Wenpeng LU[1,2], Heyan HUANG[1,3] , Chaoyong ZHU[4]

Beijing Institute of Technology (1)
Shandong Polytechnic University (2)
Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Application (BIT) (3)
University of Science &Technology of China (4)

Feature Words Selection for Knowledge-based Word Sense Disambiguation with Syntactic Parsing

Abstract. *Feature words are crucial clues for word sense disambiguation. There are two methods to select feature words: window-based and dependency-based methods. Both of them have some shortcomings, such as irrelevant noise words or paucity of feature words. In order to solve the problems of the existing methods, this paper proposes two methods to select feature words with syntactic parsing, which are based on phrase structure parsing tree (PTree) and dependency parsing tree (DTree). With the help of syntactic parsing, the proposed methods can select feature words more accurately, which can alleviate the effect of noise words of window-based method and can avoid the paucity of feature words of dependency-based method. Evaluation is performed on a knowledge-based WSD system with a publicly available lexical sample dataset. The results show that both of the proposed methods are superior to window-based and dependency-based methods, and the method based on PTree is better than the method based on DTree. Both of them are preferred strategies to select feature words to disambiguate ambiguous words.*

Streszczenie. *W artykule zaproponowano dwie metody selekcji cech słowa bazujące na analizie składni struktury frazy oraz analizie składni zależności. Badania przeprowadzono przy wykorzystaniu rożnych baz danych. Proponowana metoda ma większą dokładność niż dotychczas stosowane metody: okna i zależności. (**Selekcja cech słowa dla jednoznacznego wykrywania znaczenia z syntaktyczną analizą składni**)*

Keywords: Phrase structure parsing, Dependency parsing, Parsing tree, Word sense disambiguation.
Słowa kluczowe: analiza składni frazy, cechy słowa.

Introduction

Word sense disambiguation (WSD) is to automatically determine the meanings of ambiguous words based on their context, which is one of hot-spot issues in natural language processing (NLP). WSD is one of basic tasks in NLP, which is crucial for natural language understanding and has important applications in machine translation, information retrieval and question-answering system[1]. As Firth said, "You shall know a word by the company it keeps."[2] The meaning of the ambiguous word is related with its context, which includes lexical feature, syntactic feature, semantic feature and discourse feature[3]. The effective selection of features and the assignment of WSD weights for them are the key problems for WSD, which would directly affect their performance.

Lexical feature, that is context feature words, is the most important basis of knowledge-based WSD. In general, there are two kinds of method to select feature words: window-based method and dependency-based method. The former is easy to be realized, but it may induce some short-distance irrelevant noise words and omit some long-distance relevant words. The latter can exactly select feature words, but it is confused with the paucity of feature words.

In order to solve the shortcomings of existing methods, this paper proposes the strategy to select feature words based on syntactic parsing. The key premise of our work is that: the words that are more adjacent on syntactic relation have stronger semantic relation, so they are more suitable to be selected as feature words each other. There are two kinds of syntactic parsing: phrase structure parsing and dependency parsing. The paper respectively puts forward the methods based on phrase structure parsing and dependency parsing, which select feature words and assign WSD weights to them based on syntactic parsing tree.

In order to evaluate the performance of proposed methods, a knowledge-based WSD experimental platform has been designed. With the platform, experiments have been done on a publicly lexical sample dataset[4]. The experimental results demonstrate the good performance of the two proposed methods. Both of them are superior to the window-based and dependency-based method, and the method based on phrase structure parsing is better than the method based on dependency parsing.

The rest of this paper is structured as follows. Section 2 introduces some related work on WSD and selection of feature words. Section 3 describes the two proposed method for feature words selection based on two kinds of syntactic parsing tree. The experiments are presented in section 4. At last, the conclusions are drawn and further work is mentioned.

Related Work

Methods of WSD can be divided into supervised, unsupervised, semi-supervised and knowledge-based methods. Supervised systems[5] need to be trained with sense-tagged corpus, learn the relationship between the specific sense and the context, and get a classifier for each word. Unsupervised approaches[6] utilizes clustering technique to cluster words based on their context to distinguish senses. Semi-supervised systems[7] adopt bootstrapping methods which learn knowledge from a small sense-tagged corpus and extend their knowledge with the existing knowledge. Knowledge-based methods[8, 9] mainly utilize external knowledge base, such as dictionary and ontology et al., and compute sense relatedness[10] or gloss overlaps to choose the most appropriate sense. In the paper, the methods to select feature words are mainly for knowledge-based WSD.

Patwardhan et al.[11] and Pederson et al.[12] have proposed the method to disambiguate the ambiguous words with their context words in a certain length of context window. Firstly, *2N* content words around the target word are selected as feature words. Secondly, the relatedness of each sense of ambiguous words and feature words are computed based on WordNet. The sense which has the maximum overall relatedness with feature words is chosen as the right sense.

McCarthy et al.[8] have proposed the method to disambiguate the ambiguous words based on distributional similarity and semantic relatedness. Firstly, they select feature words based direct dependency relation. They parse a corpus with dependency parser to get a great deal of dependency triples. Based on the dependency triples, distributional similarities among words are computed and top-*N* similar words are chosen as feature words[13]. Secondly, the relatedness between each sense of ambiguous word and feature words is computed. The sense

with the maximum weighted sum of relatedness is select as the right sense.

Agirre et al.[9] have proposed the method for WSD with personalized PageRank. Similar with McCarthy et al, they collect feature words with direct dependency relation. Besides, Lu et al.[14] have proposed a WSD method based on dependency relation and Bayes model. They also select feature words with direct dependency relation.

To sum up the existing works, the methods to select feature words can be divided into either window-based or dependency-based method. The window-based method selects context words in a certain length of window as feature words. It is easy to be realized, but it doesn't consider any syntactic and semantic relations, so it is likely to induce some short-distance irrelevant words and omit some long-distance relevant words. The dependency-based method can select feature words accurately. As it only selects the words with direct dependency relations as feature words, it often only collects few feature words which is not enough to disambiguate the target words.

Chen et al.[15] have proposed a WSD strategy based on dependency parsing tree matching. In the strategy, Firstly, a large scale dependency knowledge base is built. Secondly, with the knowledge base, the matching degree between the parsing trees of each sense gloss and the sentence are computed. The sense with the max matching degree would be selected as the right sense. Inspired by the strategy, we propose to select feature words based on parsing tree. The words that are more adjacent on parsing tree have stronger semantic relation. With the help of parsing tree, the defects of windows-based and dependency-based methods can be alleviated greatly. The paper respectively puts forward the methods based on phrase structure parsing tree and dependency parsing tree.

Proposed Methods

Syntactic parsing is the process of analyzing the structure of a sentence with respect to certain formal grammar, which can determine the relations among the words and the roles of them. The syntactic structure is commonly expressed with tree data structure, which is usually called as syntactic parsing tree, that is parsing tree. According to the difference of grammars[16, 17], there are phrase structure tree (PTree) and dependency tree (DTree). The paper respectively proposes the methods to utilize PTree and DTree to select feature words.

Method based on PTree

Phrase structure grammar was proposed by Chomsky in 1956[16], which was a mature formal language theory and was applied widely. Phrase structure grammar is defined by a finite set of all vocabulary (alphabet) V_p, a finite set $\sum$ of initial string in V_p, and a finite set F of rules of the form: $X \to Y$, where X and Y are strings in V_p. Each rule means to rewrite X with Y. According the rewrite rules, we can rewrite the string continuously until PTree is generated. Here is an example sentence: *The coaches produced by FAW corporation and CN heavy duty truck factory brought the workers to the plant* . Its PTree is as Fig.1.

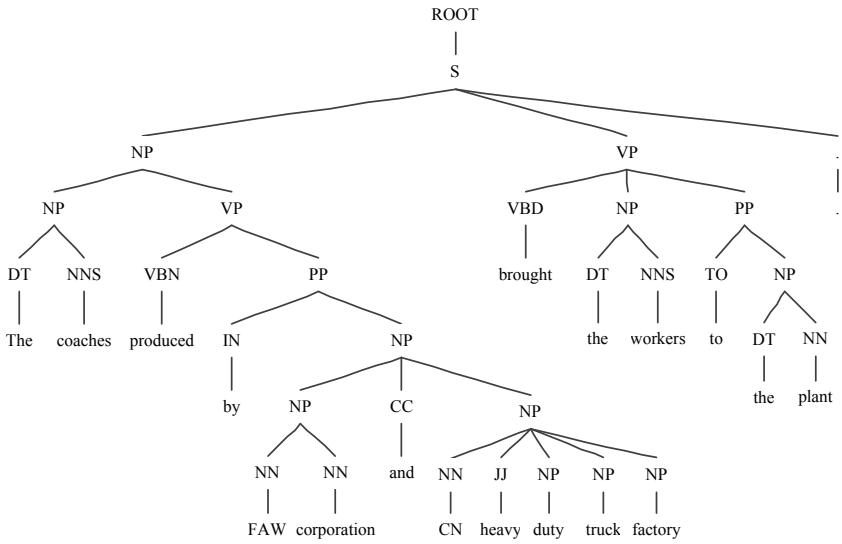

Fig. 1. Phrase Structure Parsing Tree (PTree)

PTree reflects hierarchical constituent relationships among the words and phrases of the sentence. The words that has the common ancestor and adjacent on PTree have stronger syntactic and semantic relations. According to the hierarchy structure of PTree, from the leaf node of ambiguous word to the root node, the adjacent words on PTree are collected layer by layer as feature words. According to relative position between the feature word and ambiguous word, which includes hierarchical relation and path distance on PTree, WSD weight of the feature word is assigned. Then, feature words are sorted on descending order by weight. Top-*N* feature words are selected to disambiguate the target word.

The algorithm to select feature words based on PTree is as follow.

Algorithm 1
the algorithm to select feature words based on PTree

Input:

PTree: the phrase structure parsing tree of the sentence

TargetWord: the ambiguous word

FWordNum: the number of feature words that are needed

OutPut:

PFWordSet: feature words that are selected based on *PTree*

Step1. Initialize the variable *TargetWordNode* with the leaf node of *TargetWord* on *PTree*, initialize the variable *CurNode* with it. Besides, initialize *PFWordSet* with *null*.

Step2. Based on *PTree*, expand from *CurNode* to its father node – *Father*. If *Father* is *null*, go to Step 5; otherwise, go to Step 3.

Step3. Get the set of the child nodes of *Father* – { *child* }. If the size of { *child* } is 1, save *Father* to *CurNode,* go to step 2; otherwise, add 1 to the layer number variable – *layer*, go to step 4.

Step4. Traverse { *child* } to select feature words in this iterator of expansion.

For each *child* in { *child* }
Initialize *LeafSet* with the leaf nodes of *child*
If *TargetWordNode* $\notin$ *LeafSet*
For each *leafnode* in *LeafSet*
Record the two kinds of information for *leafnode* (that is, layer number, path distance between the word and *TargetWord* on *PTree*)
Compute WSD weight of the *leafnode* with Eq.(1)
Add *leafnode* to *PFWordSet*
EndFor
Endif
EndFor

Step5. Sort *PFWordSet* on descending order by weight, and exclude the words whose orders are greater than *FWordNum*.

Step6. Return *PFWordSet*.

Note that, in Step 3, for *layer*, only the layer that has more than one child is counted. In this example, "trunk" can be collected after two layers are expanded. Its layer number is 2, path distance is 9. In Step 4, the WSD weight is assigned with Eq.(1).

$$weight(f_j) = \frac{1}{l^{\alpha}} \cdot \frac{1}{1+\beta \log_{10} d} \tag{1}$$

In the Eq.(1), f_j represents *j-th* feature word, l and d represent layer number and path distance on PTree. α and β are tuning factors, which adjust the effects of l and d.

Method based on DTree

Dependency grammar was established by Tesnière in 1959[17]. With dependency grammar, we can obtain the dependency relations among the words in a sentence. Dependency parsing structure can be represented with dependency triple or dependency parsing tree. The form of dependency triple is as: *relation (governor , dependent)*. With dependency triples, dependency parsing tree can be generated.

For the standard dependency syntactic parsing, its result is a standard tree, where each token of the sentence appears, which is a more surface-oriented representation. But, for WSD, it can be useful to regard some words, such as prepositions and conjunctions.[18] The dependency representation of collapsed dependencies with propagation of conjunct dependencies is a preferred representation, which is a more semantically interpreted representation. Dependencies involving prepositions, conjuncts, as well as information about the referent of relative clauses are collapsed to get direct dependencies between content words. Besides, dependencies involving the conjunctions are propagated. The dependency representation of collapsed dependencies with propagation of conjunct dependencies is more suitable to select feature words for WSD.

Here is an example sentence: *The coaches which brought the workers to the plant are produced by FAW corporation and CN heavy duty truck factory*. The results of dependency parsing for the example are as follow:

det(coaches-2, The-1)
nsubj(brought-4, coaches-2)
nsubjpass(produced-11, coaches-2)
rcmod(coaches-2, brought-4)
det(workers-6, the-5)
dobj(brought-4, workers-6)
det(plant-9, the-8)
prep_to(brought-4, plant-9)
auxpass(produced-11, are-10)
nn(corporation-14, FAW-13)
agent(produced-11, corporation-14)
nn(factory-20, CN-16)
amod(factory-20, heavy-17)
nn(factory-20, duty-18)
nn(factory-20, truck-19)
agent(produced-11, factory-20)
conj_and(corporation-14, factory-20)

DTree of the sentence is as Fig.2.

As shown in Fig.2, when dependencies are collapsed and propagated, the node may have more than one father node and there may appear cycles on DTree. Strictly speaking, it is a directed graph and no longer a real tree. But for the convenience of expression, we still call it DTree.

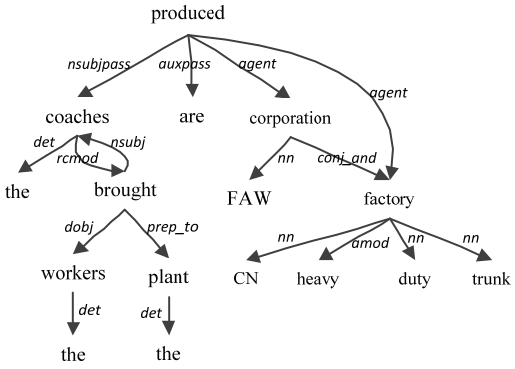

Fig. 2. Dependency Parsing Tree (DTree)

DTree reflects semantic dependency relations among the words of the sentence. The more adjacent words on DTree have stronger semantic relationship. Similar with method based on PTree, according to the structure of DTree, from the node of ambiguous word to other nodes, the adjacent words on DTree within a certain length of shortest path are collected as feature words. According to the length of shortest path between the feature word and ambiguous word, WSD weight of the feature word is assigned.

The algorithm to select feature words based on DTree is as follow.

Algorithm 2
the algorithm to select feature words based on DTree

Input:
DTree: the dependency parsing tree of the sentence
TargetWord: the ambiguous word
MaxDist: the max length of the shortest path between feature words and ambiguous word

OutPut:
DFWordSet: feature words that are selected based on *DTree*

Step1. Initialize the variable *TargetWordNode* with the node of *TargetWord* on *DTree*, initialize *DFWordSet* with *null*.

Step2. With Dijkstra algorithm, get the length of shortest path (*len*) between *TargetWord* and each other content node on *DTree*. And, add the nodes and their path lengths into the set – { *node* }.

Step3. Traverse { *node* } to select feature words.

```
    For each node in { node }
      If node.len <= MaxDist
        Compute WSD weight of the node with Eq.(2)
        Add node to DFWordSet
      Endif
    EndFor
```

Step4. Return *DFWordSet*.

In Step 3, the WSD weight is assigned with Eq.(2).

$$weight(f_j) = \frac{1}{d^{\alpha}} \tag{2}$$

In the Eq.(2), f_j represents *j-th* feature word, d represents the length of the shortest path between f_j and ambiguous word on DTree. *α* is tuning factor, which adjusts the effect of d.

Experiments

In order to evaluate the effectiveness of proposed methods, a knowledge-based WSD platform has been designed. The platform selects feature words with different methods. Through the comparison of WSD performance, this paper evaluates the effectiveness of proposed methods on a publicly available lexical sample dataset. The window-based, dependency-based, PTree-based and DTree-based methods are compared.

Framework of WSD Platform

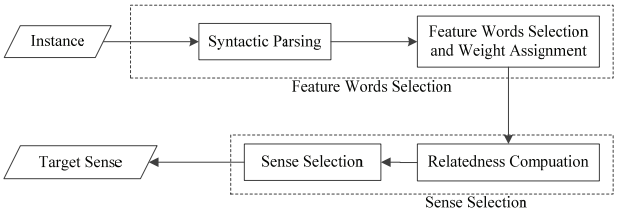

Fig. 3. Framework of WSD Platform

The knowledge-based WSD platform includes two modules, as Fig.3.[19] The first module is to select feature words, as described in section 3.1 and 3.2. Based on different strategies, feature words are selected and assigned with appropriate weights. The second module is to select the right sense of the ambiguous word. Following with McCarthy[8], we calculate the semantic relatedness between each sense of the ambiguous word and the senses of the feature words, and choose the sense with most semantic overall relatedness as the right sense of the ambiguous word. For each sense, its relatedness score is computed with Eq.(3).

$$score(ws_i) = \sum_{f_j \in F_w} weight(f_j) \times \frac{wnss(ws_i, f_j)}{\sum_{ws_{i'} \in senses(w)} wnss(ws_{i'}, f_j)} \tag{3}$$

Where:

$$wnss(ws_i, f_j) = \max_{fs_x \in senses(f_j)} (wnss(ws_i, fs_x)) \tag{4}$$

In Eq.(3), ws_i is the *i*-th sense of the target word w, $senses(w)$ is the sense set of w, $ws_i \in senses(w)$; f_j is the *j*-th feature word of w, F_w is the set of feature words of w, $f_j \in F_w$; $weight(f_j)$ is the WSD weight of the *j*-th feature word. Eq.(4) means that when $wnss(ws_i, f_j)$ is computed, the sense of feature word that maximize the relatedness score with ws_i is selected. [8]

Dataset and Evaluation Measure

The dataset is provided by Koeling et al.[4], which is a dataset for lexical sample task and 41 words are included. The instances are from three domains, which are the sport and finance sections of Reuter Corpus and the balanced British National Corpus(BNC). There are about 100 instances for each word in a specific domain. In our experiments, we select the 3216 instances that are from BNC and have gotten sense agreement by the majority of taggers. Besides, we have converted the original sense tag with WordNet1.7.1 to WordNet3.0. [19]

The instances are parsed with Stanford Parser[18]. WN Similarity package(v2.05)[20] is utilized to compute the semanic relatedness, which can provide six similarity measures and four relatedness measures. Context Vector relatedness measure is adopted in the paper.

There are several measures to evaluate the performance of WSD, such as accuracy, recall, coverage and F-Measure. In our experiments, we select recall as evaluation measure, which is computed with Eq(5).

$$R = \frac{M}{N} \times 100\% \tag{5}$$

In Eq(5), *N* is the total number of instances that need to be disambiguated and *M* is the number of instances that are disambiguated correctly.

Experiment Results

In the experiments, we respectively utilize four kinds of different methods, that are window-based, dependency-based, PTree-based and DTree-based methods, to get feature words for the target word, and compare their performances of WSD.

Result of Window-based Method (Win)

Window-based method gets feature words based on a certain length of context window. In the experiment, the size of windows is set to *2N*. Centered with the target word, *N* content words before and *N* content words following the target words are selected as feature words, whose weights of WSD are uniformly set to 1. This experimental result is shown in Fig.4. The recall reaches the max value 38.99% when size of window is 16.

Result of Dependency-based Method (Depend)

Dependency-based method only selects the content words that have direct dependency relation as feature words. WSD weight of each of feature words is uniformly assigned to 1. The feature words are closely related with the ambiguous word, but the number of them is often few. Sometimes, the method would fail to find any feature word. In the experiment, the recall of the method is only 34.55%.

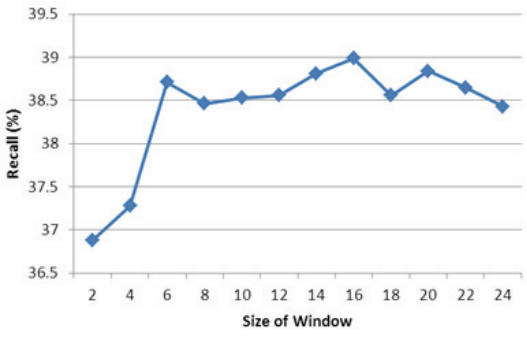

Fig. 4. Recall of Window-based Method

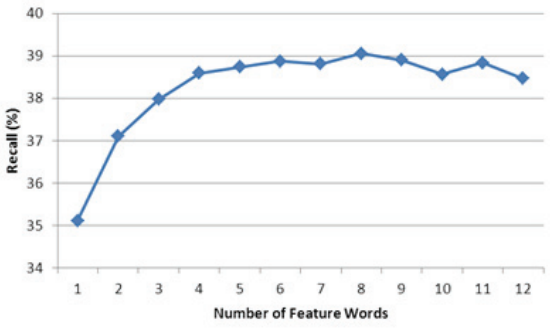

Fig. 5. Recall of Different Number of Feature Words

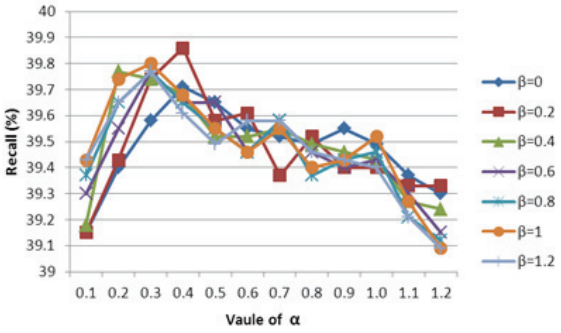

Fig. 6. Recall of PTree-based Method with n=8

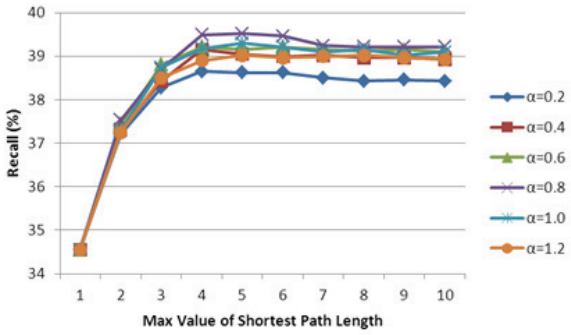

Fig. 7. Recall of DTree-based Method

Result of PTree-based Method (PTree)

As mentioned in section 3.1, PTree-based method selects adjacent words on PTree as feature words. There are three parameters that affect the effectiveness of the method, that is, the number of feature words – *n* and adjustable factors – *α, β*. Firstly, we set *α* and *β* to 0, and test the effect of different *n*. The result is shown in Fig.5, by which we can find that when *n* is 8, the best recall is achieved. Secondly, the experiments are done to determine the value of *α* and *β*. As is shown in Fig.6, the recall reaches the max value 39.86% when *n*=8, *α*=0.4 and *β*=0.2.

Result of DTree-based Method (DTree)

As mentioned in section 3.2, DTree-based method selects adjacent words on DTree as feature words. There are two parameters that affect the effectiveness of the method, that is, the max length of shortest path length – *n* and adjustable factors – *α*. The experimental result is as shown in Fig.7. The best recall achieves at 39.52% when *n*=5 and *α*=0.8. (Note: If *n* is set to 1, DTree-based method is equal with dependency-based method.).

Detailed Comparison of Different Methods

In order to compare the four methods in detail, the recall of each ambiguous word is shown in Table 1. The results demonstrate the good performance of proposed methods. Both of PTree-based and DTree-based methods are superior to window-based and dependency-based method. Compared with window-based method, the recalls of PTree-based method and DTree-based method respectively surpass it by 0.87% and 0.53%. PTree-based and DTree-based methods consider syntactic and semantic relations among words, which make them select feature words more accurately. But, window-based method simply selects the words within a certain length of window as feature words, which make it be easy to induce short-distance irrelevant noise words and omit the long-distance relevant words. Among the four methods, feature words selected by dependency-based method are most related with ambiguous words, however, the recall of the method is worst. The reason is as follow. Feature words that have direct dependency relations with the target word are often few in the sentence. In some cases, dependency-based method can't get any effective feature word. Comparing the number of instances that fail to get more than one feature word, there are 9 instances for window-based, DTree-based and PTree-based methods, 255 instances for dependency-based method. Apparently, the dependency-based method fails to get effective feature words for many instances. This leads to worse recall than other methods.

Comparing PTree-based and DTree-based method, we find that recall of the former is 0.34% higher than that of the latter. This is beyond what we expect and make us surprised. PTree reflects hierarchical constituent relationships among the words and phrases of the sentence, while DTree reflects semantic dependency relations among the words of the sentence. It seems reasonable that DTree is more suit to select feature words than PTree. But, the experimental result shows that PTree is better than DTree. The reason for this may be that the dependency parsing technology is not as mature as phrase structure parsing. In our experiments, Stanford Parser gets dependency parsing tree based on phrase structure parsing tree. This makes that the accuracy of PTree must be lower than that of DTree. With the development and maturity of dependency parsing, PTree-based method is hoped to be promoted greatly.

Conclusions and Future Work

The paper proposes feature words selection methods based on PTree and DTree for WSD, and evaluates them with a public lexical sample dataset. Compared with window-based and dependency-based methods, our proposed methods select feature words based on PTree and DTree, which can avoid the problem of noise irrelevant words of window-based method, and solve the paucity of feature words of dependency-based method. Compared with PTree-based and DTree-based methods, the former is better than the latter. The results of experiments demonstrate the good effectiveness of PTree-based and DTree-based methods in the paper. Both of the methods are preferred strategies to select feature words to disambiguate the ambiguous word.

There are two aspects in the future works. On the one hand, we would pay more attention to DTree-based method. With the development of dependency parsing, DTree-based is expected to surpass PTree-based method. On the other hand, we would like to try to consider more information to improve the effectiveness of knowledge-based WSD. Besides WN semantic relatedness, word frequency and domain information also play important roles for WSD, so we would try to integrate them into knowledge-based WSD.

Table 1. Detailed Recall of Four Methods

Word \ Method	Win	Depend	PTree	DTree	Word \ Method	Win	Depend	PTree	DTree
bank	74.19	55.91	74.19	74.19	performance	30.26	26.31	30.26	31.58
bill	60.64	45.74	58.51	57.45	phase	85.19	74.07	87.65	87.65
bond	57.61	50.00	55.43	55.43	pitch	54.55	18.18	54.55	50.00
check	15.63	31.25	15.63	12.50	receiver	20.00	23.16	22.11	24.21
chip	81.61	68.97	81.61	80.46	record	8.00	16.00	9.33	13.33
club	72.60	47.95	71.23	68.49	reserve	42.11	28.95	48.68	39.47
coach	41.57	34.83	42.70	46.07	return	30.14	20.55	30.14	30.14
competition	38.64	35.23	38.64	38.64	right	36.05	34.88	36.05	37.21
conversion	27.85	26.58	31.65	26.58	running	55.88	38.24	51.47	57.35
country	7.53	11.83	4.30	4.30	score	10.59	17.65	11.76	12.94
crew	27.85	29.11	26.58	26.58	share	30.34	32.58	30.34	31.46
delivery	15.05	15.05	16.13	16.13	star	16.67	16.67	20.24	15.48
division	27.40	28.77	30.14	28.77	strike	24.68	29.87	29.87	27.27
fan	56.63	53.01	56.63	53.01	striker	5.00	6.00	5.00	5.00
fishing	58.75	57.50	57.50	57.50	target	57.53	41.10	57.53	56.16
goal	51.02	44.90	51.02	51.02	tie	23.17	30.49	29.27	29.27
half	22.83	31.52	26.09	26.09	title	18.97	31.03	29.31	25.86
level	44.00	33.33	46.67	45.33	top	33.33	33.33	36.67	41.67
manager	70.10	54.64	68.04	67.01	transfer	60.00	42.86	58.57	55.71
market	9.84	11.48	13.11	19.67	will	53.06	30.61	51.02	51.02
package	37.21	33.72	37.21	37.21	**Average**	**38.99**	**34.55**	**39.86**	**39.52**

Acknowledgment

This work is supported by research fund of "The Theory and Method of Intelligent Processing for Mass Language Information" and "Research on Coreference Resolution for Chinese Text" of Beijing Institute of Technology and Primary Funds of National Defense. Besides, we would like to thank greatly to Rob Koeling and Diana McCarthy for kindly providing the assistance, and thank Siddharth Patwardhan and Ted Pedersen for WN Similarity package.

REFERENCES

[1] P. Resnik. WSD in NLP Applications, in Word Sense Disambiguation : Algorithms and applications, (Springer, Berlin/Heidelberg, 2007), 299-337.

[2] J. R. Firth. A synopsis of linguistic theory 1930-55. London: The Philological Society, (1957).

[3] P. Jin. Researches on Some Key Issues of Word Sense Disambiguation. PhD dissertation. Peking University, (2009).

[4] R. Koeling, D. McCarthy and J. Carroll. Domain-Specific Sense Distributions and Predominant Sense Acquisition. In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP). ACL, (2005), 419-426.

[5] E. Agirre and O. L. d. Lacalle. On Robustness and Domain Adaptation using SVD for Word Sense Disambiguation. In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008). ACL, (2008), 17-24.

[6] T. Pedersen. Unsupervised Corpus-based Methods for WSD, in Word Sense Disambiguation : Algorithms and applications, (Springer Verlag, Berlin/Heidelberg, 2007), 133-166.

[7] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd Annual Meeting on Association For Computational Linguistics. ACL, (1995), 189-196.

[8] D. McCarthy, R. Koeling, JulieWeeds and J. Carroll. Unsupervised Acquisition of Predominant Word Senses. Computational Linguistics, 33 (2007)4, 553-590.

[9] E. Agirre, O. L. d. Lacalle and A. Soroa. Knowledge-based WSD and specific domains: performing over supervised WSD. In Proceedings of the International Joint Conference on Artificial Intelligence 2009. AAAI Press, (2009), 1501-1506.

[10] S. Patwardhan, S. Banerjee and T. Pedersen. Using measures of semantic relatedness for word sense disambiguation. In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics. Springer, (2003), 241-257.

[11] S. Patwardhan, S. Banerjee and T. Pedersen. UMND1: Unsupervised Word Sense Disambiguation Using Contextual Semantic Relatedness. In Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007). ACL, (2007), 390-393.

[12] T. Pedersen and V. Kolhatkar. WordNet::SenseRelate::AllWords -A Broad CoverageWord Sense Tagger that Maximizes Semantic Relatedness. In Proceedings of NAACL HLT 2009. ACL, (2009), 17-20.

[13] D. Lin. Automatic retrieval and clustering of similar words. In Proceedings of COLING-ACL 98. ACL, (1998), pp. 768-774.

[14] Z. Lu, G. Zhang and S. Li. Word Sense Disambiguation Based on Dependency Relationship Analysis and Bayes Model. High Technology Letters, 13 (2003)5, 1-7.

[15] P. Chen, W. Ding, C. Bowes and D. Brown. A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL. ACL, (2009), 28-36.

[16] N. Chomsky. Three models for the description of language. Institute of Radio Engineers Transactions on Information Theory 2, (1956)113-124.

[17] L. Tesnière. Éléments de syntaxe structurale. Paris: Klincksieck, (1959).

[18] M.-C. d. Marneffe, B. MacCartney and C. D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. In Proceedings of 5th International Conference on Language Resources and Evaluation (LREC 2006). ELRA, (2006), 449-454.

[19] H. Huang and W. Lu. Knowledge-based Word Sense Disambiguation with Feature Words Based on Dependency Relation and Syntax Tree. International Journal of Advancements in Computing Technology, 3 (2011)8, 73-81.

[20] WordNet::Similarity, (Accessed at http://wn-similarity.sourceforge.net/).

***Authors**: Wenpeng LU, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China, E-mail: luwpeng@bit.edu.cn; Besides, Wenpeng LU is a lecturer of School of Science, Shandong Polytechnic University, China; prof. dr Heyan HUANG, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China, E-mail: hhy63@bit.edu.cn; Chaoyong ZHU, School of Computer Sciencve and Technology, University of Science &Technology of China, China 230027, E-mail: zhuxiao1@mail.ustc.edu.cn*