A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge

Ping Chen
Dept. of Computer and Math. Sciences
University of Houston-Downtown
chenp@uhd.edu

Wei Ding
Department of Computer Science
University of Massachusetts-Boston
ding@cs.umb.edu

Chris Bowes
Dept. of Computer and Math. Sciences
University of Houston-Downtown
bowesc@uhd.edu

David Brown
Dept. of Computer and Math. Sciences
University of Houston-Downtown
brownd@uhd.edu

Abstract

Word sense disambiguation is the process of determining which sense of a word is used in a given context. Due to its importance in understanding semantics of natural languages, word sense disambiguation has been extensively studied in Computational Linguistics. However, existing methods either are brittle and narrowly focus on specific topics or words, or provide only mediocre performance in real-world settings. Broad coverage and disambiguation quality are critical for a word sense disambiguation system. In this paper we present a fully unsupervised word sense disambiguation method that requires only a dictionary and unannotated text as input. Such an automatic approach overcomes the problem of brittleness suffered in many existing methods and makes broad-coverage word sense disambiguation feasible in practice. We evaluated our approach using SemEval 2007 Task 7 (Coarse-grained English All-words Task), and our system significantly outperformed the best unsupervised system participating in SemEval 2007 and achieved the performance approaching top-performing supervised systems. Although our method was only tested with coarse-grained sense disambiguation, it can be directly applied to fine-grained sense disambiguation.

1 Introduction

In many natural languages, a word can represent multiple meanings/senses, and such a word is called a homograph. Word sense disambiguation(WSD) is the process of determining which sense of a homograph is used in a given context. WSD is a long-standing problem in Computational Linguistics, and has significant impact in many real-world applications including machine translation, information extraction, and information retrieval. Generally, WSD methods use the context of a word for its sense disambiguation, and the context information can come from either annotated/unannotated text or other knowledge resources, such as WordNet (Fellbaum, 1998), SemCor (SemCor, 2008), Open Mind Word Expert (Chklovski and Mihalcea, 2002), eXtended WordNet (Moldovan and Rus, 2001), Wikipedia (Mihalcea, 2007), parallel corpora (Ng, Wang, and Chan, 2003). In (Ide and Véronis, 1998) many different WSD approaches were described. Usually, WSD techniques can be divided into four categories (Agirre and Edmonds, 2006),

- Dictionary and knowledge based methods. These methods use lexical knowledge bases such as dictionaries and thesauri, and hypothesize that context knowledge can be extracted from definitions of words. For example, Lesk disambiguated two words by finding the pair of senses with the greatest word overlap in their dictionary definitions (Lesk, 1986).

- Supervised methods. Supervised methods mainly adopt context to disambiguate words. A supervised method includes a training phase and a testing phase. In the training phase, a sense-annotated training corpus is required, from which syntactic and semantic features are extracted to create a classifier using machine

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 28–36,
Boulder, Colorado, June 2009.

learning techniques, such as Support Vector Machine (Novischi et al., 2007). In the following testing phase, a word is classified into senses (Mihalcea, 2002) (Ng and Lee, 1996). Currently supervised methods achieve the best disambiguation quality (about 80% precision and recall for coarse-grained WSD in the most recent WSD evaluation conference SemEval 2007 (Navigli et al., 2007)). Nevertheless, since training corpora are manually annotated and expensive, supervised methods are often brittle due to data scarcity, and it is hard to annotate and acquire sufficient contextual information for every sense of a large number of words existing in natural languages.

- Semi-supervised methods. To overcome the knowledge acquisition bottleneck problem suffered by supervised methods, these methods make use of a small annotated corpus as seed data in a bootstrapping process (Hearst, 1991) (Yarowsky, 1995). A word-aligned bilingual corpus can also serve as seed data (Ng, Wang, and Chan, 2003).

- Unsupervised methods. These methods acquire contextual information directly from unannotated raw text, and senses can be induced from text using some similarity measure (Lin, 1997). However, automatically acquired information is often noisy or even erroneous. In the most recent SemEval 2007 (Navigli et al., 2007), the best unsupervised systems only achieved about 70% precision and 50% recall.

Disambiguation of a limited number of words is not hard, and necessary context information can be carefully collected and hand-crafted to achieve high disambiguation accuracy as shown in (Yarowsky, 1995). However, such approaches suffer a significant performance drop in practice when domain or vocabulary is not limited. Such a "cliff-style" performance collapse is called brittleness, which is due to insufficient knowledge and shared by many techniques in Artificial Intelligence. The main challenge of a WSD system is how to overcome the knowledge acquisition bottleneck and efficiently collect the huge amount of context knowledge. More precisely, a practical WSD need figure out how to create and maintain a comprehensive, dynamic, and up-to-date context knowledge base in a highly automatic manner. The context knowledge required in WSD has the following properties:

1. The context knowledge need cover a large number of words and their usage. Such a requirement of broad coverage is not trivial because a natural language usually contains thousands of words, and some popular words can have dozens of senses. For example, the Oxford English Dictionary has approximately 301,100 main entries (Oxford, 2003), and the average polysemy of the WordNet inventory is 6.18 (Fellbaum, 1998). Clearly acquisition of such a huge amount of knowledge can only be achieved with automatic techniques.

2. Natural language is not a static phenomenon. New usage of existing words emerges, which creates new senses. New words are created, and some words may "die" over time. It is estimated that every year around 2,500 new words appear in English (Kister, 1992). Such dynamics requires a timely maintenance and updating of context knowledge base, which makes manual collection even more impractical.

Taking into consideration the large amount and dynamic nature of context knowledge, we only have limited options when choosing knowledge sources for WSD. WSD is often an unconscious process to human beings. With a dictionary and sample sentences/phrases an average educated person can correctly disambiguate most polysemous words. Inspired by human WSD process, we choose an electronic dictionary and unannotated text samples of word instances as context knowledge sources for our WSD system. Both sources can be automatically accessed, provide an excellent coverage of word meanings and usage, and are actively updated to reflect the current state of languages. In this paper we present a fully unsupervised WSD system, which only requires WordNet sense inventory and unannotated text. In the rest of this paper, section 2 describes how to acquire and represent the context knowledge for WSD. We present our WSD algorithm in section 3. Our WSD system is evaluated with SemEval-2007 Task 7 (Coarse-grained English

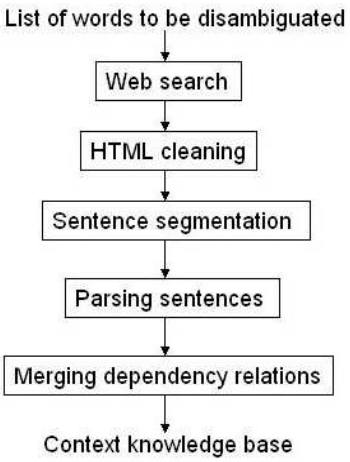

Figure 1: Context Knowledge Acquisition and Representation Process

All-words Task) data set, and the experiment results are discussed in section 4. We conclude in section 5.

2 Context Knowledge Acquisition and Representation

Figure 1 shows an overview of our context knowledge acquisition process, and collected knowledge is saved in a local knowledge base. Here are some details about each step.

2.1 Corpus building through Web search

The goal of this step is to collect as many as possible valid sample sentences containing the instances of to-be-disambiguated words. Preferably these instances are also diverse and cover many senses of a word. We have considered two possible text sources,

1. Electronic text collection, e.g., Gutenberg project (Gutenberg, 1971). Such collections often include thousands of books, which are often written by professionals and can provide many valid and accurate usage of a large number of words. Nevertheless, books in these collections are usually copyright-free and old, hence are lack of new words or new senses of words used in modern English.

2. Web documents. Billions of documents exist in the World Wide Web, and millions of Web pages are created and updated everyday. Such a huge dynamic text collection is an ideal source to provide broad and up-to-date context knowledge for WSD. The major concern about Web documents is inconsistency of their quality, and many Web pages are spam or contain erroneous information. However, factual errors in Web pages will not hurt the performance of WSD. Nevertheless, the quality of context knowledge is affected by broken sentences of poor linguistic quality and invalid word usage, e.g., sentences like "Colorless green ideas sleep furiously" that violate commonsense knowledge. Based on our experience these kind of errors are negligible when using popular Web search engines to retrieve relevant Web pages.

To start the acquisition process, words that need to be disambiguated are compiled and saved in a text file. Each single word is submitted to a Web search engine as a query. Several search engines provide API's for research communities to automatically retrieve large number of Web pages. In our experiments we used both Google and Yahoo! API's to retrieve up to 1,000 Web pages for each to-be-disambiguated word. Collected Web pages are cleaned first, e.g., control characters and HTML tags are removed. Then sentences are segmented simply based on punctuation (e.g., ?, !, .). Sentences that contain the instances of a specific word are extracted and saved into a local repository.

2.2 Parsing

Sentences organized according to each word are sent to a dependency parser, Minipar. Dependency parsers have been widely used in Computational Linguistics and natural language processing. An evaluation with the SUSANNE corpus shows that Minipar achieves 89% precision with respect to dependency relations (Lin, 1998). After parsing sentences are converted to parsing trees and saved in files. Neither our simple sentence segmentation approach nor Minipar parsing is 100% accurate, so a small number of invalid dependency relations may exist in parsing trees. The impact of these erroneous relations will be minimized in our WSD algorithm. Comparing with tagging or chunking, parsing is relatively expensive and time-consuming. However, in our method parsing is not performed in real time when we disambiguate words. Instead, sentences

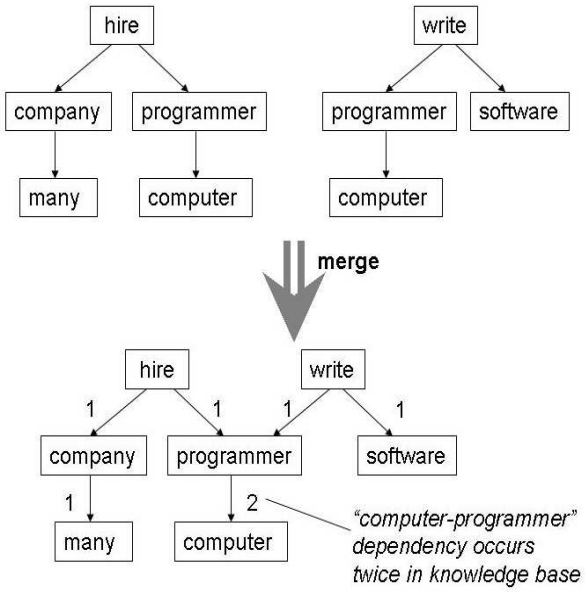

Figure 2: Merging two parsing trees. The number beside each edge is the number of occurrences of this dependency relation existing in the context knowledge base.

are parsed only once to extract dependency relations, then these relations are merged and saved in a local knowledge base for the following disambiguation. Hence, parsing will not affect the speed of disambiguation at all.

2.3 Merging dependency relations

After parsing, dependency relations from different sentences are merged and saved in a context knowledge base. The merging process is straightforward. A dependency relation includes one head word/node and one dependent word/node. Nodes from different dependency relations are merged into one as long as they represent the same word. An example is shown in Figure 2, which merges the following two sentences:

"Computer programmers write software."

"Many companies hire computer programmers."

In a dependency relation "$word_1 \rightarrow word_2$", $word_1$ is the head word, and $word_2$ is the dependent word. After merging dependency relations, we will obtain a weighted directed graph with a word as a node, a dependency relation as an edge, and the number of occurrences of dependency relation as weight of an edge. This weight indicates the strength of semantic relevancy of head word and dependent word. This graph will be used in the following WSD process as our context knowledge base. As a fully automatic knowledge acquisition process, it is inevitable to include erroneous dependency relations in the knowledge base. However, since in a large text collection valid dependency relations tend to repeat far more times than invalid ones, these erroneous edges only have minimal impact on the disambiguation quality as shown in our evaluation results.

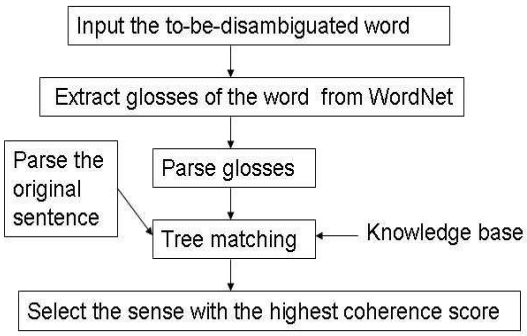

Figure 3: WSD Procedure

3 WSD Algorithm

Our WSD approach is based on the following insight:

If a word is semantically coherent with its context, then at least one sense of this word is semantically coherent with its context.

Assume that the text to be disambiguated is semantically valid, if we replace a word with its glosses one by one, the correct sense should be the one that will maximize the semantic coherence within this word's context. Based on this idea we set up our WSD procedure as shown in Figure 3. First both the original sentence that contains the to-be-disambiguated word and the glosses of to-be-disambiguated word are parsed. Then the parsing tree generated from each gloss is matched with the parsing tree of original sentence one by one. The gloss most semantically coherent with the original sentence will be chosen as the correct sense. How to measure the semantic coherence is critical. Our idea is based on the following hypotheses (assume $word_1$ is the to-be-disambiguated word):

- In a sentence if $word_1$ is dependent on $word_2$, and we denote the gloss of the correct sense of $word_1$ as g_{1i}, then g_{1i} contains the most semantically coherent words that are dependent

on $word_2$;

- In a sentence if a set of words DEP_1 are dependent on $word_1$, and we denote the gloss of the correct sense of $word_1$ as g_{1i}, then g_{1i} contains the most semantically coherent words that DEP_1 are dependent on.

For example, we try to disambiguate "company" in "A large company hires many computer programmers", after parsing we obtain the dependency relations "hire → company" and "company → large". The correct sense for the word "company" should be "an institution created to conduct business". If in the context knowledge base there exist the dependency relations "hire → institution" or "institution → large", then we believe that the gloss "an institution created to conduct business" is semantically coherent with its context - the original sentence. The gloss with the highest semantic coherence will be chosen as the correct sense. Obviously, the size of context knowledge base has a positive impact on the disambiguation quality, which is also verified in our experiments (see Section 4.2). Figure 4 shows our detailed WSD algorithm. Semantic coherence score is generated by the function $TreeMatching$, and we adopt a sentence as the context of a word.

We illustrate our WSD algorithm through an example. Assume we try to disambiguate "company" in the sentence "A large software company hires many computer programmers". "company" has 9 senses as a noun in WordNet 2.1. Let's pick the following two glosses to go through our WSD process.

- an institution created to conduct business
- small military unit

First we parse the original sentence and two glosses, and get three weighted parsing trees as shown in Figure 5. All weights are assigned to nodes/words in these parsing trees. In the parsing tree of the original sentence the weight of a node is reciprocal of the distance between this node and to-be-disambiguated node "company" (line 12 in Figure 4). In the parsing tree of a gloss the weight of a node is reciprocal of the level of this node in the parsing tree (line 16 in Figure 4). Assume that our context knowledge base contains relevant dependency relations shown in Figure 6.

Input: Glosses from WordNet;
S: the sentence to be disambiguated;
G: the knowledge base generated in Section 2;

1. Input a sentence S, W = $\{w|$ w's part of speech is noun, verb, adjective, or adverb, $w \in S\}$;
2. Parse S with a dependency parser, generate parsing tree T_S;
3. For each $w \in W$ {
4. Input all w's glosses from WordNet;
5. For each gloss w_i {
6. Parse w_i, get a parsing tree T_{wi};
7. score = TreeMatching(T_S, T_{wi});
   }
8. If the highest score is larger than a preset threshold, choose the sense with the highest score as the correct sense;
9. Otherwise, choose the first sense.
10. }

TreeMatching(T_S, T_{wi})
11. For each node $n_{Si} \in T_S$ {
12. Assign weight $w_{Si} = \frac{1}{l_{Si}}$, l_{Si} is the length between n_{Si} and w_i in T_S;
13. }
14. For each node $n_{wi} \in T_{wi}$ {
15. Load its dependent words D_{wi} from G;
16. Assign weight $w_{wi} = \frac{1}{l_{wi}}$, l_{wi} is the level number of n_{wi} in T_{wi};
17. For each n_{Sj} {
18. If $n_{Sj} \in D_{wi}$
19. calculate connection strength s_{ji} between n_{Sj} and n_{wi};
20. score = score + $w_{Si} \times w_{wi} \times s_{ji}$;
21. }
22. }
23. Return score;

Figure 4: WSD Algorithm

The weights in the context knowledge base are assigned to dependency relation edges. These weights are normalized to [0, 1] based on the number of dependency relation instances obtained in the acquisition and merging process. A large number of occurrences will be normalized to a high value (close to 1), and a small number of occurrences will be nor-

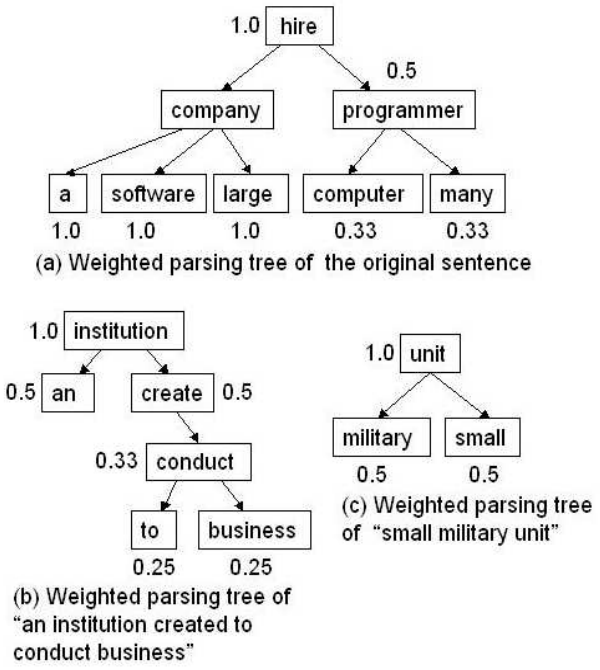

Figure 5: Weighted parsing trees of the original sentence and two glosses of "company"

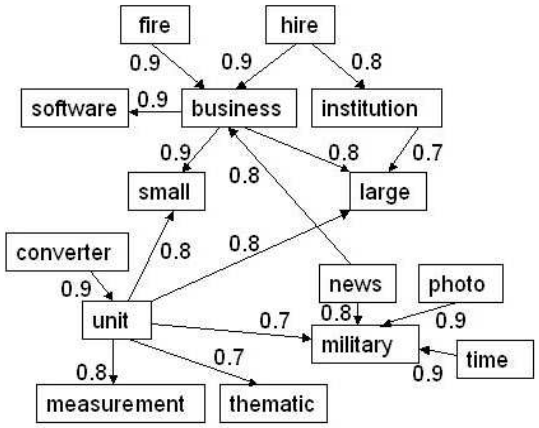

Figure 6: A fragment of context knowledge base

malized to a low value (close to 0).

Now we load the dependent words of each word in gloss 1 from the knowledge base (line 14, 15 in Figure 4), and we get {small, large} for "institution" and {large, software} for "business". In the dependent words of "company", "large" belongs to the dependent word sets of "institution" and "business", and "software" belongs to the dependent word set of "business", so the coherence score of gloss 1 is calculated as (line 19, 20 in Figure 4):

$1.0 \times 1.0 \times 0.7 + 1.0 \times 0.25 \times 0.8 + 1.0 \times 0.25 \times 0.9 = 1.125$

We go through the same process with the second gloss "small military unit". "Large" is the only dependent word of "company" appearing in the dependent word set of "unit" in gloss 2, so the coherence score of gloss 2 in the current context is:

$1.0 \times 1.0 \times 0.8 = 0.8$

After comparing the coherence scores of two glosses, we choose sense 1 of "company" as the correct sense (line 9 in Figure 4). This example illustrates that a strong dependency relation between a head word and a dependent word has a powerful disambiguation capability, and disambiguation quality is also significantly affected by the quality of dictionary definitions.

In Figure 4 the $TreeMatching$ function matches the dependent words of to-be-disambiguated word (line 15 in Figure 4), and we call this matching strategy as dependency matching. This strategy will not work if a to-be-disambiguated word has no dependent words at all, for example, when the word "company" in "Companies hire computer programmers" has no dependent words. In this case, we developed the second matching strategy, which is to match the head words that the to-be-disambiguated word is dependent on, such as matching "hire" (the head word of "company") in Figure 5(a). Using the dependency relation "hire → company", we can correctly choose sense 1 since there is no such relation as "hire → unit" in the knowledge base. This strategy is also helpful when disambiguating adjectives and adverbs since they usually only depend on other words, and rarely any other words are dependent on them. The third matching strategy is to consider synonyms as a match besides the exact matching words. Synonyms can be obtained through the synsets in WordNet. For example, when we disambiguate "company" in "Big companies hire many computer programmers", "big" can be considered as a match for "large". We call this matching strategy as synonym matching. The three matching strategies can be combined and applied together, and in Section 4.1 we show the experiment results of 5 different matching strategy combinations.

4 Experiments

We have evaluated our method using SemEval-2007 Task 07 (Coarse-grained English All-words Task) test set (Navigli et al., 2007). The task organizers provide a coarse-grained sense inventory created with SSI algorithm (Navigli and Velardi, 2005), training data, and test data. Since our method does not need any training or special tuning, neither coarse-grained sense inventory nor training data was used. The test data includes: a news article about "homeless" (including totally 951 words, 368 words are annotated and need to be disambiguated), a review of the book "Feeding Frenzy" (including totally 987 words, 379 words are annotated and need to be disambiguated), an article about some traveling experience in France (including totally 1311 words, 500 words are annotated and need to be disambiguated), computer programming(including totally 1326 words, 677 words are annotated and need to be disambiguated), and a biography of the painter Masaccio (including totally 802 words, 345 words are annotated and need to be disambiguated). Two authors of (Navigli et al., 2007) independently and manually annotated part of the test set (710 word instances), and the pairwise agreement was 93.80%. This inter-annotator agreement is usually considered an upper-bound for WSD systems.

We followed the WSD process described in Section 2 and 3 using the WordNet 2.1 sense repository that is adopted by SemEval-2007 Task 07. All experiments were performed on a Pentium 2.33GHz dual core PC with 3GB memory. Among the 2269 to-be-disambiguated words in the five test documents, 1112 words are unique and submitted to Google API as queries. The retrieved Web pages were cleaned, and 1945189 relevant sentences were extracted. On average 1749 sentences were obtained for each word. The Web page retrieval step took 3 days, and the cleaning step took 2 days. Parsing was very time-consuming and took 11 days. The merging step took 3 days. Disambiguation of 2269 words in the 5 test articles took 4 hours. All these steps can be parallelized and run on multiple computers, and the whole process will be shortened accordingly.

The overall disambiguation results are shown in Table 1. For comparison we also listed the results of the top three systems and three unsupervised systems participating in SemEval-2007 Task 07. All of the top three systems (UoR-SSI, NUS-PT, NUS-ML) are supervised systems, which used annotated resources (e.g., SemCor, Defense Science Organization Corpus) during the training phase. Our fully unsupervised WSD system significantly outperforms the three unsupervised systems (SUSSZ-FR, SUSSX-C-WD, SUSSX-CR) and achieves performance approaching the top-performing supervised WSD systems.

4.1 Impact of different matching strategies to disambiguation quality

To test the effectiveness of different matching strategies discussed in Section 3, we performed some additional experiments. Table 2 shows the disambiguation results by each individual document with the following 5 matching strategies:

1. Dependency matching only.
2. Dependency and backward matching.
3. Dependency and synonym backward matching.
4. Dependency and synonym dependency matching.
5. Dependency, backward, synonym backward, and synonym dependency matching.

As expected combination of more matching strategies results in higher disambiguation quality. By analyzing the scoring details, we verified that backward matching is especially useful to disambiguate adjectives and adverbs. Adjectives and adverbs are often dependent words, so dependency matching itself rarely finds any matched words. Since synonyms are semantically equivalent, it is reasonable that synonym matching can also improve disambiguation performance.

4.2 Impact of knowledge base size to disambiguation quality

To test the impact of knowledge base size to disambiguation quality we randomly selected 1339264 sentences (about two thirds of all sentences) from our text collection and built a smaller knowledge base. Table 3 shows the experiment results. Overall disambiguation quality has dropped slightly, which

System	Attempted	Precision	Recall	F1
UoR-SSI	100.0	83.21	83.21	83.21
NUS-PT	100.0	82.50	82.50	82.50
NUS-ML	100.0	81.58	81.58	81.58
TreeMatch	**100.0**	**73.65**	**73.65**	**73.65**
SUSSZ-FR	72.8	71.73	52.23	60.44
SUSSX-C-WD	72.8	54.54	39.71	45.96
SUSSX-CR	72.8	54.30	39.53	45.75

Table 1: Overall disambiguation scores (Our system "TreeMatch" is marked in bold)

Matching strategy	d001		d002		d003		d004		d005		Overall	
	P	R	P	R	P	R	P	R	P	R	P	R
1	72.28	72.28	66.23	66.23	63.20	63.20	66.47	66.47	56.52	56.52	65.14	65.14
2	70.65	70.65	70.98	70.98	65.20	65.20	72.23	72.23	58.84	58.84	68.18	68.18
3	79.89	79.89	75.20	75.20	69.00	69.00	71.94	71.94	64.64	64.64	72.01	72.01
4	80.71	80.71	78.10	78.10	72.80	72.80	71.05	71.05	67.54	67.54	73.65	73.65
5	80.16	80.16	78.10	78.10	69.40	69.40	72.82	72.82	66.09	66.09	73.12	73.12

Table 2: Disambiguation scores by article with 5 matching strategies

shows a positive correlation between the amount of context knowledge and disambiguation quality. It is reasonable to assume that our disambiguation performance can be improved further by collecting and incorporating more context knowledge.

Matching strategy	Overall	
	P	R
1	65.36	65.36
2	67.78	67.78
3	68.09	68.09
4	70.69	70.69
5	67.78	67.78

Table 3: Disambiguation scores by article with a smaller knowledge base

5 Conclusion and Future Work

Broad coverage and disambiguation quality are critical for WSD techniques to be adopted in practice. This paper proposed a fully unsupervised WSD method. We have evaluated our approach with SemEval-2007 Task 7 (Coarse-grained English All-words Task) data set, and we achieved F-scores approaching the top performing supervised WSD systems. By using widely available unannotated text and a fully unsupervised disambiguation approach, our method may provide a viable solution to the problem of WSD. The future work includes:

1. Continue to build the knowledge base, enlarge the coverage and improve the system performance. The experiment results in Section 4.2 clearly show that more word instances can improve the disambiguation accuracy and recall scores;

2. WSD is often an unconscious process for human beings. It is unlikely that a reader examines all surrounding words when determining the sense of a word, which calls for a smarter and more selective matching strategy than what we have tried in Section 4.1;

3. Test our WSD system on fine-grained SemEval 2007 WSD task 17. Although we only evaluated our approach with coarse-grained senses, our method can be directly applied to fine-grained WSD without any modifications.

Acknowledgments

This work is partially funded by NSF grant 0737408 and Scholar Academy at the University of Houston Downtown. This paper contains proprietary information protected under a pending U.S. patent.

References

Agirre, Eneko, Philip Edmonds (eds.). 2006. Word Sense Disambiguation: Algorithms and Applications, Springer.

Chklovski, T. and Mihalcea, R. 2002. Building a sense tagged corpus with open mind word expert. In Proceedings of the Acl-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions, Morristown, NJ, 116-122.

C. Fellbaum, WordNet: An Electronic Lexical Database, MIT press, 1998

Project Gutenberg, available at www.gutenberg.org

Hearst, M. (1991) Noun Homograph Disambiguation Using Local Context in Large Text Corpora, Proc. 7th Annual Conference of the University of Waterloo Center for the New OED and Text Research, Oxford.

Nancy Ide and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Comput. Linguist.*, 24(1):2–40.

Kister, Ken. "Dictionaries defined", Library Journal, Vol. 117 Issue 11, p43, 4p, 2bw

Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In Proceedings of the 5th Annual international Conference on Systems Documentation (Toronto, Ontario, Canada). V. DeBuys, Ed. SIGDOC '86.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proceedings of the LREC Workshop on the Evaluation of Parsing Systems*, pages 234–241, Granada, Spain.

Lin, D. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In Proceedings of the 35th Annual Meeting of the Association For Computational Linguistics and Eighth Conference of the European Chapter of the Association For Computational Linguistics (Madrid, Spain, July 07 - 12, 1997).

Rada Mihalcea, Using Wikipedia for Automatic Word Sense Disambiguation, in Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2007), Rochester, April 2007.

Rada Mihalcea. 2002. Instance based learning with automatic feature selection applied to word sense disambiguation. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ.

Dan Moldovan and Vasile Rus, Explaining Answers with Extended WordNet, ACL 2001.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 30–35, Prague, Czech Republic.

Roberto Navigli and Paola Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 27(7):10631074.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. Exploiting Parallel Texts for Word Sense Disambiguation: An Empirical Study. ACL, 2003.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 40–47, Morristown, NJ.

Adrian Novischi, Muirathnam Srikanth, and Andrew Bennett. 2007. Lcc-wsd: System description for English coarse grained all words task at semeval 2007. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 223–226, Prague, Czech Republic.

Catherine Soanes and Angus Stevenson, editors. 2003. Oxford Dictionary of English. Oxford University Press.

Rada Mihalcea, available at http://www.cs.unt.edu/ rada/downloads.html

Yarowsky, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd Annual Meeting on Association For Computational Linguistics (Cambridge, Massachusetts, June 26 - 30, 1995).