Using Deep Belief Nets for Chinese Named Entity Categorization

Yu Chen[1], You Ouyang[2], Wenjie Li[2], Dequan Zheng[1], Tiejun Zhao[1]
[1]School of Computer Science and Technology, Harbin Institute of Technology, China
{chenyu, dqzheng, tjzhao}@mtlab.hit.edu.cn
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
{csyouyang, cswjli}@comp.polyu.edu.hk

Abstract

Identifying named entities is essential in understanding plain texts. Moreover, the categories of the named entities are indicative of their roles in the texts. In this paper, we propose a novel approach, Deep Belief Nets (DBN), for the Chinese entity mention categorization problem. DBN has very strong representation power and it is able to elaborately self-train for discovering complicated feature combinations. The experiments conducted on the Automatic Context Extraction (ACE) 2004 data set demonstrate the effectiveness of DBN. It outperforms the state-of-the-art learning models such as SVM or BP neural network.

1 Introduction

Named entities (NE) are defined as the names of existing objects, such as persons, organizations and etc. Identifying NEs in plain texts provides structured information for semantic analysis. Hence the named entity recognition (NER) task is a fundamental task for a wide variety of natural language processing applications, such as question answering, information retrieval and etc. In a text, an entity may either be referred to by a common noun, a noun phrase, or a pronoun. Each reference of the entity is called a mention. NER indeed requires the systems to identify these entity mentions from plain texts. The task can be decomposed into two sub-tasks, i.e., the identification of the entities in the text and the classification of the entities into a set of pre-defined categories. In the study of this paper, we focus on the second sub-task and assume that the boundaries of all the entity mentions to be categorized are already correctly identified.

In early times, NER systems are mainly based on handcrafted rule-based approaches. Although rule-based approaches achieved reasonably good results, they have some obvious flaws. First, they require exhausted handcraft work to construct a proper and complete rule set, which partially expressing the meaning of entity. Moreover, once the interest of task is transferred to a different domain or language, rules have to be revised or even rewritten. The discovered rules are indeed heavily dependent on the task interests and the particular corpus. Finally, the manually-formatted rules are usually incomplete and their qualities are not guaranteed.

Recently, more attentions are switched to the applications of machine learning models with statistic information. In this camp, entity categorization is typically cast as a multi-class classification process, where the named entities are represented by feature vectors. Usually, the vectors are abstracted by some lexical and syntactic features instead of semantic feature. Many learning models, such as Support Vector Machine (SVM) and Neural Network (NN), are then used to classify the entities by their feature vectors.

Entity categorization in Chinese attracted less attention when compared to English or other western languages. This is mainly because the unique characteristics of Chinese. One of the most common problems is the lack of boundary information in Chinese texts. For this problem, character-based methods are reported to be a possible substitution of word-based methods. As to character-based methods, it is important to study the implicit combination of characters.

In our study, we explore the use of Deep Belief Net (DBN) in character-based entity categorization. DBN is a neural network model which is developed under the deep learning architecture. It is claimed to be able to automatically learn a deep hierarchy of the input features with increasing levels of abstraction for the complex problem. In our problem, DBN is used to automatically discover the complicated composite effects of the characters to the NE categories from the input data. With DBN, we need not to manually construct the character combination features for expressing the semantic relationship among characters in entities. Moreover, the deep structure of DBN enables the possibility of discovering very sophisticated combinations of the characters, which may even be hard to discover by human.

The rest of this paper is organized as follow. Section 2 reviews the related work on name entity categorization. Section 3 introduces the methodology of the proposed approach. Section 4 provides the experimental results. Finally, section 5 concludes the whole paper.

2 Related work

Over the past decades, NER has evolved from simple rule-based approaches to adapted self-training machine learning approaches.

As early rule-based approaches, MacDonald (1993) utilized local context, which implicate internal and external evidence, to aid on categorization. Wacholder (1997) employed an aggregation of classification method to capture internal rules. Both used hand-written rules and knowledge bases. Later, Collins (1999) adopted the AdaBoost algorithm to find a weighted combination of simple classifiers. They reported that the combination of simple classifiers can yield some powerful systems with much better performances. As a matter of fact, these methods all need manual studies on the construction of the rule set or the simple classifiers.

Machine learning models attract more attentions recently. Usually, they train classification models based on context features. Various lexical and syntactic features are considered, such as N-grams, Part-Of-Speech (POS), and etc. Zhou and Su (2002) integrated four different kinds of features, which convey different semantic information, for a classification model based on the Hidden Markov Model (HMM). Koen (2006) built a classifier with the Conditional Random Field (CRF) model to classify noun phrases in a text with the WordNet SynSet. Isozaki and Kazawa (2002) studied the use of SVM instead.

There were fewer studies in Chinese entity categorization. Guo and Jiang (2005) applied Robust Risk Minimization to classify the named entities. The features include seven traditional lexical features and two external-NE-hints based features. An important result they reported is that character-based features can be as good as word-based features since they avoid the Chinese word segmentation errors. In (Jing et al., 2003), it was further reported that pure character-based models can even outperform word-based models with character combination features.

Deep Belief Net is introduced in (Hinton et al., 2006). According to their definition, DBN is a deep neural network that consists of one or more Restricted Boltzmann Machine (RBM) layers and a Back Propagation (BP) layer. This multi-layer structure leads to a strong representation power of DBN. Moreover, DBN is quite efficient by using RBM to implement the middle layers, since RBM can be learned very quickly by the Contrastive Divergence (CD) approach. Therefore, we believe that DBN is very suitable for the character-level Chinese entity mention categorization approach. It can be used to solve the multi-class categorization problem with just simple binary features as the input.

3 Deep Belief Network for Chinese Entity Categorization

3.1 Problem Formalization

An Entity mention categorization is a process of classifying the entity mentions into different categories. In this paper, we assume that the entity mentions are already correctly detected from the texts. Moreover, an entity mention should belong to one and only one predefined category. Formally, the categorization function of the name entities is

$$f(V(e_i)) \rightarrow C \qquad (1)$$

where e_i is an entity mention from all the mention set *E*, $V(e_i)$ is the binary feature vector of e_i, $C=\{C_1, C_2, \ldots, C_M\}$ is the pre-defined categories. Now the question is to find a classification function $f: R^{|D|} \rightarrow C$ which maps the feature vector $V(e_i)$ of an entity mention to its category. Generally, this classification function is learned from training data consisting of entity mentions with labeled categories. The learned function is then used to predict the category of new entity mentions by their feature vectors.

3.2 Character-based Features

As mentioned in the introduction, we intend to use character-level features for the purpose of avoiding the impact of the Chinese word segmentation errors. Denote the character dictionary as $D=\{d_1, d_2, \ldots, d_N\}$. To an *e*, it's feature vector is $V(e)=\{ v_1, v_2, \ldots, v_N \}$. Each unit v_i can be valued as Equation 2.

$$v_i = \begin{cases} 1 & d_i \in e \\ 0 & d_i \notin e \end{cases} \qquad (2)$$

For example, there is an entity mention 克林顿 'Clinton'. So its feature vector is a vector with the same length as the character dictionary, in which all the dimensions are 0 except the three dimensions standing for 克, 林, and 顿. The representation is clearly illustrated in Figure 1 below. Since our objective is to test the effectiveness of DBN for this task. Therefore, we do not involve any other feature.

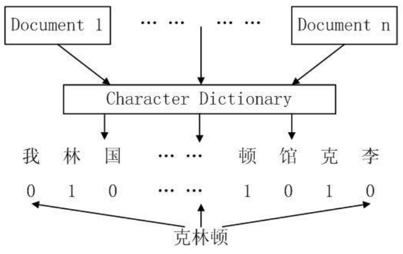


Fig. 1. Generating the character-level features

Characters compose the named entity and express its meaning. As a matter of fact, the composite effect of the characters to the mention category is quite complicated. For example, 老李 'Mr. Li' and 老挝 'Laos' both have character 老, but 老李 'Mr. Li' indicates a person but 老挝 'Laos' indicates a country. These are totally different NEs. Another example is 巴拉圭首都 'Capital of Paraguay' and 雅松森 'Asuncion'. They are two entity mentions point to the same entity despite that the two entities do not have any common characters. In such case, independent character features are not sufficient to determine the categories of the entity mentions. So we should also introduce some features which are able to represent the combinational effects of the characters. However, such kind of features is very hard to discover. Meanwhile, a complete set of combinations is nearly impossible to be found manually due to the exponential number of all the possible combinations. As in our study, we adopt DBN to automatically find the character combinations.

3.3 Deep Belief Nets

Deep Belief Network (DBN) is a complicated model which combines a set of simple models that are sequentially connected (Ackley, 1985). This deep architecture can be viewed as multiple layers. In DBN, upper layers are supposed to represent more "abstract" concepts that explain the input data whereas lower layers extract "low-level features" from the data. DBN often consists of many layers, including multiple Restricted Boltzmann Machine (RBM) layers and a Back Propagation (BP) layer.

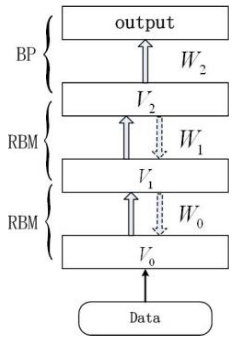


Fig. 2. The structure of a DBN.

As illustrated in Figure 2, when DBN receives a feature vector, the feature vector is processed from the bottom to the top through several RBM layers in order to get the weights in each RBM layer, maintaining as many features as possible when they are transferred to the next layer. RBM deals with feature vectors only and omits the label information. It is unsupervised. In addition, each RBM layer learns its parameters independently. This makes the parameters optimal for the relevant RBM layer but not optimal for the whole model. To solve this problem, there is a supervised BP layer on top of the model which fine-tunes the whole model in the learning process and generates the output in the inference process. After the processing of all these layers, the final feature vector consists of some sophisticated features, which reflect the structured information among the original features. With this new feature vector, the classification performance is better than directly using the original feature vector.

None of the RBM is capable of guaranteeing that all the information conveyed to the output is accurate or important enough. However the learned information produced by preceding RBM layer will be continuously refined through the next RBM layer to weaken the wrong or insignificant information in the input. Each layer can detect feature in the relevant spaces. Multiple layers help to detect more features in different spaces. Lower layers could support object detection by spotting low-level features indicative of object parts. Conversely, information about objects in the higher layers could resolve lower-level ambiguities. The units in the final layer share more information from the data. This increases the representation power of the whole model. It is certain that more layers mean more computation time.

DBN has some attractive features which make it very suitable for our problem.

1) The unsupervised process can detect the structures in the input and automatically obtain better feature vectors for classification.
2) The supervised BP layer can modify the whole network by back-propagation to improve both the feature vectors and the classification results.
3) The generative model makes it easy to interpret the distributed representations in the deep hidden layers.
4) This is a fast learning algorithm that can find a fairly good set of parameters quickly and can ensure the efficiency of DBN.

3.3.1 Restricted Boltzmann Machine (RBM)

In this section, we will introduce RBM, which is the core component of DBN. RBM is Boltzmann Machine with no connection within the same layer. An RBM is constructed with one visible layer and one hidden layer. Each visible unit in the visible layer V is an observed variable v_i while each hidden unit in the hidden layer H is a hidden variable h_j. Its joint distribution is

$$p(v,h) \propto \exp(-E(v,h)) = e^{h^T W v + b^T x + c^T h} \quad (3)$$

In RBM, the parameters that need to be estimated are $\theta = (W, b, c)$ and $(v,h) \in \{0,1\}^2$.

To learn RBM, the optimum parameters are obtained by maximizing the above probability on the training data (Hinton, 1999). However, the probability is indeed very difficult in practical calculation. A traditional way is to find the gradient between the initial parameters and the respect parameters. By modifying the previous parameters with the gradient, the expected parameters can gradually approximate the target parameters as

$$W^{(\tau+1)} = W^{(\tau)} + \eta \left. \frac{\partial P(v^0)}{\partial W} \right|_{W^\tau} \quad (4)$$

where η is a parameter controlling the leaning rate. It determines the speed of *W* converging to the target.

Traditionally, the Markov chain Monte Carlo method (MCMC) is used to calculate this kind of gradient.

$$\frac{\partial \log p(v,h)}{\partial w} = \langle h^0 v^0 \rangle - \langle h^\infty v^\infty \rangle \quad (5)$$

where $\log p(v,h)$ is the log probability of the data. $\langle h^0 v^0 \rangle$ denotes the multiplication of the average over the data states and its relevant sample in hidden unit. $\langle h^\infty v^\infty \rangle$ denotes the multiplication of the average over the model states in visible unit and its relevant sample in hidden unit.

However, MCMC requires estimating an exponential number of terms. Therefore, it typically takes a long time to converge to $\langle h^\infty v^\infty \rangle$. Hinton (2002) introduced an alternative algorithm, i.e., the contrastive divergence (CD) algorithm, as a substitution. It is reported that CD can train the model much more efficiently than MCMC. To estimate the distribution $p(x)$, CD considers a series of distributions { $p_n(x)$ } which indicate the distributions in *n* steps. It approximates the gap of two different Kullback-Leiler divergences (Kullback, 1987) as

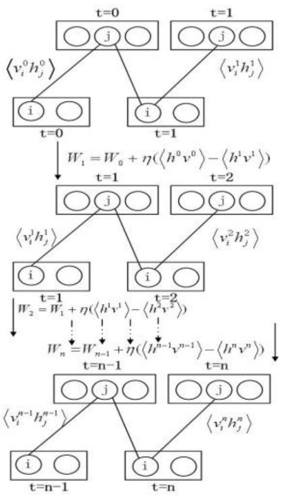
$$CD_n = KL(p_0 \| p_\infty) - KL(p_n \| p_\infty) \quad (6)$$

Maximizing the log probability of the data is exactly the same as minimizing the Kullback–Leibler divergence between the distribution of the data p_0 and the equilibrium distribution p_∞ defined by the model. In each step, the gap is approximately minimized so that we can obtain the final distribution which has the smallest Kullback-Leiler divergence with the fantasy distribution.

After *n* steps, the gradient can be estimated and used in Equation 4 to adjust the weights of RBM. In our experiments, we set *n* to be 1. It means that in each step of gradient calculation, the estimate of the gradient is used to adjust the weight of RBM. In this case, the estimate of the gradient is just the gap between the products of the visual layer and the hidden layer, i.e.,

$$\frac{\partial \log p(v,h)}{\partial W} = \langle h^0 v^0 \rangle - \langle h^1 v^1 \rangle \quad (7)$$

Figure 3 below illustrates the process of learning RBM with CD-based gradient estimation.



t=0
t=1
j
$\langle v_i^0 h_j^0 \rangle$
$\langle v_i^1 h_j^1 \rangle$
i
t=0
t=1
$W_1 = W_0 + \eta(\langle h^0 v^0 \rangle - \langle h^1 v^1 \rangle)$
t=1
t=2
$\langle v_i^1 h_j^1 \rangle$
$\langle v_i^2 h_j^2 \rangle$
t=1
t=2
$W_2 = W_1 + \eta(\langle h^1 v^1 \rangle - \langle h^2 v^2 \rangle)$
$W_n = W_{n-1} + \eta(\langle h^{n-1} v^{n-1} \rangle - \langle h^n v^n \rangle)$
t=n-1
t=n
$\langle v_i^{n-1} h_j^{n-1} \rangle$
$\langle v_i^n h_j^n \rangle$
t=n-1
t=n

Fig. 3. Learning RBM with CD-based gradient estimation

3.3.2 Back-propagation (BP)

The RBM layers provide an unsupervised analysis on the structures of data set. They automatically detect sophisticated feature vectors. The last layer in DBN is the BP layer. It takes the output from the last RBM layer and applies it in the final supervised learning process. In DBN, not only is the supervised BP layer used to generate the final categories, but it is also used to fine-tune the whole network. Specifically speaking, when the parameters in BP layer are changed during its iterating process, the changes are passed to the other RBM layers in a top-to-bottom sequence.

The BP algorithm has a feed-forward step and a back-propagation step. In the feed-forward step, the input values are propagated to obtain the output values. In the back-propagation step, the output values are compared to the real category labels and used them to modify the parameters of the model. We consider the weight w_{ij} which indicates the edge pointing from the *i*-th node in one RBM layer to the *j*-th node in its upper layer. The computation in feed-forward is $o_i w_{ij}$, where o_i is the stored output for the unit *i*. In the back-propagation step, we compute the error *E* in the upper layers and also the gradient with respect to this error, i.e., $\partial E / \partial o_i w_{ij}$. Then the weight w_{ij} will be adjusted by the gradient descent.

$$\Delta w_{ij} = -\gamma o_i \frac{\partial E}{\partial o_i w_{ij}} = -\gamma o_i \delta_j \tag{8}$$

where $-\gamma$ is used to control the length of the moving step.

3.3.3 DBN-based Entity Mention Categorization

For each entity mention, it is represented by the character feature vector as introduced in section 3.2 and then fed to DBN. The training procedure can be divided into two phases. The first phase is the parameter estimation process of the RBMs on all the inputted feature vectors. When a feature vector is fed to DBN, the first RBM layer is adjusted automatically according to this vector. After the first RBM layer is ready, its output becomes the input of the second RBM layer. The weights of the second RBM layer are also adjusted. The similar procedure is carried out on all the RBM layers. Then DBN will operates in the second phase, the back-propagation algorithm. The labeled categories of the entity mention are used to tune the parameters of the BP layer. Moreover, the changes of the BP layer are also fed back to the RBM layers. The procedure will iterate until the terminating condition is met. It can be a fixed number of iterations or a pre-given precision threshold. Once the weights of all the layers in DBN are obtained, the estimated model could be used to prediction.

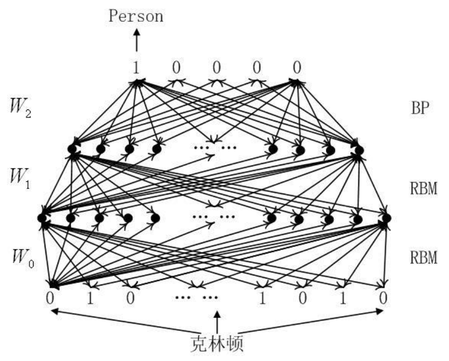


Fig. 4. The mention categorization process of DBN

Figure 4 illustrates the classification process of DBN. In prediction, for an entity mention *e*, we first calculate its feature vector *V*(*e*) and used as the input of DBN. *V*(*e*) is passed through all the layers to get the outputs for all RBM layers and last back-propagation layer. In the *i*th RBM layer, the dimensions in the input vector $V_{input\_i}(e)$ are combined to yield the dimensions of the next feature vector $V_{output\_i}(e)$ as input of the next layer. After the feature vector *V*(*e*) goes through all the RBM layers, it is indeed transformed to another feature vector *V'*(*e*) which consists of complicated combinations of the original character features and contains rich structured information between the characters. This feature vector is then fed into the BP layer to get the final category *c*(*e*).

4 Experiments

4.1 Experiment Setup

In our experiment, we use the ACE 2004 corpus to evaluate our approach. The objective of this study is that the correctly detected Chinese entity mentions categorization using DBN from the text and figure out the suitability of DBN on this task. Moreover, an entity mention should belong to one and only one category.

According to the guideline of the ACE04 task, there are five categories for consideration in total, i.e., Person, Organization, Geo-political entity, Location, and Facility. Moreover, each entity mention is expressed in two forms, i.e., the head and the extent. For example, 美国总统克林顿 'President Clinton of USA' is the extent of an entity mention and 克林顿 'Clinton' is the corresponding head. The two phrases both point to a named entity whose name is Clinton and he is the president of USA. Here we make the "breakdown" strategy mentioned in Li et al. (2007) that only the entity head is considered to generate the feature vector, considering that the information from the entity head refines the name entity. Although the entity extent includes more information, it also brings many noises which may make the learning process much more difficult.

In our experiments, we test the machine learning models under a 4-flod cross-validation. All entity mentions are divided into four parts randomly where three parts are used for training and one for test. In total, 7746 mentions are used for training and 2482 mentions are used for testing at each round. Precision is chosen as the evaluation criterion, calculated by the proportion of the number of correctly categorized instances and the number of total instances. Since all the instances should be classified, the recall value is equal to the precision value.

4.2 Evaluation on Named Entity categorization

First of all, we provide some statistics of the data set. The distribution of entity mentions in each category is given in table 1. The size of the character dictionary in the corpus is 1185, so does the dimension of each feature vector.

Type	Quantity
Person	4197
Organization	1783
Geo-political entity	287
Location	3263
Facility	399

Table 1. Number of entity mentions in each category

In the first experiment, we compare the performance of DBN with some popular classification algorithms, including Support Vector Machine (labeled by SVM) and a traditional BP neutral network (labeled by NN (BP)). To implement the models, we use the LibSVM toolkit[1] for SVM and the neural neutral network toolbox in Matlab[2] for BP. The DBN in this experiment includes two RBM layers and one BP layer. Results of the first experiment are given in Table 2.

Learning Model	Precision
DBN	91.45%
SVM	90.29%
NN(BP)	87.23%

Table 2. Performances of the systems with different classification models

In this experiment, the DBN has three RBM layers and one BP layer. And the numbers of units in each RBM layer are 900, 600 and 300 respectively. NN (BP) has the same structure as DBN. As for SVM, we choose the linear kernel with the penalty parameter C=1 and set the other parameters as default after comparing different kernels and parameters.

In the results, DBN achieved better performance than both SVM and BP neural network. This clearly proved the advantages of DBN. The deep architecture of DBN yields stronger representation power which makes it able to detect more complicated and efficient features, thus better performance is achieved.

In the second experiment, we intend to examine the performance of DBN with different number of RBM layers, from one RBM layer plus one BP layer to three RBM layers plus one BP layer. The amount of the units in the first RBM layer is set 900 and the amount in the second RBM layer is 600, if the second layer exists. As for the third RBM layers, the amount of units is set to 300.

Construction of Neural Network	Precision
Three RBMs and One BP	91.45%
Two RBMs and One BP	91.42%
One RBM and one BP	91.05%

Table 3. Performance of DBNs with different number s of RBM layers

Results in Table 3 show that the performance tends to be better when more RBM layers are incorporated. More RBM layers do enhance the representation power of DBN. However, it is also noted that the improvement is not significant from two layers to three layers. The reason may

[1] available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[2] available at http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/backprop.html

be that two-RBM DBN already has enough representation power for modeling this data set and thus one more RBM layer brings insignificant improvement. It is also mentioned in Hinton (2006) that more than three RBM layers are indeed not necessary. Another important result in Table 3 is that the DBN with One RBM and one BP performs much better than the neutral network with only BP in Table 1. This clearly showed the effectiveness of feature combination by the RBM layer again.

As to the amount of units in each RBM layer, it is manually fixed in upper experiments. This number certainly affects the representation power of an RBM layer, consequently the representation power of the whole DBN. In this set of experiment, we intend to study the effectiveness of the unit size to the performance of DBN. A series of DBNs with only one RBM layer and different unit numbers for this RBM layer is evaluated. The results are provided in Table 4 below.

Construction of Neural Network	Precision
one RBM(300 units) + one BP	90.61%
one RBM(600 units) + one BP	90.69%
one RBM(900 units) + one BP	91.05%
one RBM(1200 units) + one BP	90.98%
one RBM(1500 units) + one BP	90.61%
one RBM(1800 units) + one BP	90.57%

Table 4. Performance of One-RBM DBNs with different number of units

Based on the results, we can see that the performance is quite stable with different unit numbers. But the numbers that are closer to the original feature size seem to be some better. This could suggest that we should not decrease or increase the dimension of the vector feature too much when casting the vector transformation by RBM layers.

Finally, we show the results of the individual categories. For each category, the Precision-Recall-F values are provided in table 5, in which the *F*-measure is calculated by

$$F\text{-measure}=\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}} \quad (9)$$

Type	P	R	*F*
Person	91.26%	96.26%	93.70%
Organization	89.86%	89.04%	89.45%
Location	77.58%	59.21%	76.17%
Geo-political entity	93.60%	91.89%	92.74%
Facility	77.43%	63.72%	69.91%

Table 5. Performances of the system on each category

5 Conclusions

In this paper we presented our recent work on applying a novel machine learning model, the Deep Belief Nets, on Chinese entity mention categorization. It is demonstrated that DBN is very suitable for character-level mention categorization approaches due to its strong representation power and the ability on discovering complicated feature combinations. We conducted a series of experiments to prove the benefits of DBN. Experimental results clearly showed the advantages of DBN that it obtained better performance than existing approaches such as SVM and traditional BP neutral network.

References

David Ackley, Geoffrey Hinton, and Terrence Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science.* 9.

David MacDonald. 1993. Internal and external evidence in the identification and semantic categorization of proper names. *Corpus Processing for Lexical Acquisition,* MIT Press, 61-76.

Geoffrey Hinton. 1999. Products of experts. In Proceedings of the Ninth International. *Conference on Artificial Neural Networks (ICANN).* Vol. 1, 1–6.

Geoffrey Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation,* 14, 1771–1800.

Geoffrey Hinton, Simon Osindero, and Yee-Whey Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation*. 18, 1527–1554 .

GuoDong Zhou and Jian Su. 2002. Named entity recognition using an hmm-based chunk tagger. *In proceedings of ACL*. 473-480.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. *In proceedings of IJCNLP*. 1-7.

Honglei Guo, Jianmin Jiang, Guang Hu and Tong Zhang. 2005. Chinese named entity recognition based on multilevel linguistics features. *In proceedings of IJCNLP*. 90-99.

Jing, Hongyan, Radu Florian, Xiaoqiang Luo, Tong Zhang and Abraham Ittycheriah. 2003. How to get a Chinese name (entity): Segmentation and combination issues. *In proceedings of EMNLP*. 200-207.

Koen Deschacht and Marie-Francine Moens. 2006, Efficient Hierarchical Entity Classifier Using Conditional Random Field. *In Proceedings of the*

2nd Workshop on Ontology Learning and Population. 33-40.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. *In Proceedings of EMNLP'99*.

Nina Wacholder, Yael Ravin and Misook Choi. 1997. Disambiguation of Proper Names in Text. *In Proceedings of the Fifth Conference on Applied Natural Language Processing.*

Solomon Kullback. 1987. Letter to the Editor: The Kullback-Leibler distance. *The American Statistician* 41 (4): 340–341.

Wenjie Li and Donglei Qian. 2007. Detecting, Categorizing and Clustering Entity Mentions in Chinese Text, *in Proceedings of the 30th Annual International ACM SIGIR Conference (SIGIR'07)*. 647-654.

Yoshua Bengio and Yann LeCun. 2007. Scaling learning algorithms towards ai. *Large-Scale Kernel Machines*. MIT Press.