Exploring Recurrent Neural Networks to Detect Named Entities from Biomedical Text

Lishuang Li, Liuke Jin and Degen Huang

School of Computer Science and Technology,
Dalian University of Technology, Dalian 116024, Liaoning, China
{lils, Huangdg}@dlut.edu.cn, dllg_lkjin@mail.dlut.edu.cn

Abstract. Biomedical named entity recognition (bio-NER) is a crucial and basic step in many biomedical information extraction tasks. However, traditional NER systems are mainly based on complex hand-designed features which are derived from various linguistic analyses and maybe only adapted to specified area. In this paper, we construct Recurrent Neural Network to identify entity names with word embeddings input rather than hand-designed features. Our contributions mainly include three aspects: 1) we adapt a deep learning architecture Recurrent Neural Network (RNN) to entity names recognition; 2) based on the original RNNs such as Elman-type and Jordan-type model, an improved RNN model is proposed; 3) considering that both past and future dependencies are important information, we combine bidirectional recurrent neural networks based on information entropy at the top layer. The experiments conducted on the BioCreative II GM data set demonstrate RNN models outperform CRF and deep neural networks (DNN), furthermore, the improved RNN model performs better than two original RNN models and the combined method is effective.

Keywords: bio-NER; Recurrent Neural Network; word embeddings; bidirectional; information entropy

1 Introduction

Entity mention extraction is an important technology to provide structured information from raw text. In biomedical information extraction, named entities are defined as the names of existing objects, such as protein, genes, drugs and etc. Identifying these entities from explosive incremental biomedical texts can provide great convenience for medical researchers. Hence the biomedical named entity recognition is fundamental for a wide variety of natural language processing application in the biomedical field, such as coreference resolution, relation extraction and etc [1].

Traditional research methods for bio-NER can be mainly classified into three categories which are dictionary-based methods, rule-based methods and statistical machine learning methods [2]. Because of the irregularities and ambiguities in bio-entities nomenclature and the difficulty in constructing proper and complete rule sets, the statistical machine learning method has become a better choice. In the camp of machine learning which mainly depends on a rich set of hand-designed features, many learning models, such as Support Vector Machine (SVM) [3], Maximum Entropy (ME) [4], Hidden Markov Model (HMM) [5] and Conditional Random Fields (CRF) [6], have been adopted in the NER task. However, their performance and promotion may be affected by some common drawbacks as followings:

- with the change of corpora and languages, the process to reconstruct feature set is difficult;
- some complex features with syntactic information rely on the performance of other NLP modules;
- the features with expert knowledge are expensive to acquire.

As the shallow machine learning methods described above have strong dependency on the artificial features and are hard to represent the complex models, deep learning has been applied on NER in recent years. Collobert et al. [7] proposed unified neural network architecture and learning algorithm for various natural languages processing tasks which also achieved a better result in the NER task. Chen et al. [8] adopted deep belief network (DBN) to extract entity mentions in Chinese documents which outperformed the traditional machine learning approach. Long short-term memory was applied for named entity recognition which had complex system architecture [9].

However, at the highest level of complexity for named entity recognition, one has to take into account the concept dependencies beyond the words surrounding the word of interest. To capture dependencies beyond the input window, Recurrent neural network (RNN) architecture can exploit the time-connection feedback. RNN is a neural network model developed under the deep learning architecture [10]. Different from other feed-forward neural network language models, RNN can maintain historical information with recurrent connections, which give RNN a potential to model long span dependencies [11]. Besides, RNNs have recently been shown to produce state-of-the-art results in the spoken language understanding (SUL). For example, Mesnil et al. [12] implemented and compared two important recurrent neural network architectures, the Elman-type network [13] which considered previous hidden layer and Jordan-type network [14] which considered previous output layer. In order to improved the performance on the Elman-type network, a context real-value input vector was provided in association with each word [15].

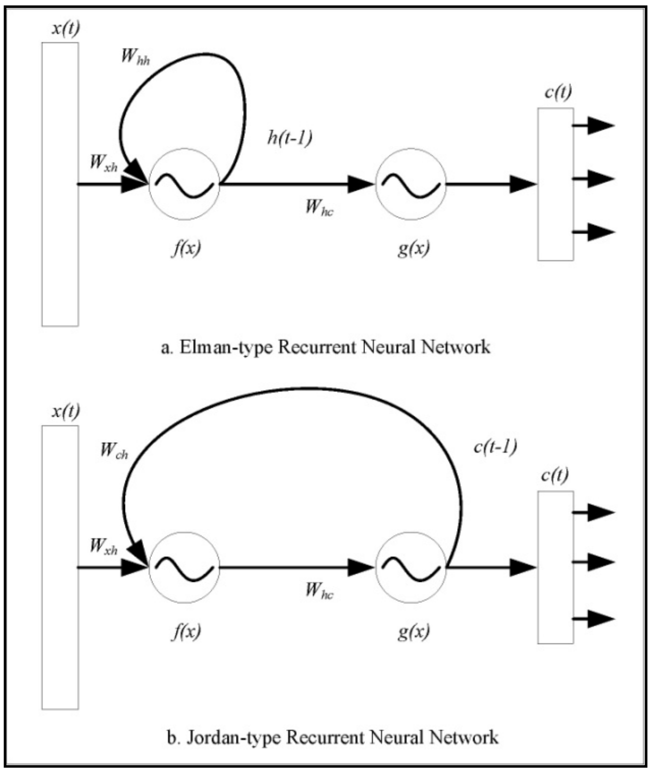
Leveraging RNN's recurrent property, in this paper we explore different RNN architectures and improve them adapted to named entity recognition. Firstly, we apply two simple types of RNN models Elman-type and Jordan-type networks on bio-NER. Then we integrate their strengths and propose a new RNN architecture. Finally, we combine the bidirectional recurrent neural networks based on information entropy. Experimental results show that the RNN architecture can achieve better performance than CRF and DNN with the same input.

The rest of this paper is organized as follows: Section 2 introduces recurrent neural network models. Application of improved RNN on bio-NER is described in Section 3. Section 4 shows the experiments and gives the comparison and analysis. Finally, conclusions are given in Section 5.

2 Recurrent neural network model

2.1 Two types of RNN architectures

Two variants of RNNs for modeling sequential prediction was described [12]: the Elman-type RNN and the Jordan-type RNN as shown in Fig. 1. The two models consist of an input layer, a hidden layer with recurrent connections that propagate time-delayed signals, and an output layer, plus the corresponding weight matrices.



x(t)
W_hh
h(t-1)
c(t)
W_xh
W_hc
f(x)
g(x)
a. Elman-type Recurrent Neural Network
x(t)
W_ch
c(t-1)
c(t)
W_xh
W_hc
f(x)
g(x)
b. Jordan-type Recurrent Neural Network

Fig. 1. Two original recurrent neural networks

In the Elman-type RNN, the output from the hidden layer at time *t*-1 is be kept and fed back to the hidden layer at time *t*, together with the raw input *x(t)* . At each time step, the input is propagated in a standard feed-forward neural network, and then stochastic gradient descent is applied to update the parameters, taking into account the influence of previous nodes through the recurrent connections. Thus previous information can be propagated to time *t* through the recurrent connections from time *t*-1. And the network can maintain and learn a sort of state summarizing past inputs, allowing it to perform sequence prediction. In Fig. 1, hidden layer of the Elman-type RNN can be represented mathematically with equation (1) and the output layer can be computed as equation (2).

$$\mathrm{h(t) = f(x(t) \cdot W_{xh} + h(t-1) \cdot W_{hh})} \tag{1}$$

$$\mathrm{c(t) = g(h(t) \cdot W_{hc})} \tag{2}$$

Besides, Jordan-type RNN is similar to Elman-type networks, except that the information of memory node is fed from the output layer instead of the hidden layer. The result of hidden layer can be obtained from the equation (3). W_{hc} is weight matrix connecting the result from output layer to the hidden layer. c(t-1) refers to values in the output layer from last node.

$$\mathrm{h(t) = f(x(t) \cdot W_{xh} + c(t-1) \cdot W_{ch})} \tag{3}$$

2.2 Improved recurrent neural network

Two simple RNN models described above exhibit two different ways using historical information, both of which only consider one layer as memory node and input their values into next hidden layer. In order to fully utilizing all the information saved in previous nodes, we give an improved RNN architecture which can not only maintain a copy of hidden layer but also record the probability of output layer. When computing hidden neurons, we take the same action with Elman-type RNN and use sigmoid function as the activation function of the hidden layer with equation (4).

$$\mathrm{f(z) = 1/1 + e^{-z}} \tag{4}$$

Different from Jordan-type RNN, the results of output layer from last node are inputted into the current output layer with equation (5). W_{cc} represents the parametric matrix connecting the output layers of two adjacent nodes. And at the output layer, we use a Softmax function to predict the probabilities, e.g. equation (6).

$$\mathrm{c(t) = g(g(t) \cdot W_{hc} + c(t-1) \cdot W_{cc})} \tag{5}$$

$$\mathrm{g(z_m) = e^{z_m} \Big/ \sum_k e^{z_k}} \tag{6}$$

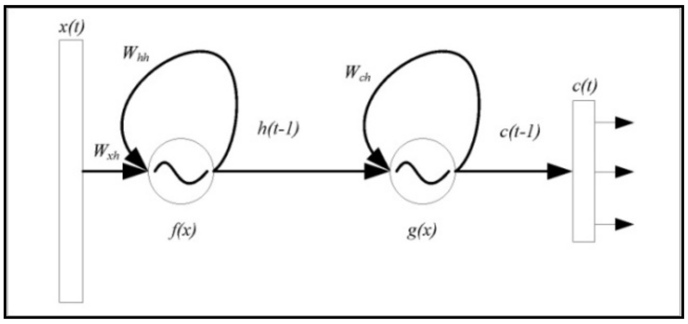


Fig. 2. Improved recurrent neural network

Fig. 2 shows our improved RNN model. For each step *t*, we take the *x(t)* and *h(t-1)* as inputs of hidden layer, then the result of calculations and *c(t-1)* are inputted into the output layer. Therefore, the hidden layer can maintain a representation of the sentence history; the output layer can maintain probabilities of previous labels which intuitively, would allow the model to perform a kind of disambiguation since no hard decision is made. Our RNN model blends the previous memories of hidden layer and output layer. Thus, the improved RNN not only receives different types of information representation from last node, but also retains consistency of information between layers.

2.3 Combined bidirectional recurrent neural networks

For sequence labeling, it is beneficial to have access to future as well as past context. However, since regular RNNs process sequences in temporal order, they ignore future context [16]. To use all available input information, it is possible to use two separate networks (one for each time direction) and then somehow merge the results[17]. Therefore we build two different RNNs respectively, namely ordered model (predicting from the past to the future) and reversed model (predicting from the future to the past). Then two unidirectional recurrent neural network models are combined to bidirectional neural networks based on information entropy at the top layer.

According to Shannon's entropy, the entropy represents the uncertainty of information. The less the entropy of the model is, the more certain the prediction is. For each word, the two models with different directions are firstly used to output the probabilities of different labels. Then the information entropy can be computed as (7). We compute the information entropy respectively ($H(p)_{ordered}$ and $H(p)_{reversed}$) based on the probability distribution at the output layer, and select the label predicted by the model which has a lower value as the correct label.

$$\mathrm{H(p)} = -\sum \mathrm{p(y)} \log \mathrm{p(y)} \tag{7}$$

3 Named entity recognition based on RNN

The system architecture for named entity recognition can be summarized in Fig. 3. It accepts an input sentence and gives a label y for each word in the sentence. The first layer obtains word embeddings by a lookup table operation. Then the context information can be extracted by concatenating all the words in the context window. Thus the representation x of input layer is obtained. Finally each word vector x is inputted into the improved RNN and the output layer gives the label.

3.1 Word embeddings

Word embeddings, also known as distributed representation, has recently been proposed to address several NLP problems and has achieved great success. A word expressed by the distributed representation is a dense, low-dimensional and real-valued vector, hopefully capturing the syntactic and semantic information in each dimension

[18]. The word vectors can capture many linguistic regularities, which is superior to one-hot representation, e.g. the biomedical term "*gene*", "*proteins*", "*kinase*" are close to each other, and far from the other type of nouns "weeks", "months", while for one-hot encoding, all the words are equally distant.

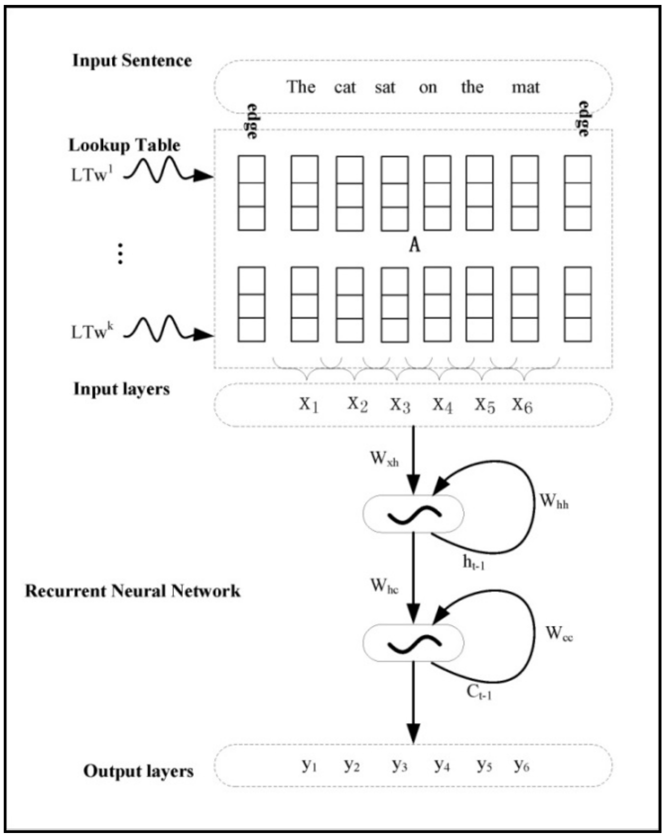


Fig. 3. Named entity recognition system with improved recurrent neural network language model

In our experiments, we use the distributed representations of words trained by Word2Vec tool developed by Mikolov [19], rather than hand-designed features. A real-valued embedding vector is associated with a word, and all the real-valued vectors are obtained by training a skip-gram model from 5.6GB unlabeled text downloaded from PubMed. In our experiments, the vector dimension is set to 200. After that, we can map each word in our corpora to an embedding and initialize the word lookup table with the embeddings. For all the words appearing in the corpus but not in the lookup table, we use "*unk*" as their unified representation, and give them a customized vector with the same dimension.

Given a sentence and a lookup table as shown in Figure 3, we construct a matrix A in which we denote $[A]_{i,j}$ the coefficient at row i and column j. And in order to capture short-term temporal dependencies in a sentence, we use a word-context window. With each word mapped to an embedding vector, the word-context window is the ordered concatenation of word embedding vectors. We concatenate the d_{win} column vectors around the i^{th} column vector in the matrix A to represent x_i:

$$([A]_{1,i-d_{win}/2} \cdots [A]_{k,i-d_{win}/2} \cdots [A]_{1,i+d_{win}/2} \cdots [A]_{k,i+d_{win}/2}) \tag{8}$$

In experiments, d_{win} is set to 5. Besides, to process the sentence boundaries, namely the first word and last word in the sentence, we set a customized vector with same dimension to represent the boundary vectors.

3.2 Chunk representation

Entity mention extraction can be considered as a sequential token tagging task as in [20]. They demonstrated that choice of encoding scheme had a big impact on the system performance and the less used BILOU formalism significantly outperformed the widely adopted BIO tagging scheme. Therefore BILOU tagging scheme is selected to find the entity boundary in our experiment. **B** refers to the beginning word of a gene name, **I** and **L** respectively indicate inside tokens and the last token in a gene name if it contains more than one word, **O** refers to the words which are not included in a gene name, and finally **U** represents the unit-length chunks.

3.3 Named entity recognition using RNN

In Fig. 4, we present the process about how to train ith word in the sentence. Before training, we first randomly initialize two parameters, one is in the hidden layer (e.g. h_0), and the other is in the output layer (e.g. c_0). The two parameters are constantly updated during the training. Then, with raw inputs $x(i\text{-}T\text{+}1)$, h_0 and c_0, we calculate the neurons in the hidden layer and the probabilities in output layer about the first word in the sliding window consisting of T words. The outputs of hidden layer and output layer are saved and inputted into the next node. We repeat the process till the last word (e.g. i^{th} word) is computed. Finally, the label of i^{th} word can be predicated.

In the Figure 4, the process fragment with the dotted box is described in more detail in the Figure 2. We can compute the hidden layer and output layer by (9) and (11), with parameters θ and λ. Any feed-forward recurrent neural network with T words in hidden layer and output layer can be seen as a composition of function $\mathrm{f}_\theta^{\mathrm{t}}(\cdot)$ and $\mathrm{g}_\lambda^{\mathrm{t}}(\cdot)$, corresponding to the i^{th} word:

$$\mathrm{f}_\theta(\cdot) = \mathrm{f}_\theta^{\mathrm{T}}(\mathrm{f}_\theta^{\mathrm{T}-1}(\cdots \mathrm{f}_\theta^{1}(\cdot)\cdots)) \tag{9}$$

$$f_\theta^1 = f(x(i-T+1) \cdot W_{xh} + h_0 \cdot W_{hh}) \tag{10}$$

$$\mathrm{g}_\lambda(\cdot) = \mathrm{g}_\lambda^{\mathrm{T}}(\mathrm{g}_\lambda^{\mathrm{T}-1}(\cdots \mathrm{g}_\lambda^{1}(\cdot)\cdots)) \tag{11}$$

$$g_{\lambda}^{1} = g(h(i - T + 1) \cdot W_{hc} + c_0 \cdot W_{cc}) \tag{12}$$

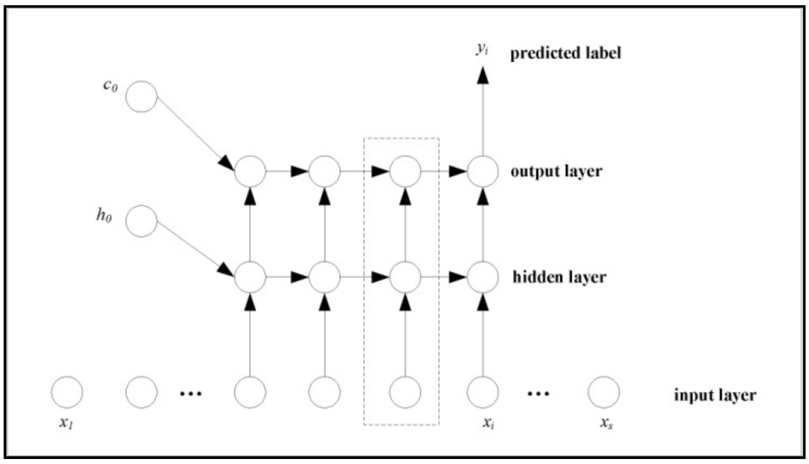


Fig. 4. The process to train the i^{th} word in the sentence

4 Experiments and results

Our experiments are carried out on the BioCreative II GM corpus which consists of 15,000 training sentences and 5,000 testing sentences. Bio-NERs in those sentences were manually annotated. Note that boundaries of gene names in text can be fuzzy even for human annotators and therefore alternative annotations were also provided with the corpus. In the experiment, we think an entity is correctly recognized if it matches the correct annotations or alternative annotations.

The results from different models will be shown in this section. All of them apply no post-processing so that the impact of post-processing is excluded. We measure the precision, recall and F-score of the improved RNN and compare with CRF baseline, deep neural network (DNN) and original RNNs. In experiments, all the deep networks are based on the common Theano neural network toolkit[1] and all the RNN models are trained with the same hyper-parameters. Besides, during the process, we set word-context window to 5, and fix the sliding window T to 5. To keep the consistent depth with RNN, we also set DNN to one hidden layer.

4.1 Evaluation methodology

We use F-score as our assessing criteria to evaluate our method. The definition of Precision (P), Recall (R) and F-score (F) are shown as (13, 14, 15). TP is short for true positives, FP represents false positives, and FN stands for false negatives.

[1] http://deeplearning.net/tutorial/rnnslu.html

$$P = TP/(TP + FP) \tag{13}$$

$$R = TP/(TP + FN) \tag{14}$$

$$F - score = 2 * P * R/(P + R) \tag{15}$$

4.2 Results and analysis

Results with different models.
We experiment using three different types of models, namely CRF, DNN and RNN. From Table 1, the results show that both CRF and RNN have better performance than DNN. And all the RNN models can achieve significant improvements by about 10% F-scores than DNN. It is shown that maintained historical information in RNN can play an important role in the sequence prediction. Furthermore, compared with CRF, the RNN models can also obtain higher F-scores by over 5%. The results demonstrate that RNN models have a distinct advantage in named entity recognition.

Table 1. Comparison among different models.

Model	P (%)	R (%)	F (%)
CRF(baseline)	79.79	69.65	74.38
DNN	75.00	63.78	68.93
Jordan	82.38	76.54	79.35
Elman	82.23	76.40	79.21
Ordered improved RNN	82.06	78.56	**80.27**
Reversed improved RNN	80.36	81.59	**80.97**
Combined *bidirectional* RNN	82.61	81.21	**81.91**

Compared with two original RNN models, the ordered improved RNN can obtain higher F-scores than Jordan-type by 0.92%, than Elman-type RNN by 1.06%, and the reversed improved RNN achieves 80.97% F-score, which is higher than Jordan by 1.62% and higher than Elman by 1.76%. The results show that both improved RNN models outperform original RNNs. Finally, the combined bidirectional improved RNN achieves the best performance of 81.91% F-score which is higher than the ordered improved RNN by 1.64% and higher than the reversed improved RNN by 0.94%. It is obviously shown that the proposed combined method is effective.

Comparison between the improved and two original models.
Fig. 5 shows the compared results among three kinds of RNNs with ordered model and reversed model respectively. We can see that even though both Jordan-type and Elman-type RNNs have similar shifting trend within specified number of iterations in both models, the improved RNN can always achieve better performance. This mainly because that the improved model not only takes into account the history messages in the hidden layer, but also considers the probabilities in the output layer.

Comparison between bidirectional and unidirectional models.
Fig. 6 describes the comparison between unidirectional and combined bidirectional models of our improved RNN. It is can be observed that the combined bidirectional model can obtain better performance. The main reason for the significant improvement lies in the combined approach which can fully weigh bidirectional historical information by computing the information entropy and choose the appropriate model as the temporary decisive model.

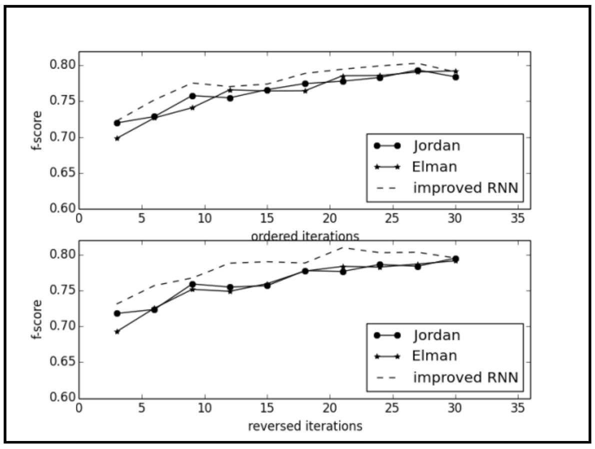


Fig. 5. Results of different RNNs with ordered and reversed models

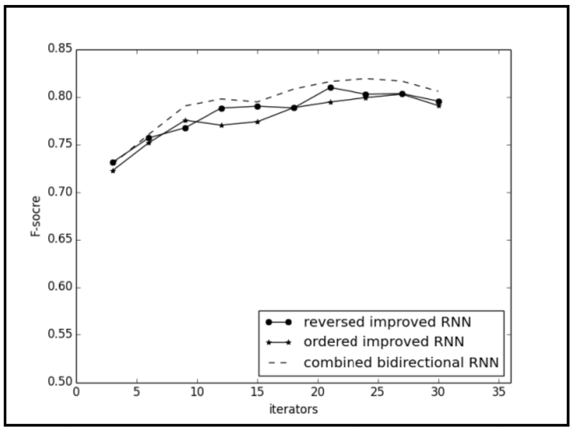


Fig. 6. Results of improved RNN with different iterations

More interestingly, we also observe that the RNNs' reversed model performs very well while its ordered model gives worse results, even though mathematically these two kinds of model are symmetric to each other. According to the similar analysis of [12], this maybe mainly because most of the named entities to be predicted in the BioCreative II are located in the first half of sentences, which makes the ordered model perform prediction with very little historical information.

5 Conclusion

In this paper, we apply different RNN models on the named entity recognition with word embeddings, and achieve better performance than CRF and DNN. Then, based on two simple original RNNs, we propose an improved RNN model which can maintain a copy of previous information at hidden layer and output layer. The experimental results show that our model is effective in NER task. Simultaneously, considering the context information from past and future has an important impact on the prediction, we present combined bidirectional recurrent neural networks based on information entropy whose performance is much higher than those of both unidirectional models.

Acknowledgment. The authors gratefully acknowledge the financial support provided by the National Natural Science Foundation of China under No. 61173101, 61173100.

References

1. Chen Y., Ouyang Y., Li W., Zheng D., Zhao T.: Using Deep Belief Nets for Chinese Named Entity Categorization. In: Proceedings of the 2010 Named Entities Workshop. 102-109 (2010).
2. Li L., Fan W., Huang D., Dang Y., Sun J.: Boosting performance of gene mention tagging system by hybrid methods. Journal of biomedical informatics,Volume 45. 156-164(2012).
3. Lee K.J., Hwang Y.S., Kim S., Rim H.C.: Biomedical named entity recognition using two-phase model based on SVMs. Journal of Biomedical Informatics. Volume 37. 436-447(2004).
4. Saha S.K., Sarkar S., Mitra P.: Feature selection techniques for maximum entropy based biomedical named entity recognition. Journal of biomedical informatics. Volume 45. 2673–2681(2009).
5. Shen D., Zhang J., Zhou G., Su J., Tan C.L.: Effective Adaptation of a Hidden Markov Model-based Named Entity Recognizer for Biomedical Domain. In: Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine. Volume 13. 49-56(2003).
6. Sun C., Guan Y., Wang X., Lin L.: Rich features based Conditional Random Fields for biological named entities recognition. *Computers in Biology and Medicine*. Volume 37. 1327-1333(2007).
7. Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K., Kuksa P.: Natural Language Processing (Almost) from Scratch. The Journal of Machine Learning Research. Volume 12. 2493-2537(2011).
8. Chen Y., Zheng D., Zhao T.: Exploring Deep Belief Nets to Detect and Categorize Chinese Entities. In: Advanced Data Mining and Applications. pp. 468-480. Springer, Berlin Heidelberg(2013)
9. Hammerton J.: Named Entity Recognition with Long Short-Term Memory. In: Proceedings of the seventh conference on Natural language learning *at* HLT-NAACL 2003. Volume 4, 172-175(2003).

10. Mikolov T., Karafiat M., Burget L., Cernoky J., Khudanpur S.: Recurrent neural network based language model. In: INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan. 1045-1048(2010).
11. Mikolov T., Kombrink S., Burget L., Cernocky J., Khudanpur S.: Extensions of recurrent neural network language model. In: Acoustics, Speech and Signal Processing (ICASSP). 5528-5531(2011).
12. Mesnil G., He X., Deng., Bengio Y.: Investigation of Recurrent Neural Network Architectures and Learning Methods for Spoken Language Understanding. In: INTERSPEECH. 3771-3775(2013).
13. Elman J.L.: Finding Structure in Time. Cognitive Science. Volume 14. 179-211(1990).
14. Jordan M.I.: Serial order: A parallel distributed processing approach. Advances in psychology, Volume 121. 471-495(1997).
15. Mikolov T., Zweig G.: Context Dependent Recurrent Neural Network Language Model. In: SLT. 234-239(2012).
16. Graves A.: Supervised Sequence Labelling with Recurrent Neural Networks. Vol. 385, Springer, Heidelberg(2012)
17. Schuster M., Paliwal K.K.: Bidirectional recurrent neural networks. Signal Processing, IEEE Transactions on. Volume 45. 2673-2681(1997).
18. Turian J., Ratinov L., Bengio Y.: Word representations : A simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics. 384-394 (2010).
19. Mikolov T., Chen K., Corrado G., Dean J.: Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv*:1301.3781(2013).
20. Ratinov L., Roth D.: Design Challenges and Misconceptions in Named Entity Recognition. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning. 147-155(2009).