Named Entity Recognition with Gated Convolutional Neural Networks

Chunqi Wang[1,2], Wei Chen[2] Bo Xu[2]

[1] University of Chinese Academy of Sciences
[2] Institute of Automation, Chinese Academy of Sciences
chqiwang@126.com, {wei.chen.media, xubo}@ia.ac.cn

Abstract. Most state-of-the-art models for named entity recognition (NER) rely on recurrent neural networks (RNNs), in particular long short-term memory (LSTM). Those models learn local and global features automatically by RNNs so that hand-craft features can be discarded, totally or partly. Recently, convolutional neural networks (CNNs) have achieved great success on computer vision. However, for NER problems, they are not well studied. In this work, we propose a novel architecture for NER problems based on GCNN — CNN with gating mechanism. Compared with RNN based NER models, our proposed model has a remarkable advantage on training efficiency. We evaluate the proposed model on three data sets in two significantly different languages — SIGHAN bakeoff 2006 MSRA portion for simplified Chinese NER and CityU portion for traditional Chinese NER, CoNLL 2003 shared task English portion for English NER. Our model obtains state-of-the-art performance on these three data sets.

1 Introduction

Named entity recognition (NER) is a challenging task in natural language processing (NLP) community. On the one hand, there is only very small amount of data for supervised training in most languages and domains. On the other hand, there are few constraints on the kinds of words that can be a name entity so that the distribution of name entities are sparse. Sparse distribution is typically difficult for models to generalize. NER is also a popular NLP task and plays a vital role for downstream systems, such as machine translation systems and dialogue systems.

Traditional NER systems are often linear statistical models, such as Hidden Makov Models (HMM), Support Vector Machines (SVM) and Conditional Random Fields (CRF) [24, 21, 18]. These models rely heavily on hand-craft features and language dependent resources. For example, gazetteers are widely used in NER systems. However, such features and resources are costly to develop and collect.

Recent years, non-linear neural networks are getting more and more interests. Collobert [6] proposed a unified architecture for sequence labeling tasks,

including NER, chunking and part-of-speech (POS) tagging, semantic role labeling (SRL). They introduced two approaches — a feed-forward neural network (FNN) approach and a convolutional neural network (CNN) [16] approach. Neural networks are able to learn features automatically and thus alleviate reliance on hand-craft features. Besides, large scale of unlabeled corpus can be used to boost performance in a multi-task manner. Recently, recurrent neural networks (RNNs), together with its variants long short-term memory (LSTM) [10] and gated recurrent unit (GRU) [5], have shown great success in NLP community [28, 29, 2]. As for NER, there are a series of works that are based on RNN [11, 4, 8, 15, 19]. Ma [19] proposed an end-to-end model that requires no hand-craft feature or data preprocessing. Despite the excellent performance of RNN based models, they are difficult to parallelize over sequence. In this perspective, CNNs have great advantages. In this paper, we propose a novel architecture for NER problems based on CNN. Instead of recurrent layers, we adopt hierarchical convolutional layers to extract features from raw sentence. We also introduce gating mechanism into the convolutional layer to allow more flexible information control. Compared to RNN based models, our model is training faster, and perform better.

We evaluate the proposed model on three benchmark data sets for two significantly different languages — SIGHAN bakeoff 2006 MSRA portion for simplified Chinese NER, SIGHAN bakeoff 2006 CityU portion for traditional Chinese NER and CoNLL 2003 shared task English portion for English NER. Our model obtains state-of-the-art performance on these three data sets. Contributions of this work are: (i) We propose a novel architecture for NER problems. (ii) We evaluate our model on three benchmark data sets for two significantly differently languages — Chinese and English. (iii) Our model obtains state-of-the-art performance on these three data sets.

2 Architecture

In this section, we describe our network architecture from bottom to top.

2.1 CNN for Encoding English Word Information

In this work, we focus on two significantly different languages: Chinese and English. In Chinese, there is no separator between words in sentences. There are mainly two approaches to handle it. One of them is to use an upstream system to segment words and feed the words into NER systems. The other is to feed the characters directly into the systems. We choose the latter approach to cut off the dependence with upstream systems. In English, unlike Chinese, there are separators, i.e. blanks, between words, therefore we adopt words as the basic input unit.

For Chinese, characters are transformed to character embeddings. Similarly, for English, words are transformed to word embeddings. However, information about word morphology is not included in word embedding, which is often crucial

for various NLP tasks. Several previous works [27, 26, 4, 19] have shown that CNN is effective in extracting morphological features from characters. In this work, we adopt a simillar network. The network accepts characters (of a word) as inputs and output a fixed dimention vector. Architecture of the network is shown in Fig. 1. The output vector is concatenated with the word embedding and fed into upper layers. Note that for Chinese, we do not need the network.

2.2 Deep CNN with Gating Mechanism

Currently, for NER problems, the main-stream approach is to consider a sentence as a sequence of tokens (characters or words) and to process them with a RNNs [11, 4, 8, 15, 19]. In this work, we adopt a novel strategy which is significantly different from previous works. Instead of RNN, we use hierarchical CNN to extract local and context information. We introduce gating mechanism into the convolutional layer. Dauphin [7] have shown that gating mechanism is useful for language modeling tasks. Fig. 2 shows the structure of one gated convolutional layer.

Formally, we define the number of input channels as N, the number of output channels as M, the length of input as L and kernel width as k. A gated convolutional layer can be written as

$$F_{gating}(X) = (X * W + b) \otimes \sigma(X * V + c) \tag{1}$$

where $*$ denotes row convolution, $X \in R^{L \times N}$ is the input of this layer, $W \in \mathbb{R}^{k \times N \times M}$, $b \in \mathbb{R}^N$, $V \in \mathbb{R}^{k \times N \times M}$, $c \in \mathbb{R}^N$ are parameters to be learned, σ is the sigmoid function and $\otimes$ represent element-wise product. We make $F_{gcnn}(X) \in \mathbb{R}^{L \times M}$ by augmenting X with padding.

Multiple gated convolutional layers are stacked to capture long distance information. On the top of the last layer, we use a linear transformation to transform output of the network to unnormalized scores of labels $E \in \mathbb{R}^{L \times C}$, where L is the length of a given sentence and C is the number of labels.

2.3 Linear Chain CRF

Though deep neural networks have the ability to capture long distance information, it has been verified that considering the correlations between adjacent labels can be very beneficial in sequence labeling problems [6, 11, 15, 19].

Correlations between adjacent labels can be modeled as a transition matrix $T \in \mathbb{R}^{C \times C}$. Given a sentence

$$\mathcal{S} = (w_1, w_2, ..., w_L) \tag{2}$$

we have corresponding scores $E \in \mathbb{R}^{L \times C}$ given by the CNN. For a sequence of labels $y = (y_1, y_2, ..., y_L)$, we define its unnormalized score to be

$$s(\mathcal{S}, y) = \sum_{i=1}^{L} E_{i,y_i} + \sum_{i=1}^{L-1} T_{y_i, y_{i+1}} \tag{3}$$

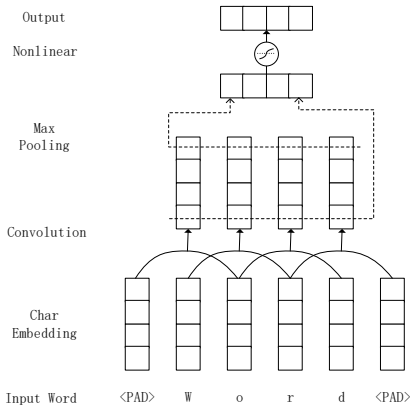

Fig. 1. Convolutional neural network for encoding English word information.

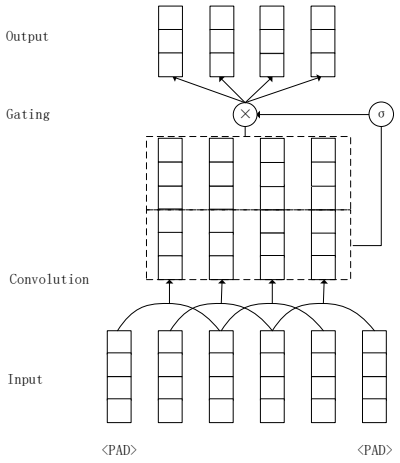

Fig. 2. Structure of one convolutional layer with gating mechanism.

Probability of the sequence of labels then be defined as

$$P(y|\mathcal{S}) = \frac{e^{s(\mathcal{S},y)}}{\sum_{y\prime\in\mathcal{Y}} e^{s(\mathcal{S},y\prime)}} \tag{4}$$

where $\mathcal{Y}$ is the set of all valid sequences of labels. This formulation is actually linear chain conditional random field (CRF) [14]. The final loss of the proposed model then be defined as the negative log-likelihood of the ground-truth sequence of labels y^*

$$\mathcal{L}(\mathcal{S}, y^\star) = -logP(y^\star|\mathcal{S}) \tag{5}$$

During training, the loss function is minimized by back propagation. During test, Veterbi algorithm is applied to find the label sequence with maximum probability.

3 Experimental Setup

3.1 Data Sets

We evaluate our model on three data sets. Two of them are Chinese and another is English.

MSRA is a data set for simplified Chinese NER. It comes from SIGHAN 2006 shared task for Chinese NER [17]. There are three types of entities tagged in this data set: PERSON, LOCATION and ORGANIZATION. Within the training data, we hold the last $\frac{1}{10}$ for development.

CityU is a data set for traditional Chinese NER. Same with the first data set, it also comes from SIGHAN 2006 shared task for Chinese NER and is tagged with the same three types of entities. We hold the last $\frac{1}{10}$ of training data for development.

CoNLL-2003 is a data set for English NER. It comes from CoNLL 2003 shared task [30]. There are four types of entities tagged in this data set: PERSON, LOCATION, ORGANIZATION and MISCELLANEOUS.

Note that we do not perform any preprocessing for these data sets. Our system is truly end-to-end.

3.2 LSTM Baseline

Since there is no appropriate public LSTM results for two Chinese data sets — MSRA and CityU, we implemented a LSTM baseline model for Chinese NER. Our baseline LSTM model is almost the same with [11]. However, there are still some differences: (i) Our baseline model accepts characters instead of words as input. (ii) Our baseline model does not use any hand-craft features. (iii) Our baseline model adopts dropout [9] in the same way with [31]. (iv) Our baseline model has more than one layer.

3.3 Dropout

Dropout [9] is a very efficient and simple way for preventing overfitting, especially when the data set is small. We apply dropout to our model on the top of all convolutional layers and embedding layers (including character embedding layer and word embedding layer) with a fixed dropout rate. We observed remarkable improvement when dropout was applied.

3.4 Tagging Scheme

We didn't pay much attention to tagging scheme. For two Chinese data sets, we use simple IOB format (Inside, Outside, Beginning). For another English data set, we use IOBES format (Inside, Outside, Beginning, End, Singleton) to keep consistent with previous works [4, 15, 19].

3.5 Pretrained Embeddings

Following Collobert [6], we use pretrained embeddings. For simplified Chinese data set MSRA, we train the character embeddings on news corpus collected by Sogou [3] with an open source tool word2vec [20]. For traditional Chinese data set CityU, we train the character embeddings on wikipiedia corpus, also by word2vec. As for English data set CoNLL-2003, we use Standford's publicly available Glove word embeddings [4] trained on 6 billion words from Wikipedia and web text [22].

We fine-tune the embeddings while training with normal back propagation.

[3] http://www.sogou.com/labs/resource/ca.php

[4] http://nlp.stanford.edu/projects/glove/

3.6 Hyper-parameters

We tune the hyper-parameters on development set by random search for each data set. For MSRA and CityU, we select the size of character embeddings from {100, 200}. For CoNLL-2003, we select the size of word embeddings from {100, 200}, the size of character embeddings from {20, 40, 60, 80}. For all data sets, we select the number of convolutional channels from {100, 200}, the kernel size from {3, 5}, the dropout rate from {0.2, 0.5} and the network depths from {3, 7, 11, 15}. The selected hyper-parameters are shown in Table 1.

We use the same way to tune hyper-parameters of our baseline LSTM model. We list these hyper-parameters in Table 2.

Table 1. Hyper-parameters we choose for our model on three data sets. SZ_womb refer to the size of word embeddings. SZ_cemb refer to the size of character embeddings. N_channels refer to the number of convolutional channels. SZ_kernel refer to the size of convolutional kernel. Depth refer to the number of convolutional layers.

Data Set	SZ_womb	SZ_cemb	N_channels	SZ_kernel	Dropout	Depth
MSRA	-	200	200	3	0.2	15
CityU	-	200	200	3	0.2	11
CoNLL-2003	100	60	200	3	0.5	3

Table 2. Hyper-parameters we choose for our LSTM baseline model on two Chinese data sets. SZ_cemb refer to the size of character embedding. SZ_hidden refer to the size of LSTM hidden state. Depth refer to the number of bi-directional LSTM layers.

Data Set	SZ_cemb	SZ_hidden	Dropout	Depth
MSRA	200	200	0.2	4
CityU	200	200	0.2	3

3.7 Optimization

For MSRA and CityU, parameter optimization is performed with Adam [13] and the initial learning rate are is to 0.001. We set the batch size to 100. This setting behaves well on both data sets. For CoNLL-2003, we use stochastic gradient descent (SGD) with a fixed learning rate 0.005 and a smaller batch size 20. We tried Adam on this data set, with a larger batch size. However, severe overfitting was obeserved.

We adopt weight normalization [25] — a simple but effective method, on all convolutional layers to accelerate the training procedure.

4 Experimental Result

4.1 Main Result

Table 3 and Table 4 give performances of the proposed model (GCNN) on three data sets. For comparison, we also list the scores of some other models. To make the comparison between our model and others fair, we mark the models that use external labeled data. On MSRA data set, our model outperforms the LSTM baseline in a large margin (1.05 in F1). On CItyU and CoNLL-2003 data sets, our model outperforms baseline slightly. Our model abtains state-of-the-art performance on three data sets without any data preprocessing, hand-craft features and external labeled data.

It seems unbelievable that our model can outperform LSTM based models. Intuitively, our model can only see a local window with limited size at every moment[5] , while LSTM based models can see the whole sentence. It is well known that a fully connected one hidden layer neural network can in principle learn any real-valued function, but much better results can be obtained with a deep problem-specific architecture which develops hierarchical representations. CNNs for computer vision are examples of this. Analogously, deep CNNs may surpass RNNs in NER problems despite the fact that RNNs have strong ability for sequence modeling.

We also explore the impact of gating mechanism, CRF and pretrained embeddings. For each data set, We train other three models with same hyper-parameters but with the absence of gating mechanism, CRF and pretrained embeddings, respectively. The differences between them and the original model are descirbed below:

CNN (-gating) We remove gating mechanism and instead use rectified linear unit (ReLU). Formally, the layer is defined as

$$F_{relu}(X) = Relu(X * W + b) \tag{6}$$

GCNN (-crf) We remove transition scores from Equation 3. Specifically, we substitute Equation 3 with

$$s(\mathcal{S}, y) = \sum_{i=1}^{L} E_{i,y_i} \tag{7}$$

GCNN (-pretrain) We use random initialized embeddings instead of pre-trained embeddings.

As shown in Table 3 and Table 4, pretrained embeddings give us the biggest improvement in all of the three data sets (+5.18 in average). Gating mechanism and CRF also give us remrakable improvement (+0.54 and +0.68 respectively).

[5] The window size equals to $d \times (k-1)+1$, where d is the network depth and k is the kernel size.

Table 3. Comparison with previous works on MSRA and CityU data set. ⋆ indicates models trained with the use of external labeled data.

Model	MSRA			CityU		
	Precision	Recall	F1	Precision	Recall	F1
Zhou [12]	88.94	84.20	86.51	-	-	-
Zhou [12]⋆	90.76	89.22	89.99	-	-	-
Chen [3]	91.22	81.71	86.20	92.66	84.75	88.53
Zhao [32]	-	-	86.30	-	-	89.18
Zhou [33]	91.86	88.75	90.28	92.33	87.37	89.78
LSTM	91.14	89.24	90.18	92.01	89.27	90.62
CNN (-gating)	91.56	90.02	90.78	91.42	89.19	90.29
GCNN (-crf)	91.06	89.50	90.27	91.51	89.07	90.27
GCNN (-pretrain)	89.59	83.55	86.46	89.76	86.26	87.98
GCNN	**92.34**	**90.15**	**91.23**	**92.68**	**88.71**	**90.65**

Table 4. Comparison with previous works on CoNLL-2003 data set. ⋆ indicates models trained with the use of external labeled data.

Model	Precision	Recall	F1
Huang [11]	-	-	90.10
Luo [18]⋆	91.50	91.40	91.20
Passos [21]	-	-	90.90
Lample [15]	-	-	90.94
Ma [19]	91.35	91.06	91.21
CNN (-gating)	90.02	90.86	90.44
GCNN (-crf)	90.37	90.69	90.53
GCNN (-pretrain)	83.55	83.45	83.50
GCNN	**91.39**	**91.09**	**91.24**

Table 5. The sum of training time and validation time of 100 epoches of various models on MSRA data set.

Model	Depth	Time (hour)	F1
LSTM	1	20	88.36
	2	36	90.07
	3	46	90.00
	4	60	90.18
GCNN	3	6	89.84
	7	12	89.64
	11	16	90.96
	15	21	**91.23**

4.2 Network Depth

With the network grows deeper, the network has stronger representation ability in principle and can see a larger context and thus higher performance is expected. Fig. 3 shows the F1 scores on three data sets with different network depth settings. With the network grows deeper, the performance of the network has a rising trend on MSRA and CityU data sets. However, we can also observe a degradation phenomenon. On CoNLL-2003 data set, a deeper network with 7 layers has a much lower score than the shallow one with only 3 layers. We leave this phenomenon for further research.

4.3 Training Efficiency

In this section, we show that our proposed model has a remarkable advantage on training efficiency compared to LSTM based models. Table 5 give summaries

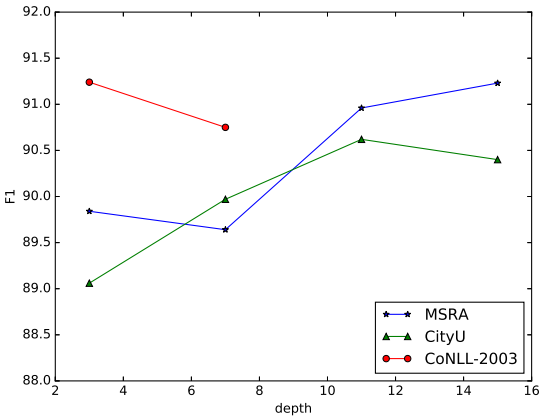

Fig. 3. Performance of our model with various depth on three data sets. On CoNLL-2003 data set, we oberserved severe degradation phenomenon as we increase network depth from 3 to 7, so we didn't try further depth.

of the training time of 100 epoches for these models on MSRA data set. We train each model in a single K20 GPU with the same batch size (100 sentences per batch) and platform (*tensorflow* [1]), therefore the statistics is reliable. Our proposed model with 15 convolutional layers consumed roughly the same amount of time with the LSTM based model with 1 bi-directional LSTM layer. However, the former outperforms the latter in a large margin (2.87 in F1). Moreover, the LSTM based model with the best performance has 4 bi-directional LSTM layers and consumed 60 hours but the performance is much lower than our CNN based model with 11 convolutional layers, which only consumed 16 hours. Our proposed model has much higher training efficiency than LSTM based models in that LSTMs are extremely deep if we unroll them over the whole sequence.

5 Related Work

There are primarily two kinds of approaches for NER problems. One is linear models, like CRF and SVM, together with carefully designed hand-craft features, as well as external knowledges. The other kind is neural network based models. Recent years, neural network based models take over the dominate position. Collobert [6] proposed a unified architecture based on FNN for sequence labeling problems, including part-of-speech (POS) tagging, NER and chunking. Their model utilizes word embeddings and thus large amount of unlabeled data can be used to pretrain the embeddings to boost NER performance. dos [26] extend their architecture for NER problems with character embeddings. Our model can be seen as an extension of their model, where the FNN is replaced with a deep CNN. Collobert [6] also proposed a CNN based model. However,

their model is significantly different from ours since theirs adopt max-pooling to encode the whole sentence into a fixed size vector and use position embeddings to demonstrate which word to be tagged while ours does not use max-pooling and thus position embeddings are not required.

Recently, RNN based approaches, together with LSTM, are prevailing. Huang [11] used bi-directional LSTM for word-level feature extraction and CRF for sequence prediction. Their model didn't take character-level information into consideration. Chiu [4] also used bi-directional LSTM for word-level feature extraction. Besides, they utilized CNN for character-level feature extraction. Unlike Huang [11], they didn't use CRF. Lample [15] proposed a model that utilizes a bi-directional LSTM for word-level feature extraction and another bi-directional LSTM for character-level feature extraction, as well as a CRF layer for sequence prediction. Ma [19] proposed a model similar with Lample [15] but use CNN instead of bi-directional LSTM for character-level feature extraction. Gillick [8] applied RNN based encoder-decoder architecture Sutskever [29] for NER problems.

Our model is significantly different with previous RNN based models in that we only use CNN for feature extraction. For English NER, we adopt CNN for character-level feature extraction, which is simillar with dos [26], Chiu [4] and Ma [19]. We also adopt CRF for sentence level prediction as Collobert [6], dos [26], Huang [11], Lample [15] and Ma [19] did.

Our work is also inspired by wroks on language modeling [23, 7]. Pham [23] applied CNN for language modeling and their model outperforms RNNs but is below state of the art LSTM based models. Dauphin [7] extended CNN with gating mechanism for language modeling and achieved a new state of the art on WikiText-103 as well as a new best single-GPU result on the Google Billion Word benchmark.

6 Conclusion and Future Work

We proposed a novel architecture based on gated CNN for solving NER problems of different languages. We evaluated our model on three benchmark data sets in two significantly different languages — Chinese and English, and achieved state-of-the-art performance on all of the three data sets. Compared to prevailing LSTM based models, our model abtains better performance and has a remarkable advantage on training efficiency.

There are two future works worth studying. One is to make the network deeper without degradation so that larger context can be utilized by the network and thus may yield better performance. The other interesting direction is to apply the architecture to other sequence labeling tasks, like POS tagging, chunking etc.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on

heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. Computer Science (2014)
3. Chen, A., Peng, F., Shan, R., Sun, G.: Chinese named entity recognition with conditional probabilistic models pp. 173–176 (2006)
4. Chiu, J.P.C., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. Computer Science (2015)
5. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
6. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research 12(1), 2493–2537 (2011)
7. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks (2016)
8. Gillick, D., Brunk, C., Vinyals, O., Subramanya, A.: Multilingual language processing from bytes. arXiv preprint arXiv:1512.00103 (2015)
9. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R., Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. Computer Science 3(4), pgs. 212–223 (2012)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)
11. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
12. Junsheng, Z., Liang, H., Xinyu, D., Jiajun, C.: Chinese named entity recognition with a multi-phase model. In: COLING ACL 2006. p. 213 (2006)
13. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. Computer Science (2014)
14. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the eighteenth international conference on machine learning, ICML. vol. 1, pp. 282–289 (2001)
15. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition (2016)
16. LeCun, Y., Boser, B., Denker, J., Henderson, D.: Backpropagation applied to handwritten zip code recognition. Neural Computation 1(4), 541–551 (1989)
17. Levow, G.A.: The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In: Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing. pp. 108–117 (2006)
18. Luo, G., Huang, X., Lin, C.Y., Nie, Z.: Joint entity recognition and disambiguation. In: Conference on Empirical Methods in Natural Language Processing. pp. 879–888 (2015)
19. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional lstm-cnns-crf (2016)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems 26, 3111–3119 (2013)
21. Passos, A., Kumar, V., Mccallum, A.: Lexicon infused phrase embeddings for named entity resolution. Computer Science (2014)

22. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Conference on Empirical Methods in Natural Language Processing. pp. 1532–1543 (2014)
23. Pham, N.Q., Kruszewski, G., Boleda, G.: Convolutional neural network language models. In: Proc. of EMNLP (2016)
24. Ratinov, L., Roth, D.: Conll 09 design challenges and misconceptions in named entity recognition. In: CoNLL '09: Proceedings of the Thirteenth Conference on Computational Natural Language Learning. pp. 147–155 (2009)
25. Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks (2016)
26. dos Santos, C., Guimaraes, V., Niterói, R., de Janeiro, R.: Boosting named entity recognition with neural character embeddings. In: Proceedings of NEWS 2015 The Fifth Named Entities Workshop. p. 25 (2015)
27. dos Santos, C.N., Zadrozny, B.: Learning character-level representations for part-of-speech tagging. In: ICML. pp. 1818–1826 (2014)
28. Sundermeyer, M., Schlüter, R., Ney, H.: Lstm neural networks for language modeling. In: Interspeech. pp. 194–197 (2012)
29. Sutskever, I., Vinyals, O., Le, Q.V., Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems 4, 3104–3112 (2014)
30. Tjong, Kim Sang, E.F., De Meulder, F.: Introduction to the conll-2003 shared task: language-independent named entity recognition. Computer Science 21(08), 142–147 (2003)
31. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
32. Zhao, H., Kit, C.: Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In: IJCNLP. pp. 106–111. Citeseer (2008)
33. Zhou, J., Qu, W., Zhang, F.: Chinese named entity recognition via joint identification and categorization. Chinese Journal of Electronics 22(2), 225–230 (2013)