 score from 89.72 to 90.96. This result is consistent with the result of (Ma and Hovy, 2016). By using segment-level CRF, it further improves the F1 score from 90.96 to 91.28. Furthermore, by using the binary dictionary feature, it improves the F1 score from 91.28 to 91.37 and by using the Wikipedia embedding feature, it improves the F1 score from 91.28 to 91.44. Eventually, we achieve the F1 score 91.55 with two dictionary features.

[3]This is same method in (Ma and Hovy, 2016) and this F-1 score is the result of our implementation.

	Test Prec.	Recall	F1
BLSTM-CNN	90.85	91.92	91.38
BLSTM-CNN-CRF	94.67	94.43	94.55
Our method	94.55	95.12	**94.84**

Table 3: Result of CoNLL 2000 Chunking.

The results of CoNLL 2000 Chunking is shown in Table 3. Similar to NER task, by adding a CRF layer to BLSTM-CNN, it improves the F1 score from 91.38 to 94.55. Furthermore, by using segment-level CRF, it improves the F1 score from 94.55 to 94.84.

In both experiments, it improves the F1 score by using segment-level CRF. On the NER experiment, the additional dictionary features help to obtain further improvement.

5 Related Work

Several different neural network methods have been proven to be effective for NER (Collobert et al., 2011; Chiu and Nichols, 2016; Lample et al., 2016; Ma and Hovy, 2016). Ma and Hovy (2016) demonstrate that combining LSTM, CNN and CRF achieves the state-of-the-art performance on NER and chunking tasks.

Mueller et al. (2013) show that higher-order CRF outperforms first-order CRF. Our work differs from their work in that it can handle segments of variable lengths and thus it is easy to incorporate dictionary features directly.

Zhuo et al. (2016) propose Gated Recursive Semi-CRF, which models a sequence of segments and automatically learns features. They combine Semi-CRF (Sarawagi and Cohen, 2005) and neural networks. However they report the F1 score 89.44% on NER and 94.73[4] on Chunking which are lower than the scores of our method.

Kong et al. (2016) propose segmental recurrent neural networks (SRNNs). SRNNs are based on Bi-LSTM feature extractor and uses dynamic programming algorithm to reduce search space.

6 Conclusion

In this paper, we propose the segment-level sequential modeling method based on a segment lat-

[4]This is under the setting without external resource. They add Brown clusters features and report the F1 score 95.01.

tice structure. Our experimental results show that our method outperforms conventional word-level neural CRF. Furthermore, two additional dictionary features help to obtain further improvement on NER task.

Acknowledgments

Part of this work was supported by JST CREST Grant Number JPMJCR1513, Japan.

References

Rich Caruana, Steve Lawrence, and C. Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, MIT Press, pages 402–408.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics* 4:357–370.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* 12:2493–2537.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural Computation* 12(10):2451–2471.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Lingpeng Kong, Chris Dyer, and Noah A Smith. 2016. Segmental recurrent neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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 the Eighteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 260–270.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1(4):541–551.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1064–1074.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 322–332.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543.

Sunita Sarawagi and William W Cohen. 2005. Semimarkov conditional random fields for information extraction. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, MIT Press, pages 1185–1192.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, volume 2, pages 231–235.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Erik F Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*. Association for Computational Linguistics, pages 127–132.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, pages 142–147.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Berlin, Germany, pages 250–259.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Jingwei Zhuo, Yong Cao, Jun Zhu, Bo Zhang, and Zaiqing Nie. 2016. Segment-level sequence modeling using gated recursive semi-markov conditional random fields. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1413–1423.