Word Sense Disambiguation with Automatically Acquired Knowledge

Ping Chen, Wei Ding, Max Choly, Chris Bowes

***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 and many real-world applications, word sense disambiguation has been extensively studied in Natural Language Processing and Computational Linguistics. However, existing methods either narrowly focus on a few specific words due to their reliance on expensive manually annotated training text, or give only mediocre performance in real-world settings. Broad coverage and disambiguation quality are critical for real-world natural language processing applications. In this paper we present a fully automatic disambiguation method that utilizes two readily available knowledge sources: a dictionary and knowledge extracted from unannotated text. Such an automatic approach overcomes the knowledge acquisition bottleneck and makes broad-coverage word sense disambiguation feasible in practice. Evaluated with two large scale WSD evaluation corpora, our system significantly outperforms the best unsupervised system and achieves the similar performance as the top-performing supervised systems.**

***Index Terms*—Natural Language Processing, Knowledge Acquisition, Word Sense Disambiguation**

I. Introduction

In natural languages, a word often represents multiple meanings, and each meaning is called a sense of this word. Word sense disambiguation (WSD) is the process of determining which sense of a word should be adopted in given context. WSD is a long-standing problem in Natural Language Processing (NLP) and Computational Linguistics (CL), and has broad impact on many important Natural Language Processing applications, such as Machine Translation, Information Extraction, and Information Retrieval. However, although a lot of WSD methods have been proposed, it is generally recognized that explicit WSD is rarely applied in any real world applications due to mediocre performance and insufficient coverage of existing WSD systems [11].

When disambiguating a limited number of preselected words, necessary knowledge can be carefully compiled to achieve very high disambiguation precision as shown in [16]. However, such approaches designed in lab setting suffer a significant performance drop in practice when domain or vocabulary is unlimited and manual knowledge acquisition becomes prohibitively expensive. The problem of WSD is knowledge intensive by nature, and many knowledge sources have been investigated ranging from manually sense-annotated text, raw text, thesaurus, to lexical knowledge base (LKB), e.g., WordNet, SemCor, Open Mind Word Expert, eXtended WordNet, Wikipedia, parallel corpora. As the role of knowledge is generally recognized in WSD, in [1] ten knowledge types used in WSD are discussed including collocation, semantic word associations, frequency of senses, semantic roles, syntactic cues, pragmatics, etc. However, identification of disambiguation-enabling knowledge types is only one side of the story, and to build a practical WSD system knowledge also needs to be efficiently acquired at a large scale. In general, knowledge used in a practical WSD system need satisfy the following criteria:

1) **Disambiguation-enabling.** Obviously useful WSD knowledge should be capable of disambiguating senses. Identification of such knowledge is still a very active research topic, and new knowledge is constantly being proposed and examined.
2) **Comprehensive and automatically acquirable.** The disambiguation knowledge need cover a large number of words and their various usage. Such a requirement is not easily satisfied since a natural language usually contains thousands of words, and some words can have dozens of senses. For example, the Oxford English Dictionary has approximately 301,100 entries, and the average polysemy of WordNet inventory is 6.18. Obviously, such a large-scale knowledge acquisition can only be achieved with automatic techniques.
3) **Dynamic and up to date.** A 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. Such dynamics requires constant and timely maintenance and updating of WSD knowledge base, which makes any manual interference (e.g., sense annotation and supervised learning) even more impractical.

Taking into consideration the large amount and dynamic nature of knowledge required by WSD, there are very limited options when choosing knowledge sources for a practical WSD system. Currently identifying suitable knowledge sources still remains as an open and critical problem in WSD and also other NLP fields [8]. Dependency knowledge was applied to WSD in [5], however, its disambiguation capability was not fully exploited by direct utilization of frequency of dependency relations, and their WSD method only achieved 73% in both precision and recall that are well below the most-frequent-

Ping Chen and Chris Bowes are with the Department of Computer and Mathematics Sciences, University of Houston-Downtown, 1 Main St., Houston, TX 77002. E-mail: chenp@uhd.edu

Wei Ding and Max Choly are with the Department of Computer Science, University of Massachusetts-Boston, 100 Morrissey Blvd., Boston, MA 02125. E-mail: ding@cs.umb.edu

sense (MFS) baseline. In this paper we normalize the absolute frequency of dependencies with Pearson's χ^2 test (details are given in Section III-B), and together with coherent fusion of three knowledge sources our WSD system can achieve above-MFS-baseline performance that is a necessary condition for a practical WSD system. The main contributions of our work are:

1) Building a fully automatic WSD system that coherently utilizes three knowledge sources: glosses from dictionaries, the most frequent sense information, and normalized dependency knowledge extracted from unannotated text. No training materials or annotated corpus are required in our WSD method. All of three knowledge sources are disambiguation-enabling, provide a comprehensive coverage of words and their usage, and are constantly updated to reflect the current state of languages. Normalized dependency knowledge extracted from unannotated text can be efficiently collected and accessed without any manual efforts. Moreover, the knowledge is not created for the purpose of WSD, which means that no extra efforts are required for their construction or maintenance. All of these properties adhere closely to our goal of building a practical WSD system.
2) State-of-art performance. Evaluated by a large real world WSD testset (SemEval 2007 Task 07), our method achieves 82.64% in both precision and recall, which clearly outperforms the best unsupervised WSD system (about 70% in precision and 50% in recall) and performs similarly as the best supervised system (83.21% in precision and recall). It is noteworthy that our system outperforms the most-frequent-sense (MFS) baseline (78.89% in the SemEval 2007 Task 07) that simply selects the most frequent sense. To our best knowledge, our method is the only fully automatic WSD technique that performs better than MFS baseline based on systems participating in the SemEval 2007 Task 07 (please refer to Table II), and may lead off the application of WSD in real world Natural Language Processing applications. One additional experiment with Senseval-2 testing corpus further confirms the effectiveness of our approach (please refer to Table I). We want to emphasize that both experiments were performed under real world settings, which is critical to support the full development of practical software systems in the future [6].

This paper is organized as follows. Section II discusses existing WSD methods. Section III describes how to acquire and represent disambiguation-enabling knowledge. We present our WSD system in section IV. Our system is evaluated with both coarse-grained and fine-grained WSD evaluation corpora: SemEval-2007 Task 07 (Coarse-grained English All-words Task) and Senseval-2 (Fine-grained English All-words Task). The experimental results are presented and analyzed in section V. We conclude in section VI.

II. Related Work

Generally WSD techniques can be divided into four categories [1],

- Dictionary and knowledge based methods. These methods use lexical knowledge bases (LKB) such as dictionaries and thesauri, and extract knowledge from word definitions [7] and relations among words/senses. Recently, several graph-based WSD methods were proposed. In these approaches, first a graph is built with senses as nodes and relations among words/senses (e.g., synonymy, antonymy) as edges, and the relations are usually acquired from a LKB (e.g., WordNet). Then a ranking algorithm is conducted over the graph, and senses ranked the highest are assigned to the corresponding words. Different relations and ranking algorithms were experimented with these methods, such as TexRank algorithm [10], personalized PageRank algorithm [2], a two-stage searching algorithm [11], centrality algorithms [13].
- Supervised methods. 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 build a classifier using machine learning techniques, such as Support Vector Machine. In the following testing phase, the classifier picks the best sense for a word based on its surrounding words. Currently supervised methods achieved the best disambiguation quality (about 80% in precision and recall for coarse-grained WSD in the most recent WSD evaluation conference SemEval 2007 [12]). Nevertheless, since training corpora are manually annotated and expensive, supervised methods are often brittle due to data scarcity, and it is impractical to manually annotate huge number of words existing in a natural language.
- Semi-supervised methods. To overcome the knowledge acquisition bottleneck faced by supervised methods, semi-supervised methods make use of a small annotated corpus as seed data in a bootstrapping process [16]. A word-aligned bilingual corpus can also serve as seed data [17].
- Unsupervised methods. These methods acquire knowledge from unannotated raw text, and disambiguate senses using similarity measures. Unsupervised methods overcome the problem of knowledge acquisition bottleneck, but none of existing methods can outperform the most frequent sense baseline, which makes them not useful at all in practice. For example, the best unsupervised systems only achieved about 70% in precision and 50% in recall in the SemEval 2007 [12] Workshop. One recent study utilized automatically acquired dependency knowledge and achieved 73% in precision and recall [5], which are still below the most-frequent-sense baseline (78.89% in precision and recall in the SemEval 2007 Task 07).

Additionally there exist some "meta-disambiguation" methods that ensemble multiple disambiguation algorithms following the ideas of bagging or boosting in supervised learning. In [15], multiple sources were utilized to achieve optimal WSD performance. Our approach is different in that our focus is identification and ensembling of new disambiguation-enabling

and efficiently acquirable knowledge sources. In this paper we propose a new fully automatic WSD method by integrating three types of knowledge: dependency relations, glosses, and the most-frequent-sense (MFS) information. In next section we will discuss how to acquire and represent the knowledge.

III. Acquisition and Representation of Disambiguation-enabling Knowledge

Adoption of multiple knowledge sources has been utilized in some WSD systems. Since our goal is to build a practical WSD system, we only choose knowledge sources that provide broad coverage and also can be automatically acquired. Three types of knowledge are used in our WSD system: normalized dependency knowledge, glosses, and most-frequent-sense (MFS) information. Sense distribution information has proved very useful in WSD. Glosses and most-frequent-sense information can be directly accessed from Lexical Knowledge Bases (LKBs). For example, in WordNet the first sense of a word is the most frequent sense. Here is the procedure we used to acquire, merge, and normalize dependency relations:

1) Corpus building through search engines
2) Document cleaning
3) Sentence segmentation
4) Parsing
5) Dependency relation merging
6) Dependency relation normalization

The first five steps of this process are discussed in the section III-A, and the sixth step, dependency relation normalization, is discussed in III-B.

A. Dependency relation acquisition and merging

To learn about a word and its usage we need collect many valid sample sentences containing the instances of this word. Preferably these instances are also semantically diverse and cover different senses of these words. To collect a large number of word instances, we choose the World Wide Web as the knowledge source. Billions of documents are freely available 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 knowledge for WSD [3]. The major concern about Web documents is inconsistency of their quality, and many Web pages are spam or contain erroneous information. However, factual errors ("President Lincoln was born in 1967.") do not hurt the performance of our WSD method as long as they are semantically valid. Instead, to our WSD method knowledge quality is more impaired by broken sentences of poor linguistic quality and invalid word usage, e.g., sentences like "Colorless green ideas sleep furiously" that follow syntax but violate common sense knowledge, or "A chair green in the room is" that violate syntax. Based on our experience these kinds of errors are relatively rare especially when text is acquired through a high quality search engine.

First target words are sent to a Web search engine as keywords. Returned documents are parsed by a dependency parser, Minipar [9], which also provides part-of-speech (POS) information. Then dependency relations extracted from different sentences are merged and saved in a knowledge base. The merging process is straightforward. A dependency relation includes one head word/node and one dependent word/node, and nodes from different dependency relations are merged as long as they represent the same word. An example is shown in Figure 1, which merges dependency relations extracted from the following two sentences:

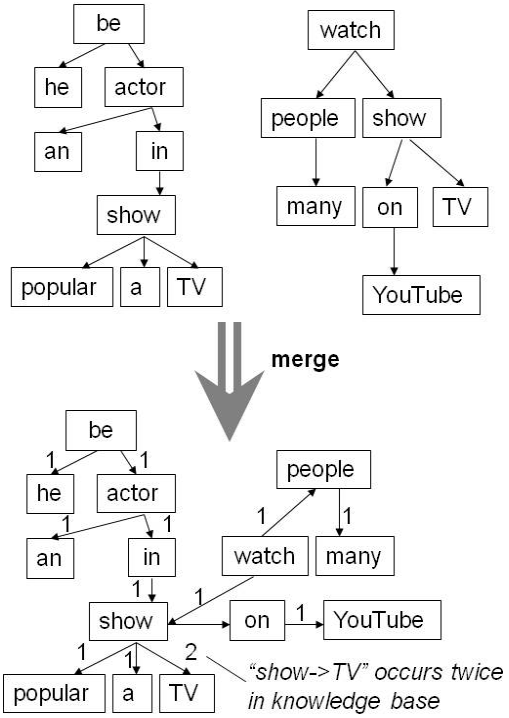

Fig. 1. Merging two parse trees. The number beside each edge is the number of occurrences of this dependency relation in the knowledge base.

"Many people watch TV shows on YouTube."

"He is an actor in a popular TV show."

After merging dependency relations, we obtain a weighted directed graph with words as nodes, dependency relations as edges, and numbers of occurrences of dependency relations as weights of edges.

B. Dependency relation normalization

Although absolute frequency of a dependency relation obtained after the merging step can reflect the semantic relatedness of head word and dependent word to a certain degree, this direct measure is inevitably distorted by the occurrence frequencies of head word and dependent word. For example, suppose that both "$wine \rightarrow red$" and "$water \rightarrow red$" occur 5 times in the knowledge base, which indicates that these two pairs of words are equally related. However, "$wine \rightarrow red$" should be a stronger connection since "water" is a more common word than "wine" and occurs more frequently. To overcome this bias, we use Pearson's χ^2 test to normalize occurrence frequency to a value within [0, 1]. Pearson's χ^2

test is efficient to check whether two random variables X and Y are independent by large samples [4].

Let n_{ij} denote the number of occurrences when $(X, Y) = (x_i, y_j)$, where $i, j = 1, 2$, χ^2 values can be calculated with a contingency table as shown below:

Y/X	X	$\neg X$	X's marginal distribution
Y	n_{11}	n_{12}	$n_{1.}$
$\neg Y$	n_{21}	n_{22}	$n_{2.}$
Y's marginal distribution	$n_{.1}$	$n_{.2}$	N

By the null hypothesis $H_0 : P(X|Y) = P(X)$, we have:

$$\chi^2 = \frac{N(n_{11}n_{22} - n_{12}n_{21})^2}{n_{1.}n_{2.}n_{.1}n_{.2}} \sim \chi^2(1)$$

Let's illustrate the calculation process through an example. Suppose from a corpus we obtain the frequency data about "red" and "water" as shown below:

Y/X	red	¬ red	total
water	5	1258	1263
¬ water	2434	768954	771388
total	2439	770212	772651

$$\chi^2 = \frac{772651 \times (5 \times 768954 - 1258 \times 2434)^2}{1263 \times 771388 \times 2439 \times 770212} = 0.259$$

Since we need $\chi^2 \gg \chi^2_\alpha(1)$ (α is the probability level), so when $\alpha = 0.85$, $\chi^2_\alpha(1) = 0.036$, and $0.259 \gg 0.036$, the connection strength of "$water \rightarrow red$" is $(1 - 0.85) = 0.15$.

Suppose from a corpus we obtain the frequency data about "wine" and "red" as shown below:

Y/X	red	¬ red	total
wine	5	125	130
¬ wine	2434	770087	772521
total	2439	770212	772651

$$\chi^2 = \frac{772651 \times (5 \times 770087 - 125 \times 2434)^2}{130 \times 772521 \times 2439 \times 770212} = 223.722$$

Given $\alpha = 0.001$, $\chi^2_\alpha(1) = 10.83$, since $\chi^2 \gg \chi^2_\alpha(1)$, the connection strength of "$wine \rightarrow red$" is $(1 - 0.001) = 0.999$.

When the number of occurrences is small, Pearson's χ^2 test becomes less useful. In our knowledge base, we assign 0 to those dependency relations whose occurrence frequencies are below a preset threshold to eliminate those unreliable connections. After the calculation of χ^2 values, this new weighted graph will be used in the following WSD process as the normalized dependency knowledge base (a sample piece is shown in Figure 5).

IV. WSD Algorithm

Our WSD system architecture is depicted in Figure 2. Figure 3 shows our detailed WSD algorithm. Two scores are calculated by functions $DepScore$ and $GlossScore$, and will be used to indicate the correct senses.

We illustrate our WSD algorithm through an example. Assume we try to disambiguate "company" in the sentence "A large company needs a sustainable business model." As a noun "company" has 9 senses in WordNet 2.1. Let's choose the following two senses to go through our WSD process:

- an institution created to conduct business
- small military unit

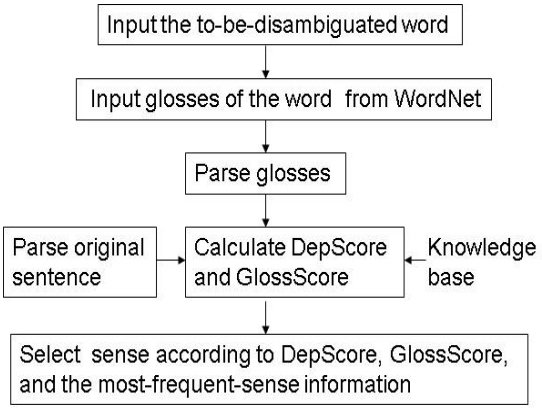

Fig. 2. WSD System Architecture

First we parse the original sentence and two glosses, and get three weighted parse trees as shown in Figure 4. Different weights are assigned to nodes/words in these parse trees. In the parse tree of the original sentence the weight of a node is reciprocal of the distance between this node and target node "company" (line 14 in the WSD algorithm shown in Figure 3). In the parse tree of a gloss the weight of a node is reciprocal of the level of this node in the parse tree (line 17 in Figure 3), and it is reasonable to assume that the higher level a word is at within a parsing tree, the more meaning this word carries comparing with the total meaning of the whole sentence.

Assume that a knowledge base contains the dependency relations shown in Figure 5. Now we load the dependent words of each word in gloss 1 from the knowledge base (line 15, 16 in Figure 3), and we get {large} for "institution" and {small, large, good} for "business". In the dependent words of "company", "large" belongs to the dependent word sets of "institution" and "business", so $score_1$ of gloss 1 based on dependencies is calculated as (lines 20, 21 in Figure 3):

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

We tried several ways to use the weights but multiplication provides the best performance.

$score_2$ of gloss 1 is generated by the overlapping words between the original sentence and gloss 1. In this example, there is only one overlapping word - "business", so $score_2$ of gloss 1 is (lines 28, 29 in Figure 3):

$0.33 \times 0.25 = 0.0825$

We go through the same process with the second gloss "small military unit". "Large" is the only dependent word of "company" in the dependent word set of "unit", so $score_1$ of gloss 2 is:

$1.0 \times 1.0 \times 0.8 = 0.8$

There are no overlapping words in the original sentence and gloss 2, so the $score_2$ of gloss 2 is 0.

Both scores generated from $DepScore$ function and $GlossScore$ function indicate that the first sense should be the right sense, so according to line 11 in the WSD algorithm

Input: Glosses from WordNet;
S: the sentence to be disambiguated;
G: the knowledge base built in Section III;
1. Input a sentence S, W = $\{w|$ w is either a noun, verb, adjective, or adverb, $w \in S\}$;
2. Parse S with a dependency parser, generate parse 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 parse tree T_{wi};
7. $score_1$ = DepScore(T_S, T_{wi});
8. $score_2$ = GlossScore(T_S, T_{wi});}
9. The sense with the highest $score_1$ is marked as $CandidateSense_1$.
10. The sense with the highest $score_2$ is marked as $CandidateSense_2$.
11. If $CandidateSense_1$ is equal to $CandidateSense_2$, choose $CandidateSense_1$;
12. Otherwise, choose the first sense.
}

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

GlossScore(T_S, T_{wi})
23. For each node $n_{Si} \in T_S$ {
24. Assign weight $w_{Si} = \frac{1}{l_{Si}}$, l_{Si} is the length between n_{Si} and w_i in T_S;}
25. For each node $n_{wi} \in T_{wi}$ {
26. Assign weight $w_{wi} = \frac{1}{l_{wi}}$, l_{wi} is the level number of n_{wi} in T_{wi};
27. For each n_{Sj} {
28. If $n_{Sj} == n_{wi}$
29. score = score + $w_{Si} \times w_{wi}$;}}
30. Return score;

Fig. 3. WSD Algorithm

we choose sense 1 of "company" as the correct sense. If $DepScore$ and $GlossScore$ point to different senses, the most frequent sense (the first sense in WordNet) will be chosen instead (line 12 in Figure 3). Apparently 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/glosses.

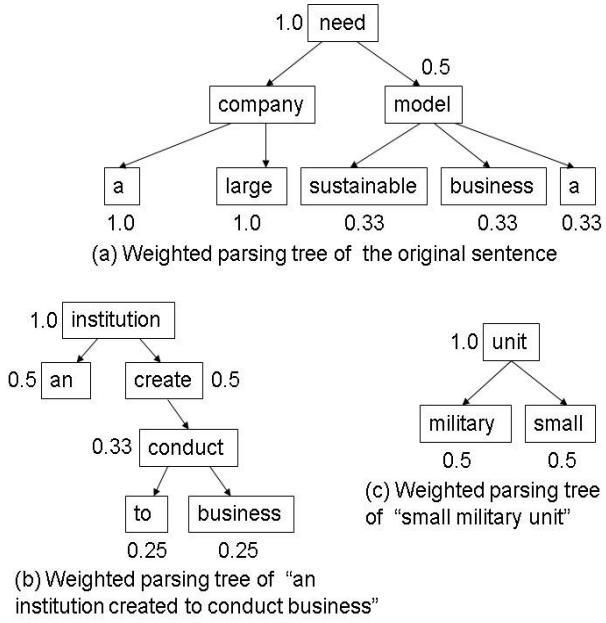

Fig. 4. Weighted parse trees of the original sentence and two glosses of "company"

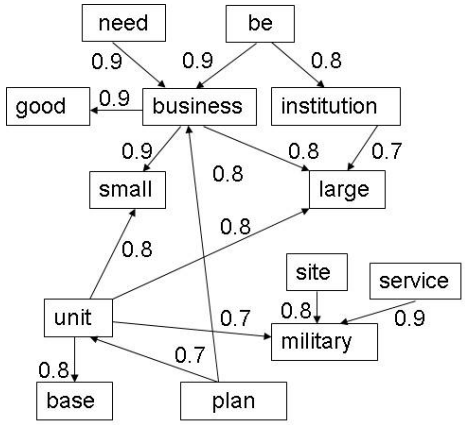

Fig. 5. A sample of normalized dependency knowledge base

In the WSD algorithm the $DepScore$ function matches the dependent words of target word (line 19 in Figure 3), and we call this matching strategy as dependency matching. This strategy will not work if a target word has no dependent words at all. In this case, we can instead match the head words that the target word is dependent on, e.g., matching "need" (the head word of "company") in Figure 4(a). Using the dependency relation "need → company", we can correctly choose sense 1 since there is no such relation as "need → unit" in the knowledge base. This strategy is especially 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 exactly matched words. Synonyms can be obtained through the synsets in WordNet. For example, when we disambiguate "company" in "A big

company needs a sustainable business model", "big" can be considered as a match for "large". We call this matching strategy as synonym matching. These three matching strategies can be combined and applied together, and [5] showed the experimental results of these matching strategies.

The $GlossScore$ function is a variant of the Lesk algorithm [7], and it is very sensitive to the words used in glosses. In a dictionary, glosses are usually very concise and include only a small number of words, so this function returns 0 in many cases and can not serve as a sufficient stand-alone disambiguation method. On the other hand, although dependency knowledge usually generates non-zero scores, dependency knowledge is noisy since a word can be a dependent of many different words and itself can mean different things, e.g., "$institution \rightarrow large$", "$family \rightarrow large$". As shown in the running example, dependency scores generated by different senses can be very close or even misleading, and due to noise dependency information only can only achieve 73.65% accuracy using SemEval 2007 Task 07 data [5]. Dependency knowledge can always point out a sense (the sense with the highest score) even it could be wrong. However, if the sense selected by dependency knowledge matches the sense selected by gloss overlapping function, it has a high probability to be correct. When both scores generated by dependency knowledge and gloss overlapping are low, most frequent sense is still the most reliable choice. With optimal combination of these three knowledge sources our method can provide broad-coverage and more accurate disambiguation that will be verified in the following experiment section.

V. Evaluation

Research on WSD not only provides valuable insights into understanding of semantics, but also can improve performance of many important Natural Language Processing applications. Recently several workshops have been organized to evaluate WSD techniques in real world settings. In this section, we will discuss our experiment results with two large scale WSD evaluation corpora, Senseval-2 fine-grained English testing corpus and SemEval 2007 Task 7 coarse-grained testing corpus. Both evaluations require the disambiguation of all nouns, verbs, adjectives, and adverbs in the testing articles, which is usually referred as "all-words" task.

A. Experiment with Senseval-2 English testing corpus

Senseval-2, the Second International Workshop on Evaluating Word Sense Disambiguation Systems, evaluated WSD systems on two types of tasks (all word or lexical sample) in 12 languages. 21 research teams participated in English all-word task [14]. In Senseval-2 testing corpus, there are totally 3 documents, which include 2473 words that need to be disambiguated. Article 1 discusses churches in England and contains 684 words that need to be disambiguated, article 2 discusses a medical discovery about genes and cancers and contains 1032 words that need to be disambiguated, and article 3 discusses children education and contains 757 words that need to be disambiguated. Table I shows our system performance along with the ten best-performing systems participated in Senseval-2. Our WSD system achieves similar performance as the best supervised system, and also outperforms MFS baseline.

System	Precision	Recall	F1 score
SMUaw (supervised)	0.69	0.69	0.69
CNTS-Antwerp (supervised)	0.636	0.636	0.636
UHD system (unsupervised)	0.633	0.633	0.633
Sinequa-LIA-HMM (supervised)	0.618	0.618	0.618
MSF baseline	0.617	0.617	0.617
UNED-AW-U2 (unsupervised)	0.575	0.569	0.572
UNED-AW-U (unsupervised)	0.556	0.55	0.553
UCLA-gchao (supervised)	0.5	0.449	0.473
UCLA-gchao2 (supervised)	0.475	0.454	0.464
UCLA-gchao3 (supervised)	0.474	0.453	0.463
DIMAP (R) (unsupervised)	0.451	0.451	0.451
DIMAP (unsupervised)	0.416	0.451	0.433

TABLE I
Comparison with top-performing systems in Senseval-2

B. Experiment with SemEval 2007 Task 7 testing corpus

To further evaluate our approach, we evaluated our WSD system using SemEval-2007 Task 07 (Coarse-grained English All-words Task) test data [12]. The task organizers provide a coarse-grained sense inventory, trial data, and test data. Since our method does not need any training or special tuning, coarse-grained sense inventory was not used. The test data includes: a news article about "homeless", a review of the book "Feeding Frenzy", an article about some traveling experience in France, an article about computer programming, and a biography of the painter Masaccio. Two authors of [12] independently annotated part of the test set (710 word instances), and the pairwise agreement was 93.80%. This inter-annotator agreement is usually considered as an upper bound for WSD systems.

We followed the WSD process described in Sections III and IV using the WordNet 2.1 sense repository. Among the 2269 target words, 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 overall disambiguation results are shown in Table II. For comparison we also listed the results of three top-performing systems and three best 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). Strictly speaking, the best performing system, UoR-SSI, does not use a supervised classifier. However, our WSD achieved similar performance using much less manually-encoded knowledge. Our fully automatic WSD system clearly outperforms the three unsupervised systems (SUSSZ-FR, SUSSX-C-WD, SUSSX-CR) and achieves performance similar as the top-performing supervised WSD systems.

It is noteworthy that our system surpasses the MFS baseline that has proved very hard to beat in many WSD evaluations. Apparently any WSD techniques that perform worse than MFS baseline will have little use in practice. Due to the noise,

System	Precision	Recall	F1 score
UoR-SSI (supervised)	83.21	83.21	83.21
UHD system (unsupervised)	**82.64**	**82.64**	**82.64**
NUS-PT (supervised)	82.50	82.50	82.50
NUS-ML (supervised)	81.58	81.58	81.58
MFS Baseline	78.89	78.89	78.89
SUSSZ-FR (unsupervised)	71.73	52.23	60.44
SUSSX-C-WD (unsupervised)	54.54	39.71	45.96
SUSSX-CR (unsupervised)	54.30	39.53	45.75

TABLE II
OVERALL DISAMBIGUATION PERFORMANCE (OUR WSD SYSTEM IS MARKED IN BOLD)

dependency knowledge itself cannot pass the MFS baseline in any of these articles. Clearly integration of three types of knowledge significantly improves the WSD performance. We examined correctly disambiguated and mis-disambiguated words, and found that DepScore and GlossScore together are highly accurate. In our experiment, these two scores point to the same senses in 1007 out of 2269 target words. Among these 1007 cases, 896 of them are correctly disambiguated. In the rest of 1262 cases, GlossScore returns many zero values due to concise glosses and short context sentences, and DepScore also makes mistakes due to noisy dependency relations since one identical word can mean different things in different dependency relations. We also experimented with just two knowledge sources: (1) glosses and MFS information; (2) glosses and dependency knowledge; (3) dependency knowledge and MFS information. When only two knowledge sources are used, we adopted score threshold to eliminate noise in order to improve accuracy, e.g., when gloss overlapping score is too small we will select the first sense, but none of these combinations can outperform the MFS baseline.

Senseval-2 and Semeval 2007 WSD test corpora provide evaluation for both coarse-grained and fine-grained senses, and cover diverse topics and a significant portion of commonly-used English words (A college graduate knows approximately 20,000 - 25,000 English words). Evaluation with these two testing corpora clearly shows the effectiveness of our approach and its potential application in many practical NLP systems.

VI. CONCLUSION

Broad coverage and disambiguation quality are critical for WSD techniques to be adopted in real-world applications. This paper presents a fully automatic WSD method that utilizes three automatically accessible and disambiguation-enabling knowledge sources: glosses from dictionaries, the most-frequent-sense information, and normalized dependency knowledge extracted from unannotated text. Our WSD method overcomes the knowledge acquisition bottleneck faced by many current WSD systems. Our main finding is the "greater-sum" disambiguation capability of these three knowledge sources. We evaluated our approach with the SemEval-2007 and Senseval-2 corpora, and achieved similar performance as the top performing supervised WSD systems. With better-than-MFS-baseline performance and by using only widely available knowledge sources, our method may provide a viable solution to the problem of WSD and can be readily used in many real world Natural Language Processing applications.

ACKNOWLEDGMENTS

This work is partially funded by NSF grant DUE 0737408 and CNS 0851984. This paper contains proprietary information protected under a pending U.S. patent (No. 61/121,015).

REFERENCES

[1] Agirre, Eneko and Philip Edmonds, editors. 2006. Word Sense Disambiguation: Algorithms and Applications, Springer.
[2] Agirre, Eneko and A. Soroa. 2009. Personalizing pagerank for word sense disambiguation. In Proceedings of the 12th conference of the European chapter of the Association for Computational Linguistics (EACL-2009), pp. 33-41.
[3] Bergsma, Shane, Dekang Lin, and Randy Goebel. 2009. Web-Scale N-Gram Models for Lexical Disambiguation. IJCAI 2009: 1507-1512.
[4] Bickel, P. J. and K. A. Doksum. 2001. Mathematical Statistics: Basic Ideas and Selected Topics (Second Edition). Prentice-Hall Inc.
[5] Chen, Ping, Wei Ding, Chris Bowes, and David Brown. 2009. A Fully Unsupervised Word Sense Disambiguation Method and Its Evaluation on Coarse-grained All-words Task. NAANLP 2009. pp. 28-36.
[6] Deal, S. V., Robert R. Hoffman. 2010. The Practitioner's Cycles, Part 1: Actual World Problems. IEEE Intelligent Systems, pp. 4-9, March-April, 2010
[7] 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. pp. 24-26.
[8] McShane, Marjorie. 2009. Reference Resolution Challenges for Intelligent Agents: The Need for Knowledge. IEEE Intelligent Systems, pp. 47-58, July-August, 2009
[9] Lin, Dekang. 1998. Dependency-based evaluation of minipar. In *Proceedings of the LREC Workshop on the Evaluation of Parsing Systems*, pp. 234–241, Granada, Spain.
[10] Mihalcea, Rada. 2005. Unsupervised Large-Vocabulary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling, in Proceedings of the Joint Conference on Human Language Technology Empirical Methods in Natural Language Processing (HLT/EMNLP), Vancouver, October, 2005. pp. 411-418.
[11] Navigli, Roberto. 2009. Word Sense Disambiguation: a Survey, ACM Computing Surveys, 41(2), ACM Press, 2009. pp. 1-69.
[12] Navigli, Roberto, 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.
[13] Navigli, Roberto and Mirella Lapata. An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation. IEEE Trans. Pattern Anal. Mach. Intell. 32(4): 678-692 (2010)
[14] SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems, July 2001, Toulouse, France.
[15] Stevenson, Mark and Y. Wilks. The Interaction of Knowledge Sources in Word Sense Disambiguation. Computational Linguistics, 27(3):321C349, 2001.
[16] 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). pp. 189-196.
[17] Zhong, Zhi and Hwee Tou Ng. 2009. Word Sense Disambiguation for All Words without Hard Labor. In Proceeding of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09), pp. 1616-1621.