Co-training an Improved Recurrent Neural Network with Probability Statistic Models for Named Entity Recognition

Yueqing Sun[1], Lin Li[1](✉), Zhongwei Xie[1], Qing Xie[1], Xin Li[2], and Guandong Xu[3]

[1] School of Computer Science and Technlogy, Wuhan University of Technology, Wuhan, China
{yqsuan,cathylilin,kevinsnest,felixxq}@whut.edu.cn
[2] iFLYTEK Big Data Research Institute, Hefei 230088, China
xinli2@iflytex.com
[3] School of Software, University of Technology Sydney, Ultimo 2007, Australia
Guandong.Xu@ust.edu.au

Abstract. Named Entity Recognition (NER) is a subtask of information extraction in Natural Language Processing (NLP) field and thus being wildly studied. Currently Recurrent Neural Network (RNN) has become a popular way to do NER task, but it needs a lot of train data. The lack of labeled train data is one of the hard problems and traditional co-training strategy is a way to alleviate it. In this paper, we consider this situation and focus on doing NER with co-training using RNN and two probability statistic models i.e. Hidden Markov Model (HMM) and Conditional Random Field (CRF). We proposed a modified RNN model by redefining its activation function. Compared to traditional sigmoid function, our new function avoids saturation to some degree and makes its output scope very close to [0, 1], thus improving recognition accuracy. Our experiments are conducted ATIS benchmark. First, supervised learning using those models are compared when using different train data size. The experimental results show that it is not necessary to use whole data, even small part of train data can also get good performance. Then, we compare the results of our modified RNN with original RNN. 0.5% improvement is obtained. Last, we compare the co-training results. HMM and CRF get higher improvement than RNN after co-training. Moreover, using our modified RNN in co-training, their performances are improved further. AQ1

Keywords: Named entity recognition · Co-training · Recurrent neural network · Probability statistic model · Natural language processing

This research project is supported by the National Social Science Foundation of China (Grant No:15BGL048), National Natural Science Foundation of China (Grant No:61602353, 61303029), 863 Program (2015AA015403), Hubei Province Science and Technology Support Project (2015BAA072).

1 Introduction

NER is a fundamental step in Natural Language Processing (NLP) which aims to identify boundaries and type of entities in text. In big data time, plenty of valuable information lies in disordered raw texts that cannot be directly used for many tasks. By doing NER we can know which category each word belongs. This technology is useful in information extraction (IE) field. Hence NER has been an essential task in several research teams, such as the Message Understanding Conferences (MUC), the Conferences on Natural Language Learning (CoNLL), etc. [1]. Also, in industry, google brain and baidu brain are very hot now and have been used in many specific applications. For example, in college entrance examination robot plan, a kind of brain-simulation program, NER is a vital subtask. However, there are some problems that should be noted:

1. The difficulty of feature-design. Most of NER researches commonly based on traditional machine learning methods. They often rely on the construction of complex hand-designed features which are derived from various linguistic analyses and maybe only adapted to specified area [2].
2. The lack of labeled data. Many NLP tasks base on big data and need large corpus especially labeled data, so is NER. But compared to the oceans of raw data that is produced every day, data with labels is in urgently lack.

For problem 1, a modified deep learning architecture named RNN is proposed and compared with two popular probability statistical models i.e. HMM and CRF. The two statistical models can learn statistical rules from a large number of training samples, so as to make predictions about the unknown. RNN belongs to deep learning which is a branch of machine learning and a development of neural network. It shakes off the requirement of hand-designed features and frees people from complex templates design. As described in [3], for tasks that involve sequential inputs, such as speech and language, it is often better to use RNN. In this paper, we modify the RNN activation function since selection of a good activation function is an important part to design a neural network. The experimental results prove that our modification gets a better achievement.

For problem 2, we utilize a co-training strategy, a kind of semi-supervised learning for the situation when train data is much less than test data. Co-training, originally proposed by A. Blum and T. Mitchell [4], is a popular strategy in semi-supervised learning. In this paper we cotrain RNN with the above two statistical models by selecting data with high confidence level to update the train set. Experimental results show that after co-training, all the models are improved.

The rest of this paper is organized as follows: Sect. 2 introduces the related works about NER researches. Section 3 describes our improved RNN and the co-training strategy. Experiments and result analysis are shown in Sect. 4. Finally, conclusion and future work are discussed in Sect. 5.

2 Related Work

As described in [5], there are three kind of methods for named entity recognition: dictionary-based methods, rule-based methods and statistical machine learning methods which rely on different theories. NER can be solved by machine learning methods, such as CRF [6,7], Support Vector Machine (SVM) [8], HMM [9] etc. These methods are commonly used for NER these years in a way of supervised learning. In addition, semi-supervised methods are also one road to this task when labeled data is difficulty to obtain.

Recently, while the probability statistical models perform well in many fields, deep neural networks as a new wave tide in machine learning, have achieved great performances in many domains such as image classification [10], knowledge discovery [11] and translation [12] etc. Collobert et al. [13] propose a unified neural network architecture and learning algorithm to do various NLP tasks and also achieved a better result for NER task. Compared to the well-known Convolutional Neural Network (CNN) which has achieved remarkable performances in image domain, RNN can exploit the time-connection feedback thus capture dependencies beyond the input window. Therefore, RNN architecture is more suitable for NER. Song et al. [14] build a simple and efficient system for bio-NER based on Recurrent Neural Network (RNN). Jason P.C. Chiu and Eric Nichols [15] present a novel neural network architecture that can automatically detect word and character level features using a hybrid bidirectional Long Short-Term Memory (LSTM) and CNN architecture.

On the other hand, as described in [16], a deep neural network is characterized by a set of weight matrices, bias vectors, and a nonlinear activation function, which gives a deep neural network the learning ability of hierarchical nonlinear mapping. But in model parameter training, weight matrices and bias vectors are updated using an error back-propagation algorithm whereas activation function is not. So the change of activation function is important for a neural network, which can speed up model training [17], enhance stability [18]. In this paper, we adopt the RNN model and modify its activation function to do NER task.

Another problem for RNN is that it needs plenty of train data. Hence in this paper we consider a co-training method which is one of useful solutions when train data is in lack. Co-training, one of the semi-supervised learning methods, was first proposed in 1998 and also has been used in NER. Tsendsuren et al. [19] present an Active Co-Training (ACT) algorithm for biomedical named-entity recognition. Li et al. [20] propose a semi-supervised approach to extract bilingual named entity and used a bilingual co-training algorithm to improve the named entity annotation quality. But using RNN to do co-training is a few in NER researches [21] and most of them are about biomedical domain. In this paper, we aim to explore the performance when co-training an improved RNN with probability statistic models for NER task.

3 Methodology

3.1 RNN

RNN has been applied in many fields and got great achievements in recent years. In this paper, we propose an improved RNN and use it to do co-training for the NER task. As one of the most successful and well-known neural network, CNN has made the remarkable achievements in multiple cross domains so it is valuable to try deep learning method for NER. Contrast to CNN, for text and language processing, RNN is proved to be good. It is a neural network model whose architecture can exploit the time-connection feedback [22]. Generally speaking, deep convolutional nets have brought breakthroughs in processing of image, video and audio etc., whereas recurrent nets have shone light on sequential data such as text and speech. A RNN and its unfolding structure in time of the computation involved in its forward computation is shown in Fig. 1. When unfolded, RNN can be regarded as a deep feedforward network and each layer shares same weights.

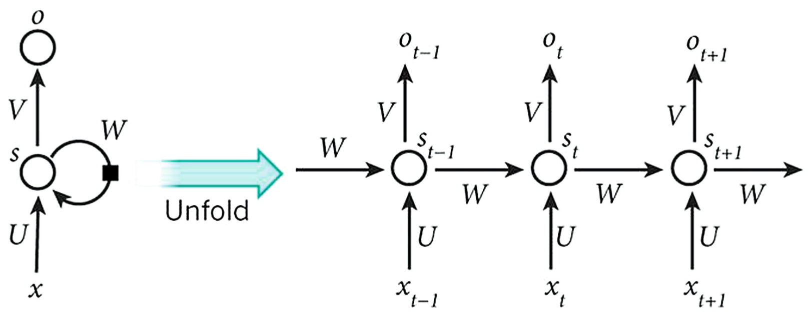

Fig. 1. A RNN and its unfold state

RNNs have many architectures and variants such as Elman-type and Jordan-type. Mesnil et al. [23] have implemented and compared the above two important RNN architectures to investigate spoken language understanding. Our RNN is based on the Elman-type described in [23] in this paper and we amend its activation function. Actually, as described in [16], many rectifier-type nonlinear functions have been proposed as activation functions, but the best nonlinear functions for any given task domain remain unknown. A same activation function performance may differ dramatically when applying it to different tasks. As for NER, compared to other well-known activation functions i.e. tanh, ReLu, PReLu, the sigmoid function, defined as Eq. 1, shows best in our experiments.

$$F(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

However sigmoid function has the saturation phenomenon (derivative tends to zero when the argument x approaches infinity) both at its left and right, which may make the training process harder. Still, sigmoid is most similar to the

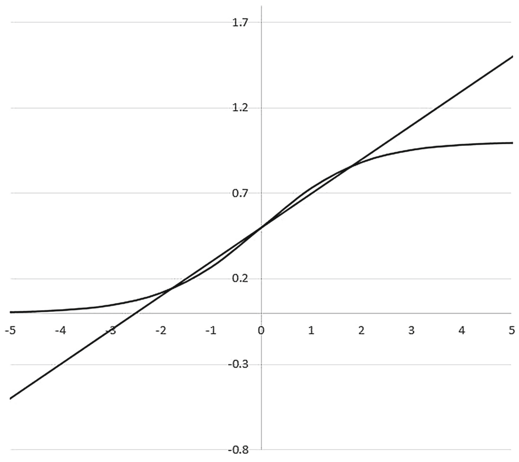

Fig. 2. Sigmoid and its linear approximation function

reflex mechanism of biological neuron and its output is always between 0 and 1, which can represent the label's prediction probability. The linear approximation function of sigmoid, expressed by Eq. 2, also performs well in our experiments. Figure 2 shows the two functions more intuitively.

$$L(x) = 0.2 \times x + 0.5 \tag{2}$$

Focusing on our NER task, we propose to combine the two above functions as our new activation function (shown in Eq. 3). Parameters a and b are coefficients which determined by experimental performances.

$$A(x) = a \times F(x) + b \times L(x) \tag{3}$$

The new activation function ameliorates the sigmoid's saturation phenomenon on the one hand and smooths the linear function on the other hand. Our co-training results using the new activation function are better than that using sigmoid or linear sigmoid function.

3.2 Co-training

In this paper, we modify activation function in deep neural network. On the other hand, we aim to explore the effect when co-training with RNN. As is known to all, co-training was proposed early years ago and wildly adopted in many tasks. To our best of knowledge, co-training using RNN is a few in NER [19,20]. Co-training is a kind of strategy in semi-supervised learning which fits for the situation when train data is limited. It uses two (or more) learners (model A and B). Wit the first same input as training data, according to different learning rules, learners produce labeled data respectively. Then the A's new labeled k data with highest confidence levels is selected and added into learner B's train set, vice versa. It will do this iteration until unlabeled data are all tagged. The co-training algorithm used in our paper is described as follows in Table 1.

Table 1. The co-training algorithm

```
Input:
A small set $T$ of original labeled samples.
A big set $U$ of unlabeled samples.
Test set $V$.
Classifiers $C_1$ and $C_2$ and their train set $s_1$ and $s_2$.
Number k selected data for each iteration.
----------------------------------------
Output:
best$C_1$, best$C_2$
----------------------------------------
Initialization:
$s_1$=$s_2$=$T$
k=300
$r_1$=$r_2$=0 //initialize the test results when testing V by classifiers
do:
    modelA=$C_1$.train($s_1$) //train classifiers with train set and save the model
    modelB=$C_2$.train($s_2$)
    temp$R_1$= modelA.test($V$)
    temp$R_2$= modelB.test($V$)
    if temp$R_1$ > $r_1$ or temp$R_2$ > $r_2$:
        $r_1$= temp$R_1$
        $r_2$= temp$R_2$
        best$C_1$=modelA
        best$C_2$=modelB
    newLabeledDataA=modelA.predict($U$) //tag the unlabeled data
    newLabeledDataB=modelB.predict($U$)
    newTrainA=newLabeledDataA.getTop(k)// select k new labeled data
    newTrainB=newLabeledDataB.getTop(k)
    $s_1$.add(newTrainA) //update train set
    $s_2$.add(newTrainB)
    $U$.remove(newTrainA)
    $U$.remove(newTrainB)
until $U$.isEmpty()
return best$C_1$, best$C_2$
```

4 Experiments

We conduct two sets of experiments. One is supervised learning and the other is semi-supervised learning with co-training, to make comparisons and explore the situation that using RNN to do co-training. The two experiments' train data sizes were totally different since semi-supervised learning works in the situation that training data is much less than testing data.

4.1 Dataset and Evaluation

Both the two sets of experiments are based on standard Airline Travel Information Systems (ATIS) benchmark [23] which contains 127 classes and uses the in/out/begin (IOB) representation. For example, a sentence can be expressed as in Table 2.

Table 2. Sentences in ATIS

sentence	find	flight	from	memphis	to	tacoma	dinner
label	O	O	O	B-fromloc.city_name	O	B-toloc.city_name	B-meal_description
sentence	cost	of	limousine	service	at	logan	airport
label	O	O	B-transport_type	O	O	B-toloc.airport_name	I-airport_name

All the results were evaluated by precision (P), recall (R) and F1-score, defined by Eqs. 4, 5 and 6 respectively.

$$P = \frac{TP}{TP + FP} \tag{4}$$

$$R = \frac{TP}{TP + FN} \tag{5}$$

$$F1 = \frac{2 \times P \times R}{P + R} \tag{6}$$

TP means true positives, FP means false positives and FN means false negatives.

4.2 Experiment 1: Supervised Learning

In this part we train four NER models as supervised learning by using different training data size. Based on ATIS original train/test proportion, i.e., train/test sets were 3983/893 sentences, we randomly select 20%, 40%, 60%, 80% and 100% of total train data (3983 sentences) as training set to train the HMM, CRF and RNN (including our modified RNN) models. The results through K-fold cross-validations are shown in Table 3. K is different based on the different size of training set. For example, when using 20%, K is five, and when using 40% and 60%, K is three. In Table 3 we can observe that how the three models perform when training on different data size. Generally speaking, the performances of all the models in NER are gradually improved when the training data size becomes larger. However, when the training data size goes larger and larger, the percentage of improvement becomes smaller. Actually, when the training data size increases from 20% to 40%, these models generally get a highest improvement. After that, increasing data only brings little benefit. This gives the reason that semi-supervised learning is feasible to achieve good results when train data is less. From Table 3, it shows that our modified RNN performs better than original RNN when train data is less than test data.

Table 3. P, R and F1 on different training data size

Training data size	20%	40%	60%	80%	100%
Models	P, R, F	P, R, F	P, R, F	P, R, F	P, R, F
HMM	0.6575, 0.6295, 0.6432	0.6776, 0.6588, 0.6680	0.6874, 0.6700, 0.6786	0.6914, 0.6750, 0.6831	0.6959, 0.6806, 0.6882
Improvements	-	**3.05%, 4.66%, 3.86%**	1.45%, 1.7%, 1.59%	0.58%, 0.75%, 0.66%	0.65%, 0.83%, 0.75%
CRF	0.9015, 0.7870, 0.8419	0.9252, 0.8543, 0.8884	0.9264, 0.8697, 0.8972	0.9295, 0.8863, 0.9074	0.9317, 0.8950, 0.9130
Improvements	-	**2.63%, 8.55%, 5.52%**	0.13%, 1.80%, 0.99%	0.33%, 1.91%, 1.14%	0.24%, 0.98%, 0.62%
RNN	0.8965, 0.8871, 0.8917	0.9350, 0.9311, 0.9333	0.9389, 0.9363, 0.9376	0.9475, 0.9415, 0.9445	0.9517, 0.9369, 0.9442
Improvements	-	**4.30%, 4.96%, 4.67%**	0.42%, 0.56%, 0.46%	0.92%, 0.56%, 0.74%	0.44%, −0.49%, −0.03%
Our Modified RNN	0.9030, 0.8992, 0.9011	0.9436, 0.9376, 0.9406	0.9380, 0.9383, 0.9381	0.9426, 0.9376, 0.9401	0.9527, 0.9450, 0.9489
Improvements	-	**4.50%, 4.27%, 4.38%**	−0.59%, 0.07%, −0.27%	0.49%, −0.07%, 0.21%	1.07%, 0.79%, 0.94%

4.3 Experiment 2: Semi-supervised Learning

To do semi-supervised learning, we assume that the labeled data are further less than the unlabeled. Thus here we reorganize the whole training data set (4876 sentences) and randomly select 1000 sentences about 20.5% as our new training set and the left as unlabeled data set to do co-training. In each iteration 300 high confidence level samples are picked from learner A and B, respectively. Those selected 300 samples are labeled by learner A or B and will be added into B or A as training set. Here we have done two group co-training: (A = HMM, B = RNN) and (A = CRF, B = RNN).

Co-training Using Original RNN with HMM and CRF. First we use the original RNN to do the above two group co-training. The before/after co-training performances are shown in Figs. 3 and 4. Since Precision and Recall show similar changing curve with F1 scores, we only report F1 scores in Figs. 3 and 4. After co-training, generally speaking, both HMM and CRF performs better and better with increasing iterations. But RNN need less iterations to achieve highest F1 scores. For example, in Fig. 3, when iterations go to 3, RNN shows best performance and its F1 score is 0.9129. We can say that RNN, a deep learning method, is good at NER and much better than traditional HMM and CRF. In addition, RNN helps them to obtain higher F1 scores by using RNN as a learner in co-training.

Through co-training, the recognition performances of two probability statistic models (here is HMM and CRF) are improved. For example, in Fig. 3 before co-training, the F1 score of HMM is 0.6236. After co-training with RNN, its F1 score rises to 0.6833 with 9.6% improvement. For CRF in Fig. 4, the largest

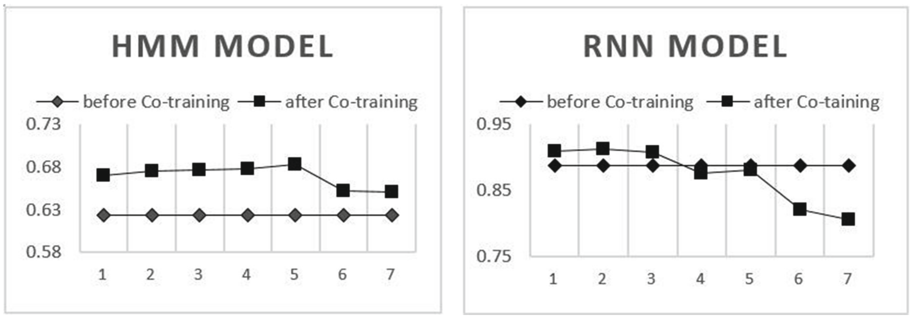

Fig. 3. Co-training results of HMM and RNN where X axis represents iteration times and Y axis represents F1 scores.

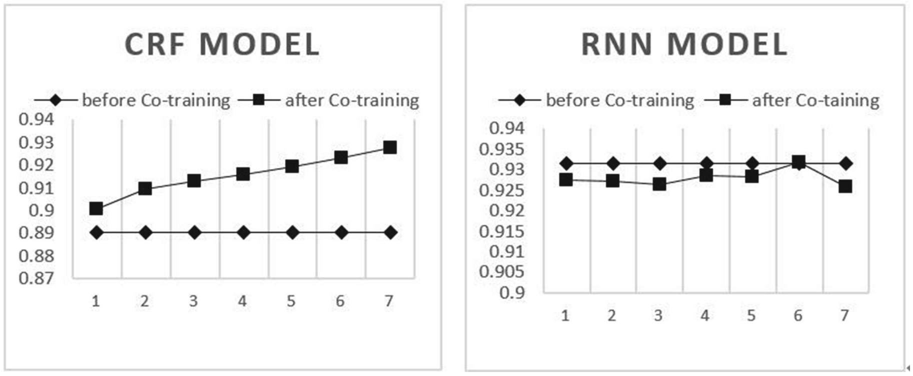

Fig. 4. Co-training results of CRF and RNN where X axis represents iteration times and Y axis represents F1 scores.

improvement is 4.14%. (0.8907 VS. 0.9276). In a word, by using co-training we can achieve better results with less training data, which give a solution when labeled data is in lack.

Co-training Using Improved RNN with HMM and CRF. In this part the improved RNN is used to redo the two group co-training. First we did several check experiments when setting a and b the different values according to their corresponding function curve trend and the (0.8, 0.2) pair is proved best in our experiments. Thus we set the new activation function coefficients a = 0.8 and b = 0.2 here. Compared to the results of co-training with original RNN, HMM and CRF get a little more improvement when co-training with improved RNN. The comparison results are reported in Table 4.

From Table 4, we can see that our improved RNN using modified activation function show better performance than HMM and CRF and the highest improvement is 5.4% compared with original RNN. Although HMM and CRF benefit a little from modified activation function, improved RNN obtains larger F1 score.

Table 4. Co-training using improved RNN with HMM and CRF

Model	HMM with RNN	HMM with Improved RNN
F1	0.6786	0.6811
Iteration	5	5
Improvement		0.4%
Model	RNN with HMM	Improved RNN with HMM
F1	0.8528	0.8987
Iteration	8	8
Improvement		5.4%
Model	CRF with RNN	CRF with Improved RNN
F1	0.8921	0.8966
Iteration	2	2
Improvement		0.5%
Model	RNN with CRF	Improved RNN with CRF
F1	0.8983	0.9008
Iteration	7	7
Improvement		0.3%

5 Conclusion and Future Work

In this paper, we consider the situation of less training data, study the influences that data sizes make on the model performance improvements. Moreover, we conduct the co-training experiments using original RNN and our modified RNN. The results of supervised learning indicate that even small train data size can get pretty good or even better achievement than that when data is bigger. The results of semi-supervised learning show that using RNN in co-training for NER task can achieve better performances when training data is less than testing data. In the future, it is worth to combine the co-training with RNN or other deep neural networks. In addition, we only change the activation function here and in the future, we are going to explore the RNN more deeply, for example improving its architecture to do NER or other related tasks.

References

1. Wahiba, B.A.K.: Named entity recognition using web document corpus. CoRR abs/1102.5728 (2011)
2. Lishuang, L., Liuke, J., Zhenchao, J., et al.: Biomedical named entity recognition based on extended Recurrent Neural Networks. In: BIBM, pp. 649–652 (2015)
3. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
4. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Eleventh Conference on Computational Learning Theory, pp. 92–100 (1998)

5. Li, L., Fan, W., Huang, D., et al.: Boosting performance of gene mention tagging system by hybrid methods. J. Biomed. Inform. **45**(1), 156–164 (2012)
6. Padmaja, S., Utpal, S., Jugal, K.: Named entity recognition in Assamese using CRFS and rules. In: IALP, pp. 15–18 (2014)
7. Tang, Z., Lingang, J., Yang, L., et al.: CRFs based parallel biomedical named entity recognition algorithm employing MapReduce framework. Cluster Comput. **18**(2), 493–505 (2015)
8. Ki-Joong, L., Young-Sook, H., Kim, S., et al.: Biomedical named entity recognition using two-phase model based on SVMs. J. Biomed. Inform. **37**(6), 436–447 (2004)
9. Gayen, V., Sarkar, K.: An HMM based named entity recognition system for indian languages: the JU system at ICON 2013. CoRR abs/1405.7397 (2014)
10. Sladojevic, S., Arsenovic, M., Anderia, A., et al.: Deep neural networks based recognition of plant diseases by leaf image classification. Comp. Int. Neurosc. **2016**(6), 1–11 (2016)
11. Janosek, M., Voln, E., Kotyrba, M.: Knowledge discovery in dynamic data using neural networks. Cluster Comput. **18**(4), 1411–1421 (2015)
12. Chollampatt, S., Kaveh, T., Hwee, T.N.: Neural network translation models for grammatical error correction. In: IJCAI, pp. 2768–2774 (2016)
13. Collobert, R., Weston, J., Bottou, L., et al.: Natural language processing (almost) from scratch. Mach. Learn. Res. **12**, 2493–2537 (2011)
14. Dingxin, S., Lishuang, L., Liuke, J., et al.: Biomedical named entity recognition based on recurrent neural networks with different extended methods. IJDMB **16**(1), 17–31 (2016)
15. Chiu, J.P.C., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. TACL **4**, 357–370 (2016)
16. Hoon, C., Sung, J.L., Jeon, G.P.: Deep neural network using trainable activation functions. In: IJCNN, pp. 348–352 (2016)
17. Anhao, X., Qingwei, Z., Yonghong, Y.: Speeding up deep neural networks in speech recognition with piecewise quantized sigmoidal activation function. IEICE Trans. **99-D**(10), 2558–2561 (2016)
18. Liew, S.S., Khalil-Hani, M., Bakhteri, R.: Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. Neurocomputing **216**, 718–734 (2016)
19. Tsendsuren, M., Meijing, L., Unil, Y., et al.: An active co-training algorithm for biomedical named-entity recognition. JIPS **8**(4), 575–588 (2012)
20. Li, Y., Huang, H., Zhao, X., Shi, S.: Named entity recognition based on bilingual co-training. In: Liu, P., Su, Q. (eds.) CLSW 2013. LNCS (LNAI), vol. 8229, pp. 480–489. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45185-0_50
21. Qikang, W., Tao, C., Ruifeng, X., et al.: Disease named entity recognition by combining conditional random fields and bidirectional recurrent neural networks. In: Database (2016)
22. Mikolov, T., Kara_t, M., Burget, L., et al.: Recurrent neural network based language model. In: INTERSPEECH, pp. 1045–1048 (2010)
23. Mesnil, G., He, X., Deng, L., et al.: Investigation of recurrent neural network architectures and learning methods for spoken language understanding. In: INTERSPEECH, pp. 3771–3775 (2013)