Recognition of Person Relation Indicated by Predicates

Zhongping Liang(✉), Caixia Yuan, Bing Leng, and Xiaojie Wang

School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
dreamfish_liang@163.com, {yuancx,xjwang}@bupt.edu.cn, lb19900314@gmail.com

Abstract. This paper focuses on recognizing person relations indicated by predicates from large scale of free texts. In order to determine whether a sentence contains a potential relation between persons, we cast this problem to a classification task. Dynamic Convolution Neural Network (DCNN) is improved for this task. It uses frame convolution for making uses of more features efficiently. Experimental results on Chinese person relation recognition show that the proposed model is superior when compared to the original DCNN and several strong baseline models. We also explore employing large scale unlabeled data to achieve further improvements.

Keywords: Person relation indicated by predicate · Dynamic Convolution Neural Network (DCNN) · Frame convolution

1 Introduction

The goal of Open Relation Extraction (ORE) is automatically extracting relation triples from large scale of free texts without consulting a prespecified relation vocabulary (Banko et al. 2007) [1]. For instance, the triple <Tom, met, Jim> in the sentence "Tom met Jim" implies an open relation.

Relation between persons is an important subclass of entity relation (ER). One of the most frequent ways to express person relations in Chinese is indicated by predicate phrases. Predicate phrases have been used to indicate a variety of relation types. On the one hand, they can indicate static relationships between persons. For example, the predicate phrase "是...哥哥 (is the brother of)" in the sentence "Tom是Jim的哥哥(Tom is the brother of Jim)" expresses a kinship between Tom and Jim. On the other hand, predicate phrases in sentences may also describe dynamic relationships between persons. For example, the predicate phrase "见到(meet)" in the sentence "Tom见到Jim (Tom met Jim)" expresses that Tom meets with Jim at some time. However, not all predicate phrases do indicate "true" relations. For example, in the sentence "听说这本书是Jim的(Tom heard that this book belonged to Jim)", the predicate "听说 (hear about)" does not indicate any relation between Tom and Jim.

J. Li et al. (Eds.): NLPCC 2015, LNAI 9362, pp. 313–324, 2015.
DOI: 10.1007/978-3-319-25207-0_26

Extracting person relations indicated by predicates is a restricted type of ORE. Since entity type is limited to person, and linguistic form of relation is limited to predicate phrase, it is therefore an easy job to extract a relation triple from a sentence with the form of <person1, verbal phrases, person2> if we confirm this sentence contains a person relation indicated by predicate. But judging whether a sentence contains a relation of this form is still very difficult because the types of relations are not limited. The focus here is therefore to recognize sentences containing this type of relation triples.

Several approaches have been explored to ORE of which the most relevant work to ours is ReVerb proposed by Fader et al. [5]. In their work, the longest verb phrases in sentences are firstly extracted by matching the POS sequence pattern of some predefined rules. Secondly, the verb phrases are filtered by some predefined constraints. Then entity pairs are extracted from both sides of the verb phrase to build relation triples of the form <entity1, verb phrase, entity2>. Finally, these triples are feed to logistic regression classifier with several simple manual features to judge whether a triple indicates a relation. The area under precision and recall curve of ReVerb is 0.47. However, unlike English, verb phrases in Chinese sentences are not always located between the two entities. Wang, et al. [6] investigated 671 predicate phrase based relation sentences and find that in 21 sentences (3.13%) relation mentions locate on the left side of the two entities, in 352 sentences (52.45%) relation mentions locate between the two entities and in 298 sentences (44.41%) relation mentions locate on the right side of the two entities. That is to say, mentions of predicate phrase based relations in Chinese sentences may locate on arbitrary side of the two entities.

This paper introduces a novel approach to recognize if there is a predicate phrase based relation in a Chinese sentence, no matter where the relation mention locates. The main inspiration of our work comes from recent advances in using Dynamic Convolution Neural Network (DCNN) to model sentences (Kalchbrenner et al. 2014) [8]. DCNN does not rely on a parse tree and handles input sentences of variable length. It induces a feature graph over the sentence capturing short and long range relevant structure features. It has been proved that DCNN outperforms other traditional approaches in both sentiment prediction and question classification. We improve DCNN in several ways and transplanted it into our task. Experimental results show that our model outperforms the original DCNN and some other baseline approaches.

2 Related Work

Previous close relation extraction aims at assigning each entity pair in a sentence a relation type. Zelenko et al. (2003) [9] proposed to extract relations by computing kernel functions between parse trees. Zhou et al. (2005) [10] explored various features in relation extraction, these features include word, entity type, mention level, overlap, phrase chunking, dependency tree, parse tree and gazetteers. Mintz et al. (2009) [11] proposed a distant supervised learning method. They first employed freebase to weakly annotate a large scale of training set, then

trained a logistic regression classifier with lexical and syntactical features. The application of close relation extraction is limited because the relation types are predefined and hard to be extended.

For other open relation extraction methods, lexical pattern approaches such as Snowball (Agichtein and Gravano, 2000) [3] and KnowItAll (Etzioni et al. 2003) [4] start with a small number of seed instances or patterns, then expand instance set and pattern set with bootstrap learning. The initial instances are first used with a large corpus to extract a new set of patterns; these new patterns are then used to extract a new set of instances; these new instances are then used to extract more patterns; the two steps are alternated in an iterative fashion until no new patterns and instances are generated or the iteration time reaches a given threshold. The assumption under lexical pattern approaches is that relation mentions have common patterns in lexical level. However, arbitrary verbs in Chinese sentences are potential mentions of predicate phrase based relations, they don't share common patterns in lexical level. Thus bootstrap learning can't extract relation indicated by predicates. Parse Tree approaches such as WOE (Fei et al.) [7] get high performance using the shortest dependency path in a parse tree. Systems using parse tree features are not practical. On the one hand, parsers are usually trained with domain-limited Treebanks, they make more parsing errors when confronted with the diversity; on the other hand, time complexity of parsers is explosive, which keeps these systems from being applied to large scale of web text.

Recently, some explorations that using deep neural network to serve Natural Language Processing (NLP) tasks began to emerge. The Deep Neural Networks with Multi-task Learning (Collobert and Weston 2008, Collobert et al. 2011) [13,14] was proposed to solve the sequence labelling problems. Their network first projects multiple features of each word into their respective vectors. Then these projected feature vectors are fed to a convolution layer and a max over time layer, followed by a softmax output layer. Their network using large scale of unlabeled data outperforms other traditional approaches in Part-of-Speech Tagging, Chunking, Named Entity Recognition and Semantic Role Labeling. The Dynamic Convolutional Neural Network (DCNN) (Kalchbrenner et al. 2014) [8] firstly projects words to their vectors, secondly alternates wide convolution layers and dynamic k-max pooling layers to get a rich set of inner representations. Finally these representations are feed to a fold layer and a full connected layer, followed by output softmax layer. DCNN outperforms other traditional approaches in both sentiment prediction and question classification.

3 The Model

This section first describes the structure of our revised DCNN model[1], then lists the differences between our model and the original DCNN, and finally explains some details in training.

[1] Code available at https://github.com/dreamfish-liang/FDCNN/

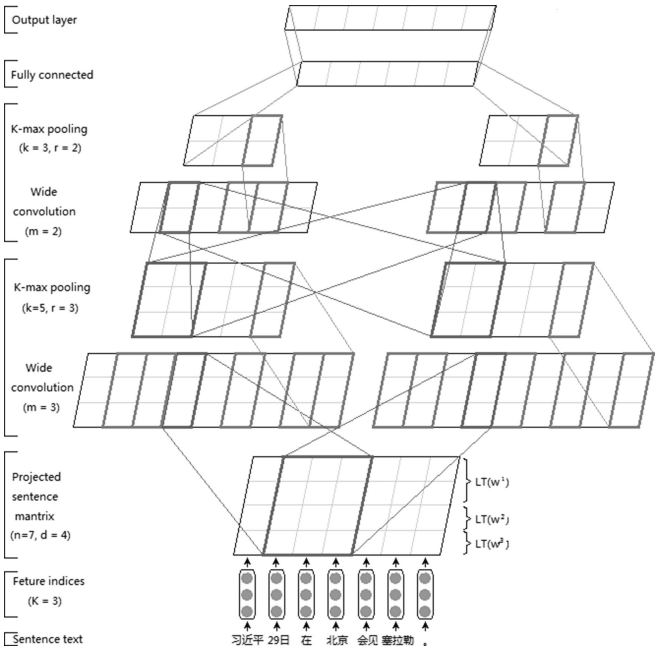

Fig. 1. The overall structure of improved DCNN

Fig.1 shows the overall network structure of our revised model for recognizing person relation indicated by predicates. It includes several forward layers. The network receives a sentence as input in the first layer, and the input sentence is projected to a matrix by lookup table operation in the second layer. This projected matrix, which includes multiple features of each word in the input sentence, is then fed to two sets of convolution and pooling layers (4 layers) to generate a rich set of inner representations. A full connection operation over the last pooling results is used to produce a fixed length sentence embedding for the variable length sentence in final hidden layer. And the network finally feeds the sentence embedding into a softmax classifier.

In Fig.1, the length of the exampled input sentence is 7. The number of different features extracted from each word is 3, and the dimension of each feature vector is respectively 2, 1 and 1 (totally 4). There are 2 convolution layers and 2 k-max pooling layers. Convolution is marked with red line, pooling is marked with green line and fully connection is marked with blue line. The widths of the convolution window at the two layers are respectively 3 and 2. The convolution output matrices have row numbers of 3 and 2. The dynamic k-max pooling layers have values k of 5 and 3.

The main differences between the above model and the original DCNN are:

1. The original DCNN only employs word as feature, while we import multiple features besides word.

2. Instead of one-dimensional convolution adopted by the original DCNN, a special two-dimensional convolution called frame convolution is used in our network.
3. The fold layer in the original DCNN is removed in our model, and is replaced by a smooth reduction produced by frame convolution.

Details of the improvements and some other descriptions are given in following parts.

3.1 Transforming Feature Indices into Vectors

The original DCNN employs only word feature, which limits its ability to capture diverse information from the input sentence. In our improved model, each word in the sentence can be represented with a couple of exact K different features. The idea that employing multiple features for each word comes from Collobert et al. (Collobert and Weston 2008, Collobert et al. 2011) [13,14]. These features can be word itself, stem, POS, indicator indicating if a word is in a given dictionary, or something else reasonable. The first layer of the network, called lookup table layer, projects the overall K different features of each word to their respective feature vector.

Formally, for the i-th feature w^i of each word w, there is a dictionary D^i and a Feature Embedding matrix $W^i \in \mathbb{R}^{|D^i| \times d^i}$ corresponding with it, where $|D^i|$ indicates the size of dictionary D^i, d^i is the dimension of the i-th feature's vector. We use w_j^i to indicate the i-th feature of the j-th word in a sentence. A lookup table operation which transforms a feature to a represented vector is described as follows:

$$LT(w_j^i) = {W^i}^T \cdot \phi(w_j^i) \tag{1}$$

where $\phi(w_j^i) \in \mathbb{R}^{d^i}$ is the binary one-hot representation of w_j^i indicating absence or presence of this feature. After lookup table layer, we obtain a frame vector representation x_j of word w_j by concatenating all result feature vectors one-by-one:

$$LT(w_j) = \left(LT(w_j^1)^T, \ldots, LT(w_j^K)^T\right)^T \tag{2}$$

where K indicates the number of different features.

Through a lookup table layer, an input sentence S is transform into a sentence matrix x:

$$x = (x_1, x_2, \ldots, x_n) \tag{3}$$

where $x_t = LT(w_t)$, $\forall t\ x_t \in \mathbb{R}^d$, $d = \sum_{i=1}^{K} d^i$, and n is the length of the input sentence. The row size of matrix x which depends on the number of different features and the dimension of each feature vector is constant, while the column size of matrix x which depends on the length of input sentence is mutable.

3.2 Frame Convolution

Instead of one-dimensional convolution which applies convolution operation row by row in the original DCNN, we introduce a frame convolution here. Frame convolution, which is a particular two-dimensional convolution, is an operation between a weight matrix and a sequence of input column vectors, resulting in another sequence of output vectors. The idea of frame convolution mainly comes from Time-Delay Neural Network (TDNN, Waibel et al. 1989) [15]. In frame convolution, the value of output sequence vectors at time t denoted by $o(t)$ is:

$$o(t) = M\left(x_{t-\frac{m-1}{2}}{}^T, \ldots, x_{t+\frac{m-1}{2}}{}^T\right)^T + b \tag{4}$$

where $M \in \mathbb{R}^{r\times(m\times d)}$ is the weight matrix, $b \in \mathbb{R}^r$ is the bias term,m is the size of convolution window, and r which is a parameter can be set by user is the dimension of the output vector $o(t)$.

Convolution can be divided into two types, wide convolution and narrow convolution. Convolution between weight matrix of column size m and input sequence of length n results an output sequence of length $m + n - 1$ in wide convolution and $m - n + 1$ in narrow convolution. Kalchbrenner et al. [8] have explained that applying the weights M in a wide convolution has some advantages over applying them in a narrow one. Therefore we adopt a wide one in our experiments.

3.3 Dynamic *K*-max Pooling

To settle the problem of sentence of various length which makes the sentence inadequate to be feed to a traditional neural network, a pooling layer is necessary. A max over time operation takes out the max value of each row (Collobert and Weston 2008, Collobert and Weston 2011) [13,14], while a k-max pooling operation takes out the top k max values of each row maintaining their original order (Kalchbrenner et al. 2014) [8]. In frame convolution, each column of output matrix is acquired from combination of m continuous columns in the input layer. Therefore, each column of the output matrix can also be regard as a represented m-gram. It's plausible that treating each column of the output matrix as an atomic element when applying k-max pooling operation is better than breaking it down. This character performs especially obvious when words employ multiple features. Following this intuition, in our architecture, we apply k-max pooling column-wise. That is we select k columns with largest 2-norm value and keep their order. Also, the pooling parameter k is not fixed, but is dynamically selected in order to allow for a smooth extraction. A dynamic k-max pooling operation is a k-max pooling operation where k is a function of the length of the sentence and the depth of the network. We simply let k be a linear function as follows:

$$k_l = \max\left(k_{top}, n + \lfloor\frac{(k_{top} - n)\times l}{L}\rfloor\right) \tag{5}$$

where n is the length of input sentence, l is the number of the current convolutional layer to which the pooling is applied, L is the total number of convolutional

layers in the network, and k_{top} is the fixed pooling parameter for the top-most convolutional layer.

3.4 One-Dimensional Convolution vs Frame Convolution

Fig. 2 illustrates the differences made by replacing one-dimensional convolution with frame convolution. One-dimensional convolution is shown in the left side, and frame convolution is shown in the right side. Both one-dimensional convolution and frame convolution employ the same two features. The first two rows in both input matrices are projected from the first feature, and the third row is projected from the second feature. Convolution is marked with red line, pooling is marked with green line.

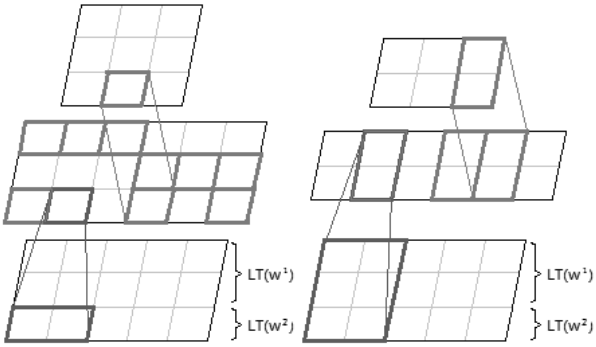

Fig. 2. The comparison of one-dimensional convolution and frame convolution

The one-dimensional convolution is an operation between a vector of weights and a vector of inputs, and is calculated between weight matrix and input matrix row by row. Convolutions of different features are computed independently. Contrast to one-dimensional convolution, in frame convolution every column of input matrix is regarded as an entirety so that units of two features is computed together during every convolution operation. In this way, different features is combined by the operation of convolution, which provides the possibility of modeling correlations and constraints between different features.

After convolution, k-max pooling operation is again acted on the input matrix row by row in one-dimensional convolution structure. In this way, the order of different values may be disrupted when several rows are taken into account although order in each row can be maintained easily. Consequently, values in the same column of result may indicate features of different words, which may disturb structure information of features. However, reviewing the frame convolution, a column is regard as a whole so that not only the relative order between rows but also that in the same row can be kept at the same time. It will help when modeling sentence.

3.5 Training

In our implement, hyperbolic tangent *tanh* is selected as non-linear function between every convolution layer and pooling layer. The network is trained by minimizing a loss function over the training data using stochastic gradient descent by backpropagation. And the objective function includes an $L2$ regularization term over the parameters.

4 Experiments

In this section, we first describe the data set on which we conduct experiments. Then we describe the three experiments respectively. We compare our model with some other models in first experiment, investigate the efficient of making use of multiple features of our model in second experiment, and explore employing large scale unlabeled data to achieve further improvement in third experiment.

4.1 Data Set

Since there was no public available corpus on extracting person relation indicated by predicate structure, we built a corpus for it. A large number of news documents from SohuNews were crawled. Raw texts are then segmented into sentences. Each sentence was segmented by a Chinese-Word-Segment (CWS) tool and a Part-Of-Speech (POS) tagger provided by FudanNLP (Xipeng et al. 2013)[16]. We randomly selected some sentences, and manually labeled if the sentence contains exactly a predicate indicated relation between two persons.

We finally get totally 10000 samples, where 3107 positive samples (including a predicate indicated relation between two persons) and 6893 negative samples. In the following experiments, we separate all the 10000 samples into two parts, 9000 for training and 1000 for testing. The ratios of positive samples and negative samples in both parts are same.

4.2 Experiment-1

In our first experiment, we propose to make a comparison between traditional classifiers and DCNNs to prove that DCNNs are effective enough to model sentences. For traditional classifier, we choose Maximum Entropy (ME), Naïve Bayes (NB), and Support Vector Machine (SVM) as control experiments. For DCNNs, we implement both one-dimensional convolution schema and frame-convolution schema. The most primary feature of a sentence is word itself, so we simply employ bag-of-word (BOW) feature for all above classifiers. Note for DCNNs, we replace all person names by a special identifier <PER> in all sentences of training and testing set.

At the meantime, we compare DCNN with traditional relation classification approach. Zhou et al. (2005) [10] employed word, entity type, mention level,

overlap, phrase chunking, dependency tree, parse tree and gazetteers for relation classification task. Entities in our problem are known and limited to persons, so features such as entity type, mention level and overlap are helpless. We here employ word and parse tree feature. The sentences are parsed with Stanford Parser (Klein et al. 2003) [18]. And we use SVMLight with Tree Kernels (Joachims et al. 1999) [19] (Moschitti et al. 2004, 2006) [20,21].

For both 1D-DCNN and F-DCNN, the dimension of word vector is set to 40, the number of convolution layers is 2, the number of convolution filters each layer is respectively 5 and 7, the size of convolution windows each layer is respectively 5 and 5, the number of hidden unit of fully connected layer is 10. For F-DCNN, the row number of convolution result matrices is respectively 20 and 10. The result is shown in Table 1.

Table 1. Result of Experiment-1

Approach	**Pre. (%)**	**Rec. (%)**	**F1 (%)**
NB	64.19	49.84	56.11
ME	57.46	57.82	57.64
SVM	73.33	38.65	50.62
SVMLight	75.86	49.20	59.68
1D-DCNN	62.50	55.91	59.02
F-DCNN	64.53	58.14	61.17

In Table 1, approaches are list in the first column, and precision, recall and F1-score are described in other columns respectively.

We see that the DCNNs significantly outperforms NB, ME and SVM in F1-score with the same bag-of-word feature. This is because that besides words, DCNN can get information about combination of words and grammars and their orders. Further on, F-DCNN outperforms traditional Tree Kernel approach (SVMLight). This may be caused by two reasons. Firstly, parsers are usually trained with domain-limited Treebanks, they make more parsing errors when confronted with web texts, and parse errors directly influence the classification results; secondly, when a tree is large, it's hard for a kernel function to capture the most key structure from the parse result. In addition, parsing is extremely time consuming, and improper to deal with large scale of web text.

F-DCNN also performs better than 1D-DCNN. Since every positive sample in our data set expresses a person relation indicated by predicates. Therefore, the common characteristic of all positive samples is that there is a dependency path connecting the two persons and the predicate phrase. A higher F1-score implies that frame convolution may have a greater ability to capture the structure information of sentences than a one-dimensional one.

4.3 Experiment-2

In this experiment, we explore more useful features to improve performance of our model. At the same time, we also intent to compare frame convolution schema network with one-dimensional convolution schema network on the performance of making use of multiple features. Since predicate in a sentence is always a verb phrase, POS feature might be a type of useful feature. The two persons are participants of relations, so making the two person names high light and labeling each word with relative offset from the two persons may also helpful. So we try these features in this experiment. The dimension of vectors for word, POS, person name high light and word's offsets from the first and second person names are respectively 40, 10, 5, 5 and 5 (totally 65). Other parameters are the same with what in the first experiment. The experiment results are shown in Table 2.

Table 2. Result of Experiment-2

Approach	Pre. (%)	Rec. (%)	F1 (%)
1D-DCNN-W	62.50	55.91	59.02
1D-DCNN-WP	64.88	61.98	63.39
1D-DCNN-WPP	59.33	62.93	61.08
1D-DCNN-WPPO	67.38	60.06	63.51
F-DCNN-W	64.53	58.14	61.17
F-DCNN-WP	68.51	59.10	63.46
F-DCNN-WPP	65.56	63.25	64.39
F-DCNN-WPPO	72.62	63.57	67.80

In Table 2, W denotes word; WP denotes word together with POS; WPP denotes word POS and person name high light (The value of this feature is 'yes' if a word is a person name, and 'no' otherwise.); WPPO denotes word, POS, person name high light and word's offsets from each of the two persons.

F-DCNN performs better than 1D-DCNN at any case. F-DCNN significantly out-performs 1D-DCNN when employing multiple features. It is due to two main reasons. One is that one-dimensional convolution doesn't blend different features of the same word together since the convolution operation between input sentence matrix and weight matrix is calculated row by row. Another reason is that pooling in 1D-DCNN is calculated row by row, and this breaks the relative positions of different features in high layer of the network. Missing order information amount different features may result in a higher error rate.

4.4 Experiment-3

The scale of our data set is totally 10000 samples, with a word vocabulary of size around 30000. We calculate out that more than 20000 (78.62%) words appear no

more than 3 times and more than half (57.19%) appear only once. This may lead to a drift of overfitting. Therefore, in our last experiment, we propose to explore employing large scale unlabeled data to improved performance. Unlabeled data again comes from SohuNews Web documents with a total size of 5GB and is handled with FudanNLP by CWS and POS. We replace all person names with a particular identifier <PER>, and train a word embedding of dimension 40 using Google open source word2vec (Mikolov et al. 2013) [17]. The parameters of DCNN are the same as what is in Experiment-2. Besides, the word vectors are initialized with what we obtain by word2vec, and fixed while training.

Table 3. Result of Experiment-3

Approach	**Pre. (%)**	**Rec. (%)**	**F1 (%)**
F-DCNN-WPPO	72.62	63.57	67.80
F-DCNN-WPPO+UN	73.47	65.49	69.25

In Table 3, F-DCNN-WPPO is the same as what is in table 2, and UN denotes using unlabeled data.

We see that, large scale of unlabeled data improves the performance in both precision and recall. After trained with large scale of unlabeled data by word2vec, words which are similar in syntax and semantic are close in their embedding space. This information is indeed helpful for recognizing person relation indicated by predicate.

5 Conclusion

This paper investigated an improved DCNN model to detect whether a sentence contains a relation. The present method extended 1D convolution to 2D. Improved model achieves better performance than original model. We also find that large number of unlabeled data is helpful. Although the model employs multiple features of each word, it fails in employing global features of a sentence. It might be a future work.

Acknowledgments. This work was partially supported by National Natural Science Foundation of China (No.61273365, No.61202248), discipline building plan in 111 base (No.B08004) and Engineering Research Center of Information Networks, Ministry of Education.

References

1. Banko, M., Cafarella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open Information Extraction from the Web. In: IJCAI, pp. 2670–2676 (2007)
2. Weston, J., Bordes, A., et al.: Connecting language and knowledge bases with embedding models for relation extraction. arXiv preprint arXiv:1307.7973 (2013)

3. Agichtein, E., Gravano, L.: Snowball: Extracting relations from large plain-text collections. In: Proceedings of the Fifth ACM Conference on Digital Libraries. ACM (2000)
4. Etzioni, O., Cafarella, M., Downey, D., et al.: Unsupervised named-entity extraction from the web: An experimental study. Artificial Intelligence **165**(1), 91–134 (2005)
5. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1535–1545. Association for Computational Linguistics (2011)
6. Wang, M., Li, L., Huang, F.: Semi-supervised Chinese Open Entity Relation Extraction. In: Proceedings of 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems, TP391.1 (2014)
7. Wu, F., Weld, D.S.: Open information extraction using Wikipedia. In: ACL 2010 Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 118–127 (2010)
8. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
9. Zelenko, D., Aone, C.: Richardella A: Kernel methods for relation extraction. The Journal of Machine Learning Research **3**, 1083–1106 (2003)
10. GuoDong, Z., Jian, S., Jie, Z., et al.: Exploring various knowledge in relation extraction. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 427–434. Association for Computational Linguistics (2005)
11. Mintz, M., Bills, S., et al.: Distant supervision for relation extraction without labeled data. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pp. 1003–1011. Association for Computational Linguistics (2009)
12. Bengio, Y., Ducharme, R., Vincent, P., et al.: A neural probabilistic language model. The Journal of Machine Learning Research **3**, 1137–1155 (2003)
13. Collobert, R., Weston, J.: A unified architecture for natural language processing: Deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 160–167. ACM (2008)
14. Collobert, R., Weston, J., Bottou, L., et al.: Natural language processing (almost) from scratch. The Journal of Machine Learning Research **12**, 2493–2537 (2011)
15. Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K.: Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing **37**, 328–339 (1989)
16. Qiu, X., Zhang, Q., Huang, X.: FudanNLP: A Toolkit for Chinese Natural Language Processing. In: Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL) (2013)
17. Mikolov T., Chen K., Corrado G., et al.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
18. Klein, D., Manning, C.D.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423–430 (2003)
19. Joachims, T.: Making large scale SVM learning practical. Universit Dortmund (1999)
20. Moschitti, A.: Making Tree Kernels Practical for Natural Language Learning. In: EACL, vol. 113(120), p. 24 (2006)
21. Moschitti A.: A study on convolution kernels for shallow semantic parsing. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, p. 335. Association for Computational Linguistics (2004)