Character-Based LSTM-CRF with Radical-Level Features for Chinese Named Entity Recognition

Chuanhai Dong[1], Jiajun Zhang[1], Chengqing Zong[1](✉), Masanori Hattori[2], and Hui Di[2]

[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
{chuanhai.dong,jjzhang,cqzong}@nlpr.ia.ac.cn
[2] Toshiba (China) R&D Center, Beijing, China
masanori.hattori@toshiba.co.jp, dihui@toshiba.com.cn

Abstract. State-of-the-art systems of Chinese Named Entity Recognition (CNER) require large amounts of hand-crafted features and domain-specific knowledge to achieve high performance. In this paper, we apply a bidirectional LSTM-CRF neural network that utilizes both character-level and radical-level representations. We are the first to use character-based BLSTM-CRF neural architecture for CNER. By contrasting the results of different variants of LSTM blocks, we find the most suitable LSTM block for CNER. We are also the first to investigate Chinese radical-level representations in BLSTM-CRF architecture and get better performance without carefully designed features. We evaluate our system on the third SIGHAN Bakeoff MSRA data set for simplfied CNER task and achieve state-of-the-art performance 90.95% F1.

Keywords: BLSTM-CRF · Radical features · Named Entity Recognition

1 Introduction

Named Entity Recognition (NER) is a fundamental technique for many natural language processing applications, such as information extraction, question answering and so on. Carefully hand-crafted features and domain-specific knowledge resources, such as gazetteers, are widely used to solve the problem. As to Chinese Named Entity Recognition (CNER), there are more complicated properties in Chinese, for example, the lack of word boundary, the complex composition forms, the uncertain length, NE nesting definition and so on [7].

Many related research regards NER as a sequence labelling task. The applied methods on CNER include Maximum Entropy (ME) [3,20], Hidden Markov Model (HMM) [8], Support Vector Machine (SVM) [19] and Conditional Random Field (CRF) algorithms [7,10]. Character-based tagging strategy achieves comparable performance without results of Chinese Word Segmentation (CWS) [2,31], which means Chinese character can be the minimum unit to identify NEs

DOI: 10.1007/978-3-319-50496-4_20

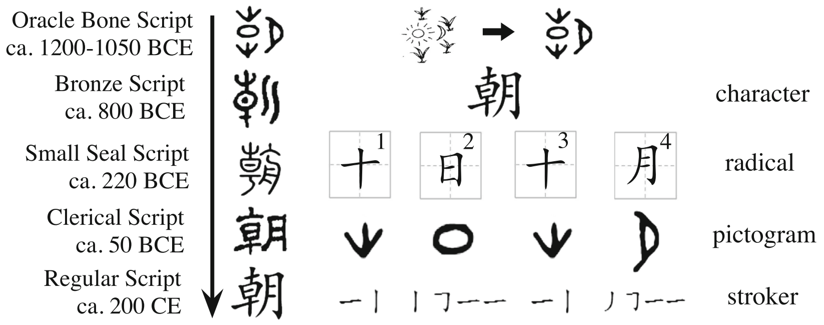

Fig. 1. Decomposition of Chinese character

instead of words. Character-based tagging simplifies the task without reducing performance, so we apply character-based tagging strategy in this paper. With the rapid development of deep learning, neural networks start to show its great capability in NLP tasks and outperform popular statistical algorithms like CRF [16]. Recurrent Neural Network (RNN) learns long distance dependencies better than CRF which utilizes features found in a certain context window. As a special kind of RNN, Long Short-term Memory (LSTM) neural network [13] is proved to be efficient in modeling sequential text [14]. LSTM is designed to cope with the gradient varnishing/exploding problems [1]. Char-LSTM [17] is introduced to learn character-level sequences, such as prefix and suffix in English. As to Chinese, each character is semantically meanful, thanks to its pictographic root from ancient Chinese as depicted in Fig. 1 [26]. The left part of Fig. 1 illustrates the evolution process of Chinese character "朝". The right part of Fig. 1 demonstrates the decomposition. This character "朝", which means "morning", is decomposed into 4 radicals[1] that consists of 12 strokes. As depicted by the pictograms in the right part of Fig. 1, the 1st radical (and the 3rd that happens to be the same) means "grass", and the 2nd and the 4th mean the "sun" and the "moon", respectively. These four radicals altogether convey the meaning that "the moment when sun arises from the grass while the moon wanes away", which is exactly "morning". On the other hand, it is hard to decipher the semantics of strokes, and radicals are the minimum semantic unit for Chinese.

In this paper, we use a character-based bidirectional LSTM-CRF (BLSTM-CRF) neural network for CNER task. By contrasting results of LSTM varients, we find a suitable LSTM block for CNER. Inspired by char-LSTM [17], we propose a radical-level LSTM for Chinese to capture its pictographic root features and get better performance on CNER task.

2 Related Work

In the third SIGHAN Bakeoff [18] CNER shared task, there are three kinds of NEs, namely locations, persons, organizations. Although other statistical models,

[1] https://en.wikipedia.org/wiki/Radical_(Chinese_characters).

such as HMM and ME, once achieved good results [3,8,20], nearly all leading performance are achieved using CRF model on this bakeoff. Many following work emphasizes on feature-engineering of character-based CRF model [7,10].

Several neural architectures have previously been proposed for English NER. Our model basically follows the idea of [17]. [17] presented a LSTM-CRF architecture with a char-LSTM layer learning spelling features from supervised corpus and didn't use any additional resources or gazetteers except a massive unlabelled corpus for unsupervised learning of pretrained word embeddings. Instead of char-LSTM for phonogram languages in [17], we propose a radical-level LSTM designed for Chinese characters. [6] uses a Convolutional Neural Network (CNN) over a sequence of word embeddings with a CRF layer on top. [14] presented a model similar to [17]'s LSTM-CRF, but used hand-crafted spelling features. [4] proposed a hybrid of BLSTM and CNNs to model both character-level and word-level representations in English. They utilized external knowledge such as lexicon features and character-type. [22] proposed a BLSTM-CNNs-CRF architecture using CNNs to model character-level information. [28] proposed a hierarchical GRU neural network for sequence tagging using multi-task and cross-lingual joint training.

Only a few work focused on Chinese radical information. [27] proposed a feed-forward neural network similar to [6], but used Chinese radical information as supervised tag to train character embeddings. [21] trained their character embeddings in a holistic unsupervised and bottom-up way based on [23,24], using both radical and radical-like components. [26] used radical embeddings as input like ours and utilized word2vec [23] package to pretrain radical vectors, but they used CNNs, while we use LSTM to obtain radical-level information.

3 Neural Network Architecture

3.1 LSTM

RNNs are a family of neural networks designed for sequential data. RNNs take as input a sequence of vectors $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ and return another sequence $(\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$ that represents state layer information about the sequence at each step in the input. In theory, RNNs can learn long dependencies but in practice they tend to be biased towards their most recent inputs in the sequence [1]. Long Short-term Memory Networks (LSTMs) incorporate a memory-cell to combat this issue and have shown great capabilities to capture long-range dependencies. Our LSTM has input gate, output gate, forget gate and peephole connection. The update of cell state use both input gate and forget gate results. The implementation is:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \qquad (\textit{input gate})$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \qquad (\textit{forget gate})$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \qquad (\textit{cell state})$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \qquad (\textit{output gate})$$
$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \qquad (\textit{output})$$

where σ is the element-wise sigmoid function, $\odot$ is the element-wise product, $\mathbf{W}$'s are weight matrices, and $\mathbf{b}$'s are biases.

We get the context vector of a character using a bidirectional LSTM. For a given sentence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ containing n characters, each character represented as a d-dimensional vector, a LSTM computes a representation $\overrightarrow{\mathbf{h}_t}$ of the left context of the sentence at every character t. Similarly, the right context $\overleftarrow{\mathbf{h}_t}$ starting from the end of the sentence should provide useful information. By reading the same sentence in reverse, we can get another LSTM which achieves the right context information. We refer to the former as the forward LSTM and the latter as the backward LSTM. The context vector of a character is obtained by concatenating its left and right context representations, $\mathbf{h}_t = \left[\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}\right]$.

3.2 CRF

The hidden context vector $\mathbf{h}_t$ can be used directly as features to make independent tagging decisions for each output y_t. But in CNER, there are strong dependencies across output labels. For example, I-PER cannot follow B-ORG, which constraints the possible output tags after B-ORG. Thus, we use CRF to model the outputs of the whole sentence jointly. For an input sentence,

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

we regard $\mathbf{P}$ as the matrix of scores outputted by BLSTM network. $\mathbf{P}$ is of size $n \times k$, where k is the number of distinct tags, and $P_{i,j}$ is the score of the j^{th} tag of the i^{th} character in a sentence. For a sequence of predictions,

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

we define its score as

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

where $\mathbf{A}$ is a matrix of transition scores which models the transition from tag i to tag j. We add *start* and *end* tag to the set of possible tags and they are the tags of y_0 and y_n that separately means the start and the end symbol of a sentence. Therefor, $\mathbf{A}$ is a square matrix of size $k+2$. After applying a softmax layer over all possible tag sequences, the probability of the sequence $\mathbf{y}$:

$$p(\mathbf{y}|\mathbf{X}) = \frac{e^{s(\mathbf{X},\mathbf{y})}}{\sum_{\tilde{\mathbf{y}} \in \mathbf{Y_X}} e^{s(\mathbf{X},\tilde{\mathbf{y}})}} \tag{2}$$

We maximize the log-probability of the correct tag sequence during training:

$$\log(p(\mathbf{y}|\mathbf{X})) = s(\mathbf{X}, \mathbf{y}) - \log(\sum_{\tilde{\mathbf{y}} \in \mathbf{Y_X}} e^{s(\mathbf{X},\tilde{\mathbf{y}})}) \tag{3}$$

$$= s(\mathbf{X}, \mathbf{y}) - \underset{\tilde{\mathbf{y}} \in \mathbf{Y_X}}{\text{logadd}}\; s(\mathbf{X}, \tilde{\mathbf{y}}) \tag{4}$$

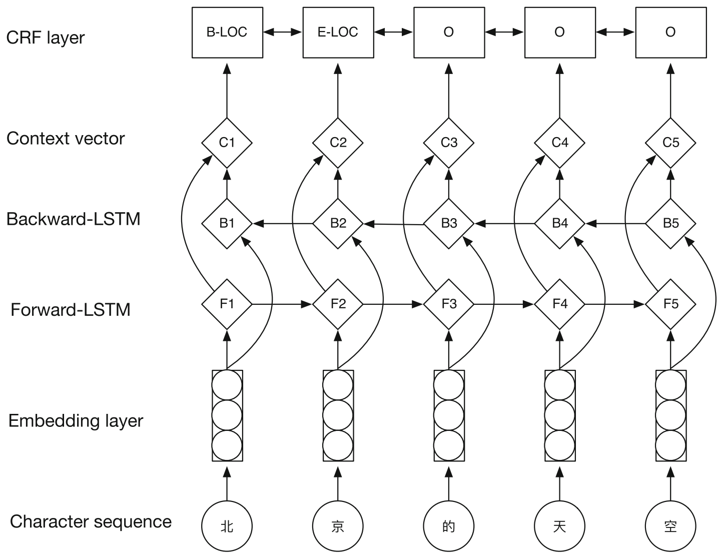

Fig. 2. Main architecture of character-based BLSTM-CRF.

where $\mathbf{Y_X}$ represents all possible tag sequences including those that do not obey the IOB format constraints. It's evident that invalid output label sequences will be discouraged. While decoding, we predict the output sequence that gets the maximum score given by:

$$\mathbf{y}^* = \underset{\tilde{\mathbf{y}} \in \mathbf{Y_X}}{\arg\max}\ s(\mathbf{X}, \tilde{\mathbf{y}}) \tag{5}$$

We just consider bigram constraints between outputs and use dynamic programming during decoding (Fig. 2).

3.3 Radical-Level LSTM

Chinese characters are often composed of smaller and primitive radicals, which serve as the most basic unit for building character meanings [21]. These radicals are inherent features inside Chinese characters and bring additional information that has semantic meaning. For example, the characters "你"(you), "他"(he), and "们"(people) all have the meanings related to human because of their shared radical "亻"(human), a variant of Chinese character "人"(human) [21]. Intrinsically, this kind of radical semantic information is useful to make characters with similar radical sequences close to each other in vector space. It motivates us to focus on the radicals of Chinese characters.

In modern Chinese, character usually contains several radicals. In MSRA data set, including training set and test set, 75.6% characters have more than one radical. We get radical compositions of Chinese characters from *online Xinhua Dictionary*[2]. In simplified Chinese, radicals inside a character may have changed from its original shape. For example, the first radical of the Chinese character “腿”(leg) is “月”(moon), which is the simplified form of traditional radical “肉”(meat), while the radical of “朝”(morning) is also “月”(moon) and actually means moon. To deal with these variants, we replace the most important simplified radical, which is also called *bù*(meaning “categories”), with its traditional shape of radical to restore its original meaning. Both the simplified radical and the traditional radical of a character can be found in *online Xinhua Dictionary*, too. For a monoradical character, we just use itself as its radical part. After this substitution, we get all the composing radicals to build a radical list of every Chinese character. As each radical of a character has a unique position, we regard the radicals of one character as a sequence in writing order. We employ a radical-level bidirectional LSTM to capture the radical information. Figure 3 shows how we obtain the final input embeddings of a character.

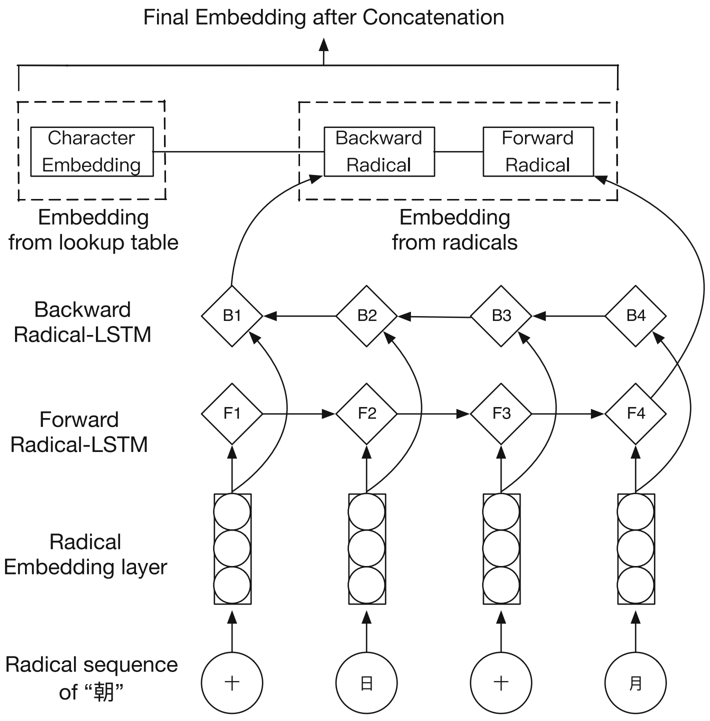

Fig. 3. The final embeddings of Chinese character “朝”. We concatenate the final outputs of the radical-level BLSTM to the character embedding from a lookup table as the final representation for the character “朝”.

[2] http://tool.httpcn.com/Zi/.

3.4 Tagging Scheme

As we use a character-based tagging strategy, we need to assign a named entity label to every character in a sentence. Many NEs span multiple characters in a sentence. Sentences are usually represented the IOB format(Inside, Outside, Beginning). In this paper, we use IOBES tagging scheme. Using this scheme, more information about the following tag is considered.

4 Network Training

4.1 LSTM Variants

We compare results of LSTM variants on CNER to find a better variant of LSTM. The initial version of LSTM block [13] included cells, input and output gates to solve the gradient varnishing/exploding problem. So we keep input and output gate in most of the variants. The derived variants of LSTM mentioned in Sect. 3.1 are the following:

1. No Peepholes, No Forget Gate, Coupled only Input Gate (NP, NFG, CIG)
2. Peepholes, No Forget Gate, Coupled only Input Gate (P, NFG, CIG)
3. No Peepholes, Forget Gate, Coupled only Forput Gate (NP, FG, CFG)
4. No Peepholes, Forget Gate, Coupled Input and Forget Gate (NP, FG, CIFG)
5. No Peepholes, Forget Gate(1), Coupled Input and Forget Gate (NP, FG(1), CIFG)
6. Gated Recurrent Unit (GRU)

Considering formule *cell state* in Sect. 3.1, if we use CIG to update cell state, there will be only one gate for both the input and the cell state, so forget gate will be omitted. This is equivalent to setting $\mathbf{f}_t = \mathbf{1} - \mathbf{i}_t$ instead of using the forget gate independently. GRU [5] is a variant of LSTM without having separate memory cells and exposes the whole state each time. (5) means bias of forget gate are initialized to 1 instead 0. Results of different variants are reported in Sect. 5.2.

4.2 Pretrained Embeddings

There are usually too many parameters to learn from only a limited training data in deep learning. To solve this problem, unsupervised learning method to pre-train embeddings emerged, which only used large unlabelled corpus. Instead of randomly initialized embeddings, well pretrained embeddings have been proved important for performance of neural network architectures [11,17]. We observe significant improvements using pretrained character embeddings over randomly initialized embeddings. Here we use gensim[3] [25], which contains a python version implementation of word2vec. These embeddings are fine-tuned during training. We use Chinese Wikipedia backup dump of 20151201. After transforming traditional Chinese to simplified Chinese, removing non-utf8 chars and unifying

[3] https://radimrehurek.com/gensim/index.html.

different styles of punctuations, we get 1.02 GB unlabelled corpus. Character embeddings are pretrained using CBOW model because it's faster than skip-gram model. Results using different character embedding dimension are shown in Sect. 5.2. Radical embeddings are randomly initialized with dimension of 50.

4.3 Training

We use dropout training [12] before the input to LSTM layer with a probability of 0.5 in order to avoid overfitting and observe a significant improvement in CNER performance. According to [17], we train our network using the back-propagation algorithm updating our parameters on every training example, one at a time. We use stochastic gradient decent (SGD) algorithm with a learning rate of 0.05 for 50 epochs on training set. Dimension of LSTM is the same as its input dimension.

5 Experiments

5.1 Data Sets

We test our model on MSRA data set of the third SIGHAN Bakeoff Chinese named entity recognition task. This dataset contains three types of named entities: locations, persons, organizations. Chinese word segmentation is not available in test set. We just replace every digit with a zero and unify the styles of punctuations appeared in MSRA and pretrained embeddings.

5.2 Results

Table 1 presents our comparisons with different variants of LSTM block. (1) achieves best performance among all the variants. We observe that peephole connections do not improve performance in CNER, but they increase training time because of more connections. Different from conclusions of [9,15], performance decreases after adding the forget gate no matter how to update cell state.

Table 1. Results with LSTM variants.

ID	Variants of LSTM	F1
(1)	NP, NFG, CIG	90.75
(2)	P, NFG, CIG	90.37
(3)	NP, FG, CFG	89.85
(4)	NP, FG, CIFG	89.90
(5)	NP, FG(1), CIFG	90.45
(6)	GRU	90.43

Table 2. Results with different character embedding dimensions.

Dimension	F1
50	88.92
100	90.75
200	90.44

But performance is prompted after setting bias of the forget gate to 1, which make the forget gate to tend to remember long distance dependencies. The way to couple the input and forget gates does not have significantly impact on performance, which is the same as [9]. GRU needs less training time due to the simplification of inner structure without hurting performance badly. Finally, we choose (1) as the LSTM block in the following experiments.

Table 2 shows results with different character embedding dimensions. We use pretrained character embeddings, dropout training, BLSTM-CRF architecture on all three experiments. Different from results reported by [17] in English, 50 dims is not enough to represent Chinese character. 100 dims achieve 1.83% better than 50 dims in CNER, but no more improvement is observed using 200 dims. We use 100 dims in the following experiments.

Our architecture have several components that have different impact on the overall performance. Without CRF layer on top, the model does not converge to a stable state in even 100 epochs using the same learning rate 0.05. We explore the impact that dropout, radical-level representations, pretraining of character emebeddings have on our LSTM-CRF architecture. Results with different architectures are given in Table 3. We find that radical-level LSTM gives us an improvement of +0.53 in F_1 with random initialized character embeddings. It is evident that radical-level information is effective for Chinese. Pretrained character embeddings, which is trained using unlabelled Chinese Wikipedia corpus by unsupervised learning, increase result by +1.84 based on dropout training. Dropout is important and gives the biggest improvement of +3.88. Radical-level LSTM makes out-of-vocabulary characters, which are initialized with random embeddings, close to known characters that have similar radical components. Only 3 characters in the training and test set can not be found in Wikipedia corpus. In other words, there are few characters initialized with random embeddings. So we do not find further improvement using both radical-level LSTM and well pretrained character embeddings. Radical-level LSTM is obviously effective when there is no large corpus for character pretrainings.

Table 3. Results with different components.

Variant	F1
random + dropout	88.91
random + radical + dropout	89.44
pretrain + dropout	90.75
pretrain	86.87

Table 4 shows our results compared with other models for Chinese named entity recognition. To show the capability of our model, we train our model for 100 epochs instead of 50 epochs in the previous experiments. Zhou [31] got first place using word-based CRF model with delicated hand-crafted features in the

Table 4. Results with different methods. We train our models for 100 epochs in this experiment. [a]Indicates results in the open track.

Model	PER-F	LOC-F	ORG-F	P	R	F
Zhou [31]	90.09	85.45	83.10	88.94	84.20	86.51
Chen [2]	82.57	90.53	81.96	91.22	81.71	86.20
Zhou [32]	90.69	91.90	86.19	91.86	88.75	90.28
Zhang [29][a]	96.04	90.34	85.90	92.20	90.18	**91.18**
BLSTM-CRF + radical	89.62	91.76	85.79	91.39	88.22	89.78
BLSTM-CRF + pretrain	91.77	**92.10**	**87.30**	91.28	**90.62**	**90.95**

closed track on MSRA data set with F1 86.51%. Chen [2] achieved F1 86.20% using character-base CRF model. Zhou [32] used a global linear model to identify and categorize CNER jointly with 10 carefully designed feature templates for CNER and 31 context feature templates from [30]. Zhou [32] adopted a more granular labelling schemes for example changing PER tags that are shorter than 4 characters and begin with a Chinese surname to Chinese-PER. In the open track, Zhang [29] got first place using ME model combining knowledge from various sources with 91.18%, such as person name list, organization name dictionary, location keyword list and so on. Our BLSTM-CRF with radical embeddings outperforms previous best CRF model by +3.27 in overall. Our BLSTM-CRF with pretrained character embeddings outperforms all the previous models except for results of Zhang [29] in the open track and achieves state-of-the-art performance with F1 90.95%. Especially for ORG entities, which is the most difficult category to recognize, our approach utilizes the capability of LSTM to learn long-distance dependencies and achieves a remarkable performance. The main reason that Zhang [29] obtained better performance than ours by +0.23 in overall F_1 is that they used additional name dictionaries to achieve very high PER-F while we do not use those dictionaries. Our neural network architecture does not need any hand-crafted features which are important in Zhou [32].

6 Conclusion

This paper presents our neural network model, which incorporates Chinese radical-level information to character-based BLSTM-CRF and achieves state-of-the-art results. We utilize LSTM block to learn long distance dependencies which are useful to recognize ORG entities. Different from research focused on feature engineering, our model does not use any hand-crafted features or domain-specific knowledge and thus, it can be transferred to other domains easily. In the future, we would like to transfer our model to Chinese social media domain.

Acknowledgements. This research work has been partially funded by the Natural Science Foundation of China under Grant No. 91520204 and No. 61303181.

References

1. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**, 157–166 (1994)
2. Chen, A., Peng, F., Shan, R., Sun, G.: Chinese named entity recognition with conditional probabilistic models. In: Proceedings of 5th SIGHAN Workshop on Chinese Language Processing, pp. 173–176 (2006)
3. Chieu, H.L., Ng, H.T.: Named entity recognition: a maximum entropy approach using global information. In: Proceedings of 19th International Conference on Computational Linguistics, vol. 1, pp. 1–7. Association for Computational Linguistics, Morristown (2002)
4. Chiu, J.P.C., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. In: Transactions of the ACL, pp. 1–9 (2015)
5. Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: encoder-decoder approaches. In: SSST-8, pp. 103–111 (2014)
6. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011)
7. Duan, H., Zheng, Y.: A study on features of the CRFs-based Chinese. Int. J. Adv. Intell. **3**, 287–294 (2011)
8. Fu, G., Luke, K.K.: Chinese named entity recognition using lexicalized HMMs. ACM SIGKDD Explor. Newsl. **7**, 19–25 (2005)
9. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: a search space odyssey. p. 10 (2015). arXiv
10. Han, A.L.-F., Wong, D.F., Chao, L.S.: Chinese named entity recognition with conditional random fields in the light of Chinese characteristics. In: Kłopotek, M.A., Koronacki, J., Marciniak, M., Mykowiecka, A., Wierzchoń, S.T. (eds.) IIS 2013. LNCS, vol. 7912, pp. 57–68. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38634-3_8
11. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**, 504–507 (2006)
12. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. pp. 1–18 (2012). arXiv e-prints
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
14. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging (2015). arXiv
15. Jozefowicz, R., Zaremba, W., Sutskever, I.: An empirical exploration of recurrent network architectures. In: Proceedings of 32nd International Conference on Machine Learning, pp. 2342–2350 (2015)
16. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of 18th International Conference on Machine Learning, ICML 2001, pp. 282–289 (2001)
17. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. pp. 1–10 (2016). arXiv
18. Levow, G.A.: The third international Chinese language processing bakeoff: word segmentation and named entity recognition. In: Computational Linguistics, pp. 108–117 (2006)

19. Li, L., Mao, T., Huang, D., Yang, Y.: Hybrid models for Chinese named entity recognition. In: Proceedings of 5th SIGHAN Workshop on Chinese Language Processing, pp. 72–78 (2006)
20. Li, W., Li, J., Tian, Y., Sui, Z.: Fine-grained classification of named entities by fusing multi-features. pp. 693–702 (2012)
21. Li, Y., Li, W., Sun, F., Li, S.: Component-enhanced Chinese character embeddings. In: EMNLP, pp. 829–834 (2015)
22. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF (2016). arXiv:1603.01354v4 [cs.LG]
23. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of International Conference on Learning Representations (ICLR 2013), pp. 1–12 (2013)
24. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 1–9 (2013)
25. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50. ELRA, Valletta, Malta (2010). http://is.muni.cz/publication/884893/en
26. Shi, X., Zhai, J., Yang, X., Xie, Z., Liu, C.: Radical embedding: delving deeper to Chinese radicals. In: Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (vol. 1: Long Papers), pp. 594–598 (2015)
27. Sun, Y., Lin, L., Yang, N., Ji, Z., Wang, X.: Radical-enhanced chinese character embedding. In: Loo, C.K., Yap, K.S., Wong, K.W., Teoh, A., Huang, K. (eds.) ICONIP 2014. LNCS, vol. 8835, pp. 279–286. Springer, Heidelberg (2014). doi:10.1007/978-3-319-12640-1_34
28. Yang, Z., Salakhutdinov, R., Cohen, W.: Multi-task cross-lingual sequence tagging from scratch (2016). arXiv preprint arXiv:1603.06270
29. Zhang, S., Qin, Y., Wen, J., Wang, X.: Word segmentation and named entity recognition for SIGHAN Bakeoff3. In: Proceedings of 5th SIGHAN Workshop on Chinese Language Processing, pp. 158–161 (2006)
30. Zhang, Y., Clark, S.: A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In: Proceedings of 2010 Conference on Empirical Methods in Natural Language Processing, pp. 843–852 (2010)
31. Zhou, J., He, L., Dai, X., Chen, J.: Chinese named entity recognition with a multi-phase model. In: Proceedings of 5th SIGHAN Workshop on Chinese Language Processing, pp. 213–216 (2006)
32. Zhou, J., Qu, W., Zhang, F.: Chinese named entity recognition via joint identification and categorization. Chin. J. Electron. **22**, 225–230 (2013)