LETTER *Special Section on Knowledge-Based Software Engineering*

Character Feature Learning for Named Entity Recognition

Ping ZENG[†,††], **Qingping TAN**[†,†††a)], **Haoyu ZHANG**[†,††], **Xiankai MENG**[†,††], **Zhuo ZHANG**[†,††], **Jianjun XU**[†,††], ***Nonmembers***, ***and*** **Yan LEI**[†††], ***Student Member***

SUMMARY The deep neural named entity recognition model automatically learns and extracts the features of entities and solves the problem of the traditional model relying heavily on complex feature engineering and obscure professional knowledge. This issue has become a hot topic in recent years. Existing deep neural models only involve simple character learning and extraction methods, which limit their capability. To further explore the performance of deep neural models, we propose two character feature learning models based on convolution neural network and long short-term memory network. These two models consider the local semantic and position features of word characters. Experiments conducted on the CoNLL-2003 dataset show that the proposed models outperform traditional ones and demonstrate excellent performance.

key words: *named entity recognition, character representation learning, character feature learning, knowledge extraction*

1. Introduction

Named entity recognition (NER) is an important NLP task. Traditional NER approaches include rule-based and statistical methods. Rule-based methods [1] require linguists or domain experts to design rule templates and usually achieve good results, but they depend heavily on sophisticated manual design and obscure domain knowledge and possess poor versatility. Statistical methods include support vector machine [2], hidden Markov model [3], and condition random field (CRF) [4], which require design rule templates and rely on complex and taxing feature engineering.

Deep neural NER models, which do not rely on complex feature engineering, have received considerable attention recently. [5] proposed the use of a DNN network model for the relationship between a word and a tag. This model is relatively simple, does not consider the existence of contextual information among words, and demonstrates a weak performance. To learn word context semantic associations, [6] proposed the use of long short-term memory network (LSTM) or bidirectional LSTM (BiLSTM) modeling of word sequences. To consider the presence of context association between tags, [6]–[9] proposed adding a linear chain CRF layer above the BiLSTM layer, this addition further enhances the learning capability. These models, which use BiLSTM to model word sequences and CRF to model tag sequences, are called BiLSTM-CRF models and dominant in the field of NER.

These models can effectively model word and tag features, but many of the words to be predicted may not exist in the training set. To address this problem, researchers have attempted to learn word character features by using neural networks. Word characters often have local combinations, and words with similar meanings may have similar constituent structures (e.g., "*sample*" and "*example*"), which are called local semantic features. The local semantic features of words are commonly learned by using convolution neural networks (CNNs) [7]–[13]. In addition, position features between word characters, such as "*pre*", often appear at the beginning of a word, and "*ing*" often appears at the end of the word, character position features are normally learned by using LSTM or BiLSTM [8], [14].

CNN and (Bi)LSTM (short for LSTM/BiLSTM) can learn several semantic features of a word. However, CNN is ineffective in modeling the positional features of character sequences, and (Bi)LSTM is not as good as CNN in modeling the local semantics of words. To synthesize the adventges of CNN and (Bi)LSTM, this study developed a CNN+(Bi)LSTM concatenation model and a CNN+(Bi)LSTM stack model, which can learn the local semantic and position features of word sequences. Experimental results showed that the CNN+BiLSTM concatenation and CNN+LSTM stack models, which obtained 91.58% and 91.52% F1 scores on the CoNLL-2003 dataset, respectively, possessed a strong learning capability and outperformed state-of-the-art models.

The main contributions of this work are as follows. First, it evaluates the effect of different character feature learning modules on NER tasks. Second, to the best of our knowledge, this study is the first to jointly use CNN and (Bi)LSTM to model character features. Lastly, the proposed models achieve excellent performance.

2. Our Models

We referred to the BiLSTM-CRF model and proposed two models that can learn many character features. The two models integrate the semantic information contained in a word into the word embedding task by concatenation or stacking. These end-to-end models do not require complex

Manuscript received August 15, 2017.
Manuscript publicized April 20, 2018.

[†]The authors are with College of Computer, National University of Defense Technology, Changsha, China.
[††]The authors are with National Key Laboratory for Parallel and Distributed Processing, Changsha, China.
[†††]The author is with the School of Software Engineering, Chongqing University, Chongqing, China.
a) E-mail: eric_tan@nudt.edu.cn

DOI: 10.1587/transinf.2017KBL0001

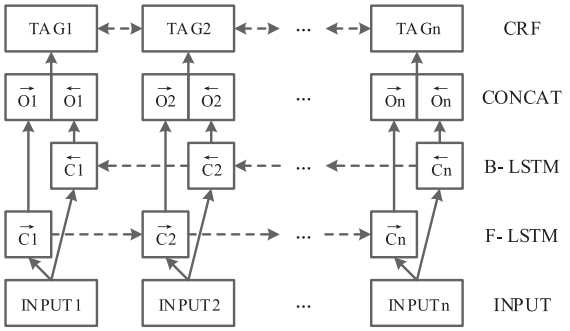

Fig. 1 Overall architecture of the model

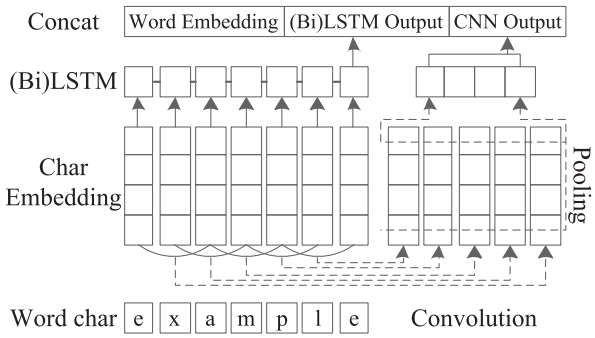

Fig. 2 Learning word character features by concatenation

data preprocessing.

2.1 Overall Architecture

The model consists of an input layer, a BiLSTM layer, a concatenation layer, and a CRF layer. The overall model architecture is shown in Fig. 1. The input layer consists of processed word vectors (called input vectors), and the number of input vectors is equal to the number of words in the sentence. The input vector is entered into a two-way LSTM network consisting of forward and backward LSTMs, and the time step is equal to the number of input vectors. The output of BiLSTM is concatenated into a new vector in series and inputted to CRF.

This model is a variant of the BiLSTM-CRF model. The difference is that our model introduces CNN and (Bi)LSTM modules at the input layer. The two modules learn word character features by concatenation or stacking, and these character features are concatenated together with word embedding to form the input vector.

2.2 Concatenation Model for Character Feature Learning

Figure 2 shows the process of learning character features by concatenation. The character vectors are fed to the (Bi)LSTM and CNN modules to calculate the output values. These output values are eventually concatenated with word embedding (query from the word embedding table) into a new input vector (corresponding to the INPUT layer vector in Fig. 1). The rest of the calculation processes do not interfere with one another, except for sharing the same character vector.

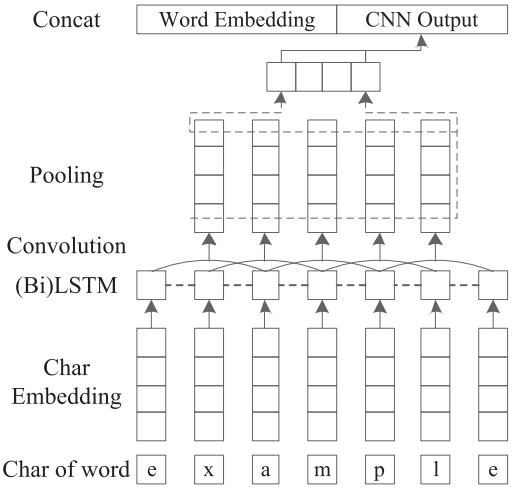

Fig. 3 Learning word character features by stacking

The concatenation method preserves the word embedding, the CNN vector, and the (Bi)LSTM vector by considering the semantic features of the word itself, the local semantic features, the position-related features of the word characters, and so on. The implication feature is rich, and the mutual interference is small.

2.3 Stack Model for Character Feature Learning

Figure 3 shows the process of learning character features by stacking. The character vector is entered into the (Bi)LSTM module, and the output of each step in the (Bi)LSTM module is regarded as the input of the CNN module. Word embedding is then concatenated with the output of the CNN module into a new input vector (corresponding to the INPUT layer vector in Fig. 1). As a result, the input vector dimension is smaller than the concatenation model, and the character position feature can be fused to the local semantic features via implicit stacking.

2.4 Model Details and Training

All of the BiLSTM units in the model are consistent with that mentioned in [16]. All LSTM units are defined in the same manner as that in [17]. The forget gate bias is initialized to 1, and the other vectors are initialized to 0. The CNN in the character feature module uses the method mentioned in [7], and the pooling layer uses the max pooling method.

All matrix parameters are initialized by the uniform sampling $U \sim \left[-\sqrt{6/(r+c)}, +\sqrt{6/(r+c)}\right]$ (r is the number of matrix rows and c is the number of matrix columns) mentioned in [18], [19]. All word embedding or character vectors are initialized by $U \sim \left[-\sqrt{3/d}, +\sqrt{3/d}\right]$ (d is the dimension of the vector) and fine-tuned in the training process.

We used the CRF mentioned in [20] to encode NER

tags. In the training process, we used the backpropagation (BP) algorithm [21], backpropagation through time (BPTT) algorithm [17], and momentum mini-batch random gradient descent optimizer with a dynamic decay learning rate to optimize the training parameters, word embedding, and character vectors. The training objective of the model is to maximize the likelihood function $L(\theta)$ as follows:

$$\begin{cases} L(\boldsymbol{\theta}) = \sum \log p(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}) \\ p(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}) = \dfrac{\prod_{i=1}^{n} f_i(y_{i-1}, y_i, \mathbf{x})}{\sum_{y' \in y(\mathbf{x})} \prod_{i=1}^{n} f_i(y'_{i-1}, y'_i, \mathbf{x})} \\ f_i(y_{i-1}, y_i, \mathbf{x}) = \exp(\boldsymbol{\theta}(\mathbf{X}_i)) = \exp(\mathbf{W}\mathbf{X}_i + \mathbf{b}) \end{cases},$$

where θ denotes the training parameter, $\mathbf{x}$ denotes the input vector, $\mathbf{y}$ denotes the target tag corresponding to $\mathbf{x}$, $y(\mathbf{x})$ denotes the set of all possible tags corresponding to $\mathbf{x}$, $\mathbf{W}$ denotes the weight matrix, and $\mathbf{b}$ denotes the bias. In the prediction process, the Viterbi algorithm [22] is used for decoding. The objective function is

$$\mathbf{y}^* = \arg\max_{\mathbf{y} \in y(\mathbf{x})} p(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}).$$

To improve the generalizability of the model, we applied dropout [23] on the character vector before inputting to the CNN layer and on the input and output vectors of the BiLSTM layer. Before entering the model, the data were shuffled by batch. In the validation set, we used the early stopping method [24] to prevent over-training. These methods were proven to be valid by [8], [9].

3. Experiments

We used the Lasagne framework [25] to implement the model. On the basis of the CoNLL-2003 [15] English dataset, we evaluated the effects of the character learning components of the model and compared them with those obtained by Chiu and Nichols [7], Luo et al. [26], Lample et al. [8], and Ma and Hovy [9]. Other models were also compared.

3.1 Datasets

We used the CoNLL-2003 English dataset for experimental evaluation. The training set of the dataset contains 14,987 sentences and 204,567 tags. The validation set contains 3,466 sentences and 51,578 tags. The test set contains 3,684 sentences and 46,666 tags. We used the BIOES tagging scheme [8], [9] to process the data and improve the model's capabilities. We also used Glove 100-dim word embedding data [27] as the initial input vector of a word. Words that do not appear in Glove were randomly initialized, as described in Sect. 2.6.

3.2 Parameter Setting

We set the model parameters according to [9], as shown in Table 1. We set the word character feature dimension to 30. Thus, the number of BiLSTM units was set to 15 (the final output dimension was 30 after concatenation). The number of LSTM units was set to 30, and the CNN filter was also set to 30. According to these settings, the word feature dimension is 260 in the final concatenation model and 230 in the stack model.

Table 1 Parameter setting for the proposed model

Layer	Parameter Name	Value
Character Layer	CNN window size	3
	CNN filter size	30
	Number of LSTM units	30
	Number of BiLSTM units	15
BiLSTM Layer	Number of BiLSTM units	200
	Initial state	0.0
	Initial bias	1.0
Other	Batch size	10
	Dropout rate	0.5
	Initial learning rate	0.015
	Learning rate decay rate	0.05
	Early stopping patient	15
	Gradient clipping	5.0

Table 2 Results of the model component test

Character model	Acc.	Prec.	Recall	F1
None	97.54	90.80	88.33	89.55
CNN	97.92	91.02	91.44	91.23
BiLSTM	98.02	91.34	91.37	91.35
LSTM	97.59	90.64	88.64	89.63
CONCAT(CNN+BiLSTM)	**98.03**	91.29	**91.75**	91.52
CONCAT(CNN+LSTM)	97.99	91.25	91.48	91.36
STACK(CNN+BiLSTM)	**98.03**	91.35	91.50	91.42
STACK(CNN+LSTM)	98.02	**91.46**	91.69	**91.58**

3.3 Results

To comprehensively verify the validity of the character feature learning module and the overall learning capability of the model, we conducted component and comparison tests with other models and used the conlleval perl script [28] to calculate the output.

The main purpose of the component test was to verify the effect of character features on the overall learning capability of the model. We tested the learning capability of various character feature components by replacing the character feature learning module in the overall model. The results are shown in Table 2. The results demonstrate that the CNN+LS TM stack and CNN+BiLSTM concatenation models have the best learning capability because both models can learn the local semantic and position features of word characters. BiLSTM is generally superior to LSTM because it is more capable of capturing semantic associations among characters. CNN is better than LSTM in our experiments, this finding is consistent with the results reported in previous work.

Table 3 shows the F1 scores of the main NER models established after 2009. As shown in the table, our model has a higher F1 score than previous models. The STACK(CNN +LSTM)-BiLSTM-CRF and CONCAT(CNN+BiLSTM)-BiLSTM-CRF models showed the best performance in the CoNLL-2003 dataset. This result indicates that the

Table 3 Related NER work comparison. Given that the same evaluation criteria and datasets were used, we directly referred to the results reported by relevant work. * The highest F1 score of Ma and Hovy's model in our environment is 91.23.

Model	F1
Ratinov and Roth (2009)	90.80
Lin and Wu (2009)	90.90
Collobert et al. (2011)	89.59
Passos et al. (2014)	90.90
Huang et al. (2015)	90.10
Chiu and Nichols (2015)	90.77
Luo et al. (2015)	91.20
Lample et al. (2016)	90.94
Ma and Hovy(2016)*	91.23
Our STACK(CNN+LSTM)	**91.58**
Our STACK(CNN+BiLSTM)	**91.42**
Our CONCAT(CNN+LSTM)	**91.36**
Our CONCAT(CNN+BiLSTM)	**91.52**

recognition capability of the NER model can be further improved by fully learning the character features.

4. Conclusion

We investigated the influence of character feature learning on NER tasks and proposed two models of learning local semantic and position-related features in word characters by concatenation and stacking. To the best of our knowledge, the proposed models are the first to use CNN and (Bi)LSTM to learn character features in NER tasks.

The experiments showed that the concatenation and stack models performed well in different character feature learning tasks. CONCAT-BiLSTM-CRF and STACK-BiLSTM-CRF models outperformed previous models.

In the future, we will consider using the attention mechanism [29] to enhance model capabilities at the character feature level to learn the combined weights of different character modules. We will also consider the attention mechanism at the word level to model the correspondence among tags and words.

Acknowledgments

This study is supported by the Scientific Research Fund of Hunan Provincial Education Department (No. 15A007) and the National Natural Science Foundation of China (No. 61202116, No.61602504).

References

[1] K. Riaz, "Rule-based named entity recognition in Urdu," Proc. 2010 named entities workshop, pp.126–135, Association for Computational Linguistics, July 2010.

[2] M. Asahara and Y. Matsumoto, "Japanese Named Entity extraction with redundant morphological analysis," Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, vol.2003, pp.8–15, Association for Computational Linguistics, 2003.

[3] Y. Benajiba and P. Rosso, "Arabic named entity recognition using conditional random fields," Proc. Workshop on HLT & NLP within the Arabic World, LREC, vol.8, pp.143–153, May 2008.

[4] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," Proc. seventh conference on Natural language learning at HLT-NAACL 2003, vol.4, pp.188–191, Association for Computational Linguistics, May 2003.

[5] 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, pp.2493–2537, Aug. 2011.

[6] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015. arXiv preprint arXiv:1508.01991.

[7] J.P. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," Transactions of the Association for Computational Linguistics, vol.4, pp.357–370, 2016.

[8] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," Proceedings of NAACL-HLT 2016, pp.260–270, 2016.

[9] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp.1064–1074, 2016.

[10] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," Advances in Neural Information Processing Systems, pp.2042–2050, 2014.

[11] C.D. Santos and B. Zadrozny, "Learning character-level representations for part-of-speech tagging," Proc. 31st International Conference on Machine Learning (ICML-14), pp.1818–1826, 2014.

[12] M. Labeau, K. Löser, A. Allauzen, and R.J. von Neumann, "Non-lexical neural architecture for fine-grained POS Tagging," EMNLP, pp.232–237, Sept. 2015.

[13] C.N.D. Santos and V. Guimaraes, "Boosting named entity recognition with neural character embeddings," Proceedings of the Fifth Named Entity Workshop, joint with 53rd ACL and the 7th IJCNLP, pp.25–33, 2015.

[14] M. Rei, G.K. Crichton, and S. Pyysalo, "Attending to characters in neural sequence labeling models," Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics, pp.309–318, 2016.

[15] E.F.T.K. Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," Proc. seventh conference on Natural language learning at HLT-NAACL 2003, vol.4, pp.142–147, Association for Computational Linguistics, May 2003.

[16] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N.A. Smith, "Transition-based dependency parsing with stack long short-term memory," Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp.334–343, 2015.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol.9, no.8, pp.1735–1780, 1997.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," Proc. Thirteenth International Conference on Artificial Intelligence and Statistics, pp.249–256, March 2010.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," Proc. IEEE international conference on computer vision, pp.1026–1034, 2015.

[20] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[21] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representation by back-propagation of errors," Nature, vol.323, pp.533–536, 1986.

[22] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE Trans. Inf. Theory, vol.13, no.2, pp.260–269, 1967.

[23] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks

from overfitting," Journal of Machine Learning Research, vol.15, no.1, pp.1929–1958, 2014.

[24] R. Caruana, S. Lawrence, and C.L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," Advances in Neural Information Processing Systems, pp.402–408, 2001.

[25] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S.K. Snderby, D. Nouri, … and J. De Fauw, Lasagne: First release, http://dx.doi.org/10.5281/zenodo.27878, Aug. 2015.

[26] G. Luo, X. Huang, C.Y. Lin, and Z. Nie, "Joint named entity recognition and disambiguation," Proc. EMNLP, pp.879–880, Sept. 2015.

[27] J. Pennington, R. Socher, and C.D. Manning, "Glove: Global vectors for word representation," EMNLP, vol.14, pp.1532–1543, Oct. 2014.

[28] E. Tjong and K. Sang, conlleval, July 26, 2017, URL http://www.cnts.ua.ac.be/conll2002/ner/bin/conlleval.txt.

[29] M.T. Luong, H. Pham, and C.D. Manning, "Effective approaches to attention-based neural machine translation," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp.1412–1421, 2015.