Chinese Journal of Electronics
Vol.29, No.3, May 2020

Biomedical Named Entity Recognition Based on Self-supervised Deep Belief Network

ZHANG Yajun[1], LIU Zongtian[2] and ZHOU Wen[2]

(1. *Shanghai Institute of Precision Measurement and Test, Shanghai 201109, China*)

(2. *School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China*)

Abstract — Named entity recognition is a fundamental and crucial issue of biomedical data mining. For effectively solving this issue, we propose a novel approach based on Deep belief network (DBN). We select nine entity features, and construct feature vector mapping tables by the recognition contribution degree of different values of them. Using the mapping tables, we transform words in biomedical texts to feature vectors. The DBN will identify entities by reducing dimensions of vector data. The extensive experimental results reveal that the novel approach has achieved excellent recognition performance, with 69.96% maximum value of *F*-measure on GENIA 3.02 testing corpus. We propose a self-supervised DBN, which can decide whether to add supervised fine-tuning or not according to the recognition performance of each layer, can overcome the errors propagation problem, while the complexity of model is limited. Test analysis shows that the new DBN improves recognition performance, the *F*-measure increases to 72.12%.

Key words — Biomedical named entity recognition, Feature selection, Feature vector mapping, Threshold judgement, Self-supervision.

I. Introduction

Biomedical named entity recognition (BNER) means to identify specified type entities from biomedical texts, such as proteins, genes, RNA *etc.*, which has wide application values, such as disease diagnosis, prevention and treatment[1−3]. BNER is also a challenging research field in the bioinformatics[4−6]. The main reasons are: new entities will continue to appear, then form a named entity open set; a same name may represent various category entities, we need analyze the context to correctly distinguish them; many entities have several different names. Moreover, there are also other issues, for example, some entity names are too long, compound words are too many, the proportion of abbreviation words is too large, many entity names are nested *etc.*

At present, many technologies have been developed for BNER in biomedical texts, which can be roughly divided into the following two categories: based on lexicons[7−9] and based on heuristic rules[10−12]. For lexicon-based methods, their recognition process depend entirely on lexicons, generally employing various matching methods to search strings in lexicons. Thus, the performance of lexicon-based methods mainly depend on matching algorithm and lexicon quality: Matching algorithm directly affects the similarity between lexicon information and text information, and the completeness of lexicon directly determines the precision and recall rate of gene-named entity recognition; the main drawbacks of this method are name conflict and limited coverage. Heuristic rule-based methods can generate rule templates artificially or heuristically by analyzing the internal and external features of named entities to realize the recognition of biological named entities. Therefore, researchers can flexibly formulate and extend rules to process various complex linguistic phenomena in biomedical literatures. However, the heuristic rule-based method also has drawbacks: 1) Rule creation generally takes a lot of time, and requires the participation of experts, which requires a lot of manpower; 2) Defined rules are usually employed only in a specific field, and it is generally difficult to apply to other fields, even if it can, it also needs a lot of changes.

In this study, we propose a novel BNER approach based on DBN. It does not require pre-defined rules or build a complete lexicon, so the recognition performance has little change when solving BNER issues in different fields. The approach selects recognition-features through statistical analysis, so it can ensure the objectivity and

Manuscript Received Oct. 8, 2018; Accepted Jan. 2, 2019. This work is supported by the National Natural Science Foundation of China (No.61305053, No.61273328, No.71203135).

 DOI:10.1049/cje.2020.03.001

accuracy of the system. The approach is based on deep learning, which can be used to deal with massive feature information with high dimension, so sample data capacity does not affect recognition performance. In addition, deep belief networks is a multi classifier, which can effectively complete multi classification tasks.

II. Study Model

1. Restricted Boltzmann machine

RBM is a generative stochastic neural network proposed by Hinton and Sejnowsk[13] and is illustrated in Fig.1. This network consists of some visible units and some hidden units. Visible or hidden variables are both binary variables whose state belongs to {0,1}. The entire network is a bipartite graph and an edge connection exists between only visible and hidden units.

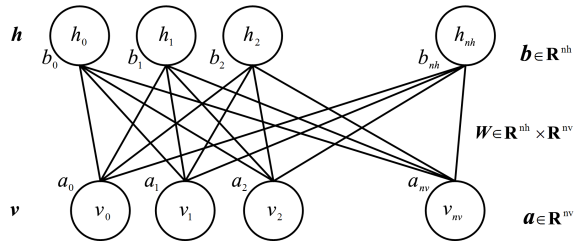

Fig. 1. RBM structure

In the Fig.1, $\boldsymbol{v}$ represents visible units, $\boldsymbol{h}$ represents hidden units, $\boldsymbol{W}$ represents the weight value between visible and hidden layers, and $\boldsymbol{a}$ and $\boldsymbol{b}$ represent their offset values, respectively. According to the known node value of a visible layer, we can obtain that of a hidden layer using the joint probability distribution equation as follows.

$$P(h_j{=}\,1|\boldsymbol{v}) = \frac{1}{1+\exp(-b_j-\sum\limits_{i=1}^{n_v} w_{j,i}v_i)} \tag{1}$$

According to Ref.[13], RBM is a symmetrical network. Therefore, we can obtain the node value of a visible layer from a known node value of a hidden layer using the equation as follows.

$$P(v_i{=}\,1|\boldsymbol{h}) = \frac{1}{1+\exp(-a_i-\sum\limits_{j=1}^{n_h} w_{j,i}h_j)} \tag{2}$$

In a traditional Markov chain approach, when solving for maximum joint probability $P(\boldsymbol{h}^{\infty}\boldsymbol{v}^{\infty})$ and initial joint distribution probability $p(\boldsymbol{v},\boldsymbol{h})$ guaranteeing the convergence rate is difficult. Moreover, the step ∞ is also difficult to calculate. Therefore, Hinton[14] proposed using Contrastive divergence (CD) guidelines, which can speed up calculation without compromising accuracy. The Kullback-Leibler distance is used to measure the "difference" of two probability distributions, expressed as $KL(P||P')$ as shown in the following equation.

$$CD_n = KL\left(P_0||P_\infty\right) - KL\left(P_n||P_\infty\right) \tag{3}$$

2. Self-supervised deep belief network

In the standard DBN network, which is consists of several RBM layers and a BP network (as shown in Fig.2(a)), each RBM layer training is unsupervised[15,16], however, it is finally reverse supervised fine-tuned using the BP network[17], which is a supervised classifier and employs Sigmod as the activation function. This model can improve the training efficiency, but which could cause errors propagation in the upper RBM layers and ultimately affect recognition performance. Therefore, we propose a novel DBN based on self supervision.

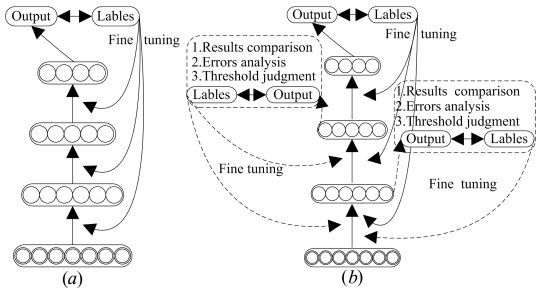

Fig. 2. Comparison of the two DBNs. (a)The standard DBN structure; (b)The Self-supervised DBN structure

In the Self-supervised DBN, RBM training results are estimated. Determining the application of BP training with supervision depends on the estimated results, as shown in Fig.2(b). Average error and its standard deviation are regarded as evaluation criteria. If one of them is greater than the given threshold value, the BP algorithm is used to fine-tune RBM network parameters in order to perform training with supervision. For example, a Self-supervised DBN has four RBM layers. After the third layer training, the analysis result shows that the average error is greater than or equal to the threshold, than the model will employ the BP algorithm to fine-tune all RBM layers that layer number is less than or equal to three. On the contrast, if the average error is less than the threshold, the model will not add the fine-tuning process. For the fine-tuning process can reduce errors propagation in upper RBM layer, thus the Self-supervised DBN improve recognition performance. Furthermore, the Self-supervised DBN does not require supervised training in all RBM layers. Therefore, it saves DBN network training time and improves recognition performance (shown in Algorithm 1).

Two evaluation parameters of DBN are applied based on self supervision: average error of one-time RBM training, as shown in Eq.(4), and unbiased standard deviation of errors of one-time RBM training, as shown in Eq.(5). We define the average value of total error

and average value of total error standard deviation as thresholds by training standard DBN several times.

$$\overline{e} = \frac{e_1 + e_2 + e_3 + \cdots + e_n}{n} = \frac{\sum_{i=1}^{n} e_i}{n} \tag{4}$$

$$S = \sqrt{\frac{\sum_{i=1}^{n} (e_i - \overline{e})^2}{n}} \tag{5}$$

Algorithm 1 Self-supervised DBN training process

Step 1 RBM layers set Rlevels, and initialize the current RBM training layer $CL = 1$, the input feature vector assigned to v^0.

Step 2 As the input vector v^0 RBM training algorithms, call the CD algorithm weight training and other layers of RBM.

Step 3 Calculating the training error e, if $e>=$ threshold value, using feature vector of the current RBM network output converted to values, assigning it to v^0. Calling BP algorithm, using BP network to fine-tune RBM layers.

Step 4 The use of features of the current RBM network (or Step 2) output converted to values, assigned to v^0.

Step 5 If the current RBM training layer $CL =$ training parameters RLevels, go to Step 5, otherwise $CL = CL + 1$, and go to Step 2.

Step 6 Using BP network to train a supervised classifier, and fine-tune the entire top-down reverse DBN network.

Step 7 The end of training.

Self-supervised DBN has some attractive features:

1) Referencing supervision way of human learning process, overcoming defects of existing supervision ways. This self-supervised learning way is more in line with human cognition law, easy to be analyzed and understood, has more strong theoretical basis.

2) Using uniform threshold monitoring can ensure that recognition ability of all RBM layers has a high consistency, so as to increase the stability of network recognition performance on the whole.

3) The recognition error of each layer can be suppressed to the maximum extent, so as to reduce the possibility of error layer by layer transfer.

4) With high recognition ability, compared with the recognition performance improvement that the time increase is limited, so the overall recognition performance is very good.

III. BNER Model Based on Self-supervised DBN

The framework of the BNER model based on Self-supervised DBN is shown as Fig.3. First, we analyze the existing biomedical data set (GENIA3.02), and select appropriate features. The rationality of feature selection will determine the recognition ability of the model. We need to analyze entity feature values and vectorize them for DBN testing and training, and apply mapping tables to match feature values and vector values. Finally getting feature vector sets of the training and testing corpus. A feature vector includes original words, entity feature vector values, an annotated label. DBN will train vector set with annotation labels to generate stable deep training network, so as to identify named entities in the testing corpus. In this study, two DBNs are used to test the same data set, and the testing results are compared and analyzed in detail.

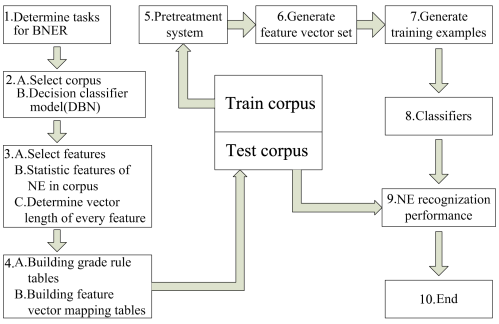

Fig. 3. The frame of BNER model based on DBN

1. Word structure feature (F1)

Complex word structure itself is a unique recognition feature, so we can recognize BNE by word structures. In this study, Word structure is built by 13 sub features, which are shown as Table 1.

Table 1. Sub features of word structure

ID	Feature description	Examples
S1	First letter uppercase	Activation
S2	All uppercase letters	DNA
S3	Combination of uppercase and lowercase letters	PMNs
S4	Alternating combination of letters and numbers	CDD11b
S5	Sequential combination of letters and numbers	gp120
S6	Roman numerals	III
S7	Including a hyphen	NF-kappa
S8	Begin with a hyphen	-141
S9	End with a hyphen	PKC
S10	Greek letter	alpha
S11	Including slashes	NGFI-B/nur77
S12	Left bracket	(
S13	Right bracket	)

Thirteen dimensions of structure feature vectors express thirteen sub features. Each vector dimension represents a sub feature. If results of a candidate word matching thirteen Regular Expressions correspond with several sub features, then the vector values of the word at corresponding dimensions are all 1.

2. Word structure feature (F2)

Keywords are high frequency words of BNEs. For getting the keyword feature vector values of candidate words, we need to construct a keyword vector mapping table. This procedure is as follows:

1) Statistic words labeled as BNEs in training corpus, and recording numbers of each word as CF in the entities and as TF in the training corpus.

2) Calculating weights of keywords' feature information, with the following formula: $Weight = CF/TF \times 100$.

3) Dividing grades according to weight values, the rules are shown as the following Table 2.

4) Building a keyword vector mapping table, which contains keys and values, each key is a keyword, each value is a vector value that is generated by the weight grade table. If a candidate word equals with the key in mapping table, we can get a vector value from the table.

Table 2. Weight grade table

Grade	Define	Feature vector
1	1≤Weight≤10	0 0 0 0 0 0 0 0 0 1
2	11≤Weight≤20	0 0 0 0 0 0 0 0 1 1
3	21≤ Weight≤30	0 0 0 0 0 0 0 1 1 1
4	31≤ Weight≤40	0 0 0 0 0 0 1 1 1 1
5	41≤ Weight≤50	0 0 0 0 0 1 1 1 1 1
6	51≤ Weight≤60	0 0 0 0 1 1 1 1 1 1
7	61≤ Weight≤70	0 0 0 1 1 1 1 1 1 1
8	71≤Weight≤80	0 0 1 1 1 1 1 1 1 1
9	81≤Weight≤90	0 1 1 1 1 1 1 1 1 1
10	91≤Weight≤100	1 1 1 1 1 1 1 1 1 1

3. Affix feature (F3)

In this paper, we use four characters placed before or after a word as affix features. Such as "lipoxygenase", the prefix of that is "lipo", suffix is "nase". For obtaining affix feature vectors, we need to build a prefix and a suffix vector mapping tables. The following is the prefix table construction steps. (The suffix table is created by the same process).

1) Selection first four characters of words that length are more than 5 in the training corpus to build a training corpus prefix list, statistics each prefix numbers as TF.

2) Selection first four characters of entities that length are more than 5 in the training corpus to build a entity prefix list, statistics each prefix numbers as CF.

3) Calculating weights of prefix features, with the following formula: $Weight = CF/TF \times 100$.

4) Dividing grades according to weight values, the rules are shown as the Table 2.

5) Building a prefix vector mapping table, which contains keys and values, each key is a prefix, each value is a vector value that is generated by the weight grade table. If the prefix of a candidate word equals with the key in mapping table, we can get a vector value from the table.

4. Morphology feature (F4)

The BNEs have high specificity, they usually have the same morphology. The general morphological features representation is an uppercase letter replacement for A, a lowercase letter replacement for a, a number replacement for 0, others replacement for X. We need to construct a morphology vector mapping table. The process is as follows:

1) Replacing all words in the training corpus by "AaOx" form, statistics each morphology corresponding to words numbers as TF in the training corpus, numbers as CF in named entities;

2) Calculating weights of morphology features, with the following formula: $Weight = CF/TF \times 100$.

3) Dividing grades according to weight values, the rules are shown as the Table 2.

4) Building a morphology vector mapping table, which contains keys and values, each key is a morphology, each value is a vector value that is generated by the weight grade table. If the morphology of a candidate word equals with the key in mapping table, we can get a vector value from the table.

5. Boundary word feature (F5)

Most BNEs are composed of multi-words, Using boundary information can improves the boundary recognition ability. For getting boundary word feature vectors, we need to construct boundary word vector mapping tables, the steps are following (We use the building process of left boundary table as an example).

1) Statistic first words of compound BNEs of training corpus, and recording numbers of them as CF in the entities and as TF in the training corpus.

2) Calculating weights of boundary word features, with the following formula: $Weight = CF/TF \times 100$.

3) Dividing grades according to weight values, the rules are shown as the Table 2.

4) Building a boundary word vector mapping table, which contains keys and values, each key is a boundary word, each value is a vector value that is generated by the weight grade table. If the left boundary of a candidate word equals with the key in mapping table, we can get the vector value from the table.

6. Unary word feature (F6)

There are many unary words composed by only one word in BNEs, such as IGF2, IL-2A *etc.* Unary word features can provide accurate and effective information for BNER. We select words that are only labeled as BNEs and never labeled as non entities from the training corpus to build an unary word vector mapping table. An unary word feature vector is ten binary dimensions. if a candidate word is an unary word, then its ten bit binary values of feature vector are all 1.

7. Nested feature (F7)

A compound word is composed by several root morphemes following certain rules. In this paper, entities with nested structure are all compound words. Such as "NF-kappaB element", a BNE, contains the entity "NF-kappaB", the nested structure increases the difficulty of identifying entity boundaries. This paper individually labels nested structures in entities, as a recognition feature for BNER, reducing named entity boundary recognition error rate. We select words that labeled as named entities and never labeled as non entities from the training corpus to build a set, then generates a nested word list by filtering the set to get words that can be concurrently labeled as unary and compound words. A nested word feature vector is ten binary dimensions. if a candidate word is a nested word, then its ten bit binary values of feature vector are all 1.

8. Common word feature (F8)

Common words can not reflect unique characteristics of named entities, have little significance for BNER, therefore, we can ignore these words. Currently there is not a English common word list in biomedical fields, we construct a common word list as following steps.

1) We obtain a common word list, which is called as C_word, using WordNet as etymology, statistic and abstraction most common words from four billion corpora on the Internet, including 6035 common words.

2) Searching words only labeled as non named entities without being labeled as named entities in the training corpus, building a training corpus common words list C_word_T;

3) Extracting common words in the C_word and C_word_T lists, finally obtaining common word list C_word_list;

4) A common word feature vector is ten binary dimensions. if a candidate word is a common word, then its ten bit binary values of feature vector are all 1.

9. Context features (F9)

Context information refers to two words before and behind a BNE, using the context information can improve the recognition ability of entity boundaries. For getting context feature vector values, we need to construct a context vector mapping table:

1) Statistic context words in the training corpus, recording each context word number as CF and total number in the training corpus as TF.

2) Calculating weights of context word features, with the following formula: $Weight = CF/TF \times 100$.

3) Dividing grades according to weight values, the rules are shown as the Table 2.

4) Building a context word vector mapping table, which contains keys and values, each key is a context word, each value is a vector value that is generated by the weight grade table. If the context word of a candidate word equals with the key in mapping table, we can get the vector value from the table.

IV. Test Results

Experiments have been conducted on GENIA corpus (v3.0p), which consists of 2000 MEDLINE abstracts annotated with Penn Treebank POS tags. We divided 2000 abstracts into 10 collections for 10-fold cross validation. The values of DBN-classifier parameters numepochs, batchsize, momentum, and alpha are set to 1, 100, 0, and 1, respectively. In our study, we applied recall rate (R), precision (P) and F-measure to evaluate recognition performance. Moreover, we used F incremental (FI), F incremental rate (FIR), F comparison incremental (FCI), and F comparison incremental rate ($FCIR$) to contrast two DBN-classifiers' performance.

1. Threshold analysis of self-supervised classifier

The mechanism of self-supervised classifier is that evaluating the mean and standard deviation of training errors in a RBM training period, if either is greater than a threshold, than applying the BP network to fine-tune the RBM layers to achieve the following: reduce the probability of errors being layer-wise propagation, and improve recognition performance. The rationality of the threshold selection impacts recognition performance and training time. If the threshold is lower, Fine-tuning during every RBM training period is possible, which greatly increases the training time. If the threshold is too higher, it may add the difficulty of eliminating those errors in the lower layer. In this study, we apply mean and standard deviation of errors as the threshold values after many times training the standard DBN. In Table 3, we increase the capacity of training set to acquire results. In these tests, the number of RBM layers is four, all feature layers are added, and $R1$ means the error in the first RBM layer training. Analysis of the data in Table 3 enables us to set the threshold to an average of 13 and the standard deviation to 3.

2. Recognition performance of standard classifier

We apply DBNi to complete two sets of experiments. I is the number of RBM layers contained in DBN. In the first experiment, we tested recognition performance with the same feature layers while the number of RBM layers increases. For the second experiment, we tested the performance when adding different feature layers with the same number of RBM layers.

Table 4 shows the results of the first experiment. When the number of RBM layers is four, the model get the best performance, the R, P, and F are 69.01%,

70.93% and 69.96%. But, when the number of RBM layers are greater than four, its performance declines faster and faster with each increase in the number of RBM layers. This reveals that the performance is not directly proportional to the number of RBM layers increasing. Therefore, in a practical application system, we should select a suitable depth of RBM based on massive experimental analysis.

Table 3. Threshold analysis of self-supervised classifier

Model+ Features	NUM	$R1$ (%)	$R2$ (%)	$R3$ (%)	$R4$ (%)	AVG (%)	SD (%)
DBN4+ ALL	220000	18.25	14.17	13.61	11.52	14.39	2.82
	240000	18.18	14.09	13.52	11.49	14.32	2.8
	260000	17.93	13.09	12.51	11.48	13.75	2.86
	280000	17.12	13.11	12.52	11.45	13.55	2.48
	300000	17.08	13.01	12.46	11.42	13.49	2.48
	320000	17.01	12.99	12.45	10.46	13.23	2.75
	340000	16.98	12.97	11.42	10.41	12.95	2.89
	360000	16.97	12.93	11.38	10.41	12.92	2.89
	380000	16.9	11.89	11.35	9.36	12.38	3.21
	400000	15.26	11.21	11.38	9.32	11.79	2.49
AVG	310000	17.17	12.95	12.26	10.73	13.28	2.77

Note: All represents all entity features, AVG is the average value, NUM is the number of vectors, SD is the standard deviation.

Table 4. Standard DBN recognition performance based on different feature layers

Model+Features	R	P	F	FI	FIR(%)
DBN1+All	65.15	67.23	66.17	0	0
DBN2+All	67.18	70.07	68.59	2.42	3.66
DBN3+All	67.21	70.56	68.84	0.25	0.36
DBN4+All	69.01	70.93	69.96	1.12	1.63
DBN5+All	67.21	70.53	68.83	−1.13	−1.62
DBN6+All	65.13	68.12	66.59	−2.24	−3.25

Note: F_n is the F-measure when the number of RBM layers is n, FI is difference between F_n and F_{n-1}, FIR is equal to $\frac{F_n - F_{n-1}}{F_{n-1}}$.

The second experiment mainly validates the effect of recognition performance when introducing different entity feature layers. Table 5 shows the results, which reveals that adding feature layers can considerably improve overall performance. With an increase in the number of feature layers, all three performance indicators show improvement. However, each feature layer has different contribution for improving recognition performance. From Table 5, keywords features contributes most to performance, the FI and FIR are 4.21% and 7.43%, whereas common word features contributes minimally, the FI and FIR are only 1.00% and 1.47%.

3. Performance comparison of two classifiers

A comparison of the standard and self-supervised DBN classifiers is shown in Table 6. The average increment of three indicators and the total absolute increment of the self supervised classifier is greater than that of the standard classifier. The highest increment rate is 3.65%. We can conclude that adding self-supervised learning can improve recognition performance to a certain extent.

Table 5. Standard DBN recognition performance based on different feature layers

Model+Features	R	P	F	FI	FIR(%)
DBN4+F1	52.11	62.05	56.65	0	0
DBN4+F[1-2]	57.83	64.23	60.86	4.21	7.43
DBN4+F[1-3]	57.93	66.82	62.06	1.2	1.97
DBN4+F[1-4]	59.53	67.19	63.13	1.07	1.72
DBN4+F[1-5]	60.17	69.08	64.32	1.19	1.88
DBN4+F[1-6]	63.11	69.88	66.32	2	3.11
DBN4+F[1-7]	65.34	70.58	67.86	1.54	2.32
DBN4+F[1-8]	66.83	71.02	68.86	1	1.47
DBN4+F[1-9]	69.01	70.93	69.96	1.1	1.6

Note: F_c is the F-measure when the number of feature layers is c, FI is difference between F_c and F_{c-1}, FIR is equal to $\frac{F_c - F_{c-1}}{F_{c-1}}$.

Table 6. Comparison of recognition effect between self-supervised and standard DBN based on different numbers of RBM layers

Model+ Features	R	P	F	FI	FIR (%)	FCI	$FCIR$ (%)
DBN1+All	65.15	67.23	66.17	0	0	0	0
DBN2+All	68.17	71.23	69.67	3.5	5.29	1.08	1.55
DBN3+All	69.11	73.96	71.45	1.78	2.55	2.61	3.65
DBN4+All	70.13	74.22	72.12	0.67	0.94	2.16	3
DBN5+All	68.85	72.88	70.81	−1.31	−1.82	1.98	2.8
DBN6+All	67.63	70.12	68.85	−1.96	−2.77	2.26	3.28

Note: F_s is the F-measure of the self-supervised classifier, F_u is the F-measure of the standard classifier, FCI is difference between F_s and F_u, $FCIR$ is equal to $\frac{F_s - F_u}{F_u}$.

Table 7 shows a comparison of the standard and self supervised classifiers. Experimental results show that adding feature layers can considerably improve overall performance for the two classifiers. Keywords features contribute the most whereas context features contribute minimally. The recognition performance of the self classifier is better than that of the standard classifier with the highest increment rate being 8.35%.

Table 7. Comparison of recognition effect between self-supervised and standard DBN based on different feature layers

Model+ Features	R	P	F	FI	FIR (%)	FCI	$FCIR$ (%)
DBN4+F1	57.11	67.35	61.81	0	0	5.16	8.35
DBN4+F[1-2]	62.31	69.47	65.7	3.89	6.29	4.84	7.37
DBN4+F[1-3]	65.03	70.21	67.52	1.82	2.77	5.46	8.09
DBN4+F[1-4]	66.31	70.92	68.54	1.02	1.51	5.41	7.89
DBN4+F[1-5]	67.22	71.61	69.35	0.81	1.18	5.03	7.25
DBN4+F[1-6]	68.02	72.36	70.12	0.77	1.11	3.8	5.42
DBN4+F[1-7]	69.32	73.02	71.12	1	1.43	3.26	4.58
DBN4+F[1-8]	69.55	73.34	71.39	0.27	0.38	2.53	3.54
DBN4+F[1-9]	70.13	74.22	72.12	0.73	1.02	2.16	3

Note: See the note to Table 6.

Table 8 shows the training time of the two DBN classifiers. Form the table, the training time of the self supervised DBN is longer, the reason is that adding fine-tuning process increases the training time. However, the maximum increase time is 30.45%, considering the improved performance, we can accept such a limited increase in time. The standard DBN classifier has a shorter training time. We chose the classifier's way of supervision according to specific situations and platform performance in actual applications. If training efficiency is emphasized, the standard DBN classifier is the better of the two classifiers. Self supervision is chosen because of its better recognition performance.

Table 8. Comparison of the training time of the two classifiers

Condition	Time of one period	U(ms)	S(ms)	IR(%)
A	Average	50.8	64.9	27.76
	Total	307	391	27.36
B	Average	29.3	38.1	30.03
	Total	266	347	30.45

Note: In condition A, all feature layer are the same, we increase the RBM layers from 1 to 6. In condition B, RBM layer is fixed, we increase the feature layers from 1 to 9. U is the standard classifier. S is the self-supervised classifier. IR refers to increment rate.

We compared the recognition performance of our approach to that of other teams[2,18−20], the result is shown as Table 9.

Table 9. Comparison with other teams

Teams	Approaches	Class	R	P	F
ZHAO[18]	HMM	Shallow machine learning	69.4	63	66
Settles[19]	CRF		69	70	69.5
LEE *et al.*[2]	SVM		71.8	69.8	70.8
Yao *et al.*[20]	DNN	Deep learning	76.13	66.54	71.01
Our result	DBN		69.01	70.93	69.96
Our result	SDBN		70.13	74.22	72.12

Note: DNN is the deep neutral network, SDBN is the self supervised DBN.

Comparison with the best shallow machine learning approach, the F-measure increase by 1.11%. Our approach has the following advantages:

1) The existing shallow machine learning approaches can achieve state-of-the-art performance on small-scale data, but with the increase of data size, the operation time and recognition ability will be greatly affected. However, deep learning can deal with the massive high-dimensional feature information, and thus the recognition performance will not be affected by the capacity of sample data.

2) Most of the existing machine learning approaches are typical binary classification, *e.g.* SVM. The SVM is difficulty in solving the multi-classification problem, although we can solve the problem by other methods, such as the SVM decision tree and constructing combination of multiple classifiers. However, the implementation process is very complex, and the accuracy of recognition is low and poor scalability. Self supervised DBN proposed in this paper is a multi classifier, which can easily complete multi classification tasks.

3) The existing machine learning approaches are more inclined to employ supervised training mode, this mode could improve the recognition performance, but the training time is longer. The self-supervised mode proposed in this paper could integrate advantages of unsupervised and supervised modes, which achieves performance improvement, but the growth of training time is limited.

V. Conclusions

In recent years, there are greatly improvement for BNER in corpora construction, feature selection and identification method, but word forms of BNEs are complex and varied, it faces great challenges to improve the performance in an actual application. Therefore, for a future study, we will research on the following aspects:

1) Extracting features that have more generalized ability. We could apply deeper information (such as syntactic knowledge and semantic knowledge), extract more nature and more generalization features (such as semantic features), so as to improve the identification ability of unknown named entities.

2) Fusing multiple classification approaches. Due to the approach based on rules need the participation of domain experts, the consumption of manpower and time, and lack of portability. For statistical approaches, though with a certain objectivity but dependent on large-scale corpus and the processing time is too long. And the mixed use of the two approaches will have better portability and shorter training time, is a more ideal way.

References

[1] T.h. Tsai, W.C. Chou, S.H. Wu, *et al.*, "Integrating linguistic knowledge into a conditional random field framework to identify biomedical named entities", *Expert Systems with Applications*, Vol.30, No.1, pp.117–128, 2006.

[2] K.J. Lee, Y.S. Hwang and H.C. Rim, "Two-phase biomedical named entity recognition based on svms", *Proc. of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, Sapporo, Hokkaido, Japan, pp.33–40, 2003.

[3] S.F. Altschul, T.L. Madden, A.A. Schaffer, *et al.*, "Gapped blast and psi-blast: A new generation of protein database search programs", *Nucleic Acids Research*, Vol.25, No.17, pp.3389–3402, 1997.

[4] K.J. Lee, Y.S. Hwang, S. Kim, *et al.*, "Named entity recognition using two-phase model based on svms", *Journal of Biomedical Informatics*, Vol.37, No.6, pp.436–447, 2004.

[5] Y. Song, E. Kim, G.G. Lee, *et al.*, "Posbiotmner: A trainable biomedical named-entity recognition system", *Bioinformatics*,

Vol.21, No.11, pp.2794–2796, 2005.
[6] R.T.H. Tsai, C.L. Sung, H.J. Dai, *et al.*, "Nerbio: Using selected word conjunctions,term normalization, and global patterns to improve biomedical named entity recognition", *BMC Bioinformatics*, Vol.7, No.5, pp.5–11, 2006.
[7] Y. Tsuruoka and J. Tsujii, "Boosting precision and recall of dictionary-based protein name recognition", *Proc. of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, Sapporo, Hokkaido, Japan, pp.41–48, 2003.
[8] Z. Yang, H. Lin and Y. Li, "Exploiting the performance of dictionary-based bio-entity name recognition in biomedical literature", *Computational Biology and Chemistry*, Vol.32, No.4, pp.287–291, 2008.
[9] M. J. Schuemie, B. Mons, M. Weeber, *et al.*, "Evaluation of techniques for increasing recall in a dictionary approach to gene and protein name identification", *Journal of Biomedical Informatics*, Vol.40, No.3, pp.316–324, 2007.
[10] K. Franzen, G. Eriksson, F. Olsson, *et al.*, "Protein names and how to find them", *International Journal of Medical Informatics*, Vol.67, No.1, pp.49–61, 2002.
[11] D. Hanisch, K. Fundel, H.T. Mevissen, *et al.*, "Rule-based protein and gene entity recognition", *BMC Bioinformatics*, Vol.6, No.1, pp.1–14, 2005.
[12] Y. Tsuruoka, J. McNaught and S. Ananiadou, "Normalizing biomedical terms by minimizing ambiguity and variability", *BMC Bioinformatics*, Vol.9, No.3, pp.2–3, 2008.
[13] G.E. Hinton and R.R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", *Science*, Vol.313, No.5786, pp.504–507, 2006.
[14] G.E. Hinton, "Training products of experts by minimizing contrastive divergence", *Neural Computation*, Vol.14, No.8, pp.1771–1880, 2002.
[15] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines", *Journal of Machine Learning Research*, Vol.5, No.2, pp.1967–2006, 2009.
[16] G.E. Hinton, S. Osindero and Y.W. Teh, "A fast learning algorithm for deep belief nets", *Neural Computation*, Vol.18, No.7, pp.1527–1554, 2006.
[17] C. Guoan, "Fast backpropagation learning using optimal learning rate and momentum", *Journal of Southeast University*, Vol.10, No.3, pp.517–527, 1999.
[18] S. Zhao, "Named entity recognition in biomedical texts using an hmm model", *Proc. of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, pp.84–87, 2004.
[19] B. Settles, "Biomedical named entity recognition using conditional random fields and rich feature sets", *Proc. of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, pp.104–107, 2004.
[20] L. Yao, H. Liu, Y. Liu, *et al.*, "Biomedical named entity recognition based on deep neutral network", *International Journal of Hybrid InformationTechnology*, Vol.8, No.8, pp.279–288, 2015.

ZHANG Yajun was born in 1985. He received the Ph.D. degree at Shanghai University in 2017. His research interests include deep learning, concept lattice and knowledge reasoning.
(Email: zyj1985email@163.com)

LIU Zongtian was born in 1946. He is a professor of Shanghai University. His research interests include program proof and program transformation, software quality management, data mining, rough sets and concept lattices.
(Email: ztliu@shu.edu.cn)

ZHOU Wen was born in 1979. She is an associate professor of Shanghai University. His research interests include machine learning, data mining, complex networks and natural language processing.
(Email: zhouwen@shu.edu.cn)