Learning Semantic Classes for Word Sense Disambiguation

Abstract

Word Sense Disambiguation suffers from a long-standing problem of knowledge acquisition bottleneck. Although state of the art supervised systems report good accuracies for selected words, they have not been shown to be promising in terms of scalability. In this paper, we present an approach for learning coarser and more general set of concepts from a sense tagged corpus in order to alleviate the knowledge acquisition bottleneck. We show that these general concepts can be transformed to fine grained word senses using simple heuristics, and applying the technique for recent SENSEVAL data sets shows that our approach can yield state of the art performance.

1 Introduction

Word Sense Disambiguation (WSD) is the task of determining the meaning of a word in a given context. This task has a long history in natural language processing, and is considered to be an intermediate task, success of which is considered to be important for other tasks such as Machine Translation, Language Understanding, and Information Retrieval.

Despite a long history of attempts to solve WSD problem by empirical means, there is not any clear consensus on what it takes to build a high performance implementation of WSD. Algorithms based on Supervised Learning, in general, show better performance compared to unsupervised systems. But they suffer from a serious drawback: the difficulty of acquiring considerable amounts of training data, also known as knowledge acquisition bottleneck. In the typical setting, supervised learning needs training data created for each and every polysemous word; Ng (1997) estimates an effort of 16 person-years for acquiring training data for 3,200 significant words in English. Mihalcea and Chklovski (2003) provide a similar estimate of an 80 person-year effort for creating manually labelled training data for about 20,000 words in a common English dictionary.

Two basic approaches have been tried as solutions to the lack of training data, namely unsupervised systems and semi-supervised bootstrapping techniques. Unsupervised systems mostly work on knowledge-based techniques, exploiting sense knowledge encoded in machine readable dictionary entries, taxonomical hierarchies such as WORDNET (Fellbaum, 1998), and so on. Most of the bootstrapping techniques start from a few ‘seed’ labelled examples, classify some unlabelled instances using this knowledge, and iteratively expand their knowledge using information available within newly labelled data. Some others employ hierarchical relatives such as hypernyms and hyponyms.

In this work, we present another practical alternative: we reduce the WSD problem to a one of finding generic semantic class of a given word instance. We show that learning such classes can help relieve the problem of knowledge acquisition bottleneck.

1.1 Learning senses as concepts

As the semantic classes we propose learning, we use top-most nodes of WORDNET hypernym trees,
also known as unique beginners. By learning these generic senses, we show that we can reuse training data, without having to rely on specific training data for each word. This can be done because the semantic classes are common to words; for learning the properties of a given class, we can use the data from various words. For instance, the noun crane falls into two semantic classes ANIMAL and ARTEFACT. We can expect the words such as pelican and eagle (in the bird sense) to have similar usage patterns to those of ANIMAL sense of crane, and to provide common training examples for that particular class.

For learning these classes, we can make use of any training example labelled with WordNet senses for supervised WSD, as we describe in section 3.1. In this paper, we limit our training data to a portion of SEMCOR corpus (Fellbaum, 1998).

Once the classification is done for an instance, the resulting semantic classes can be transformed into finer grained senses using some heuristical mapping, as we show in the next sub section. This would not guarantee a perfect conversion because such a mapping can miss some finer senses, but as we show in what follows, this problem in itself does not prevent us from attaining good performance in a practical WSD setting.

1.2 Information loss in coarse grained senses

As an empirical verification of the hypothesis that we can still build effective fine-grained sense disambiguators despite the loss of information, we analyzed the performance of a hypothetical coarse grained classifier that can perform at 100% accuracy. As the general set of classes, we used WordNet unique beginners, of which there are 25 for nouns, and 15 for verbs.

To simulate this classifier on Senseval English all-words tasks’ data (Edmonds and Cotton, 2001; Snyder and Palmer, 2004), we mapped the fine-grained senses from official answer keys to their respective beginners. There is an information loss in this mapping, because each unique beginner can typically include more than one sense. To see how this ‘classifier’ fares in a fine-grained task, we can map the ‘answers’ back to WordNet fine-grained senses, by picking up the sense with the lowest sense number that falls within each unique beginner. In principal, this is the most likely sense within the class, because WordNet senses are said to be in descending order of frequency. Since this sense is not necessarily the same as original sense of the instance, the accuracy of the fine-grained answers will be below 100%.

Figure 1 shows the performance of this transformed fine-grained classifier (CG) for nouns and verbs with Senseval-2 and 3 English all words task data (marked as S2 and S3 respectively), along with the baseline WordNet first sense (BL), and the best-performer classifiers (CL), SMUaw (Mihalcea, 2002) and GAMBL-AW (Decadt et al., 2004) respectively.

There is a considerable difference in the improvement over baseline between the state-of-the-art systems and the hypothetical optimal coarse-grained system. This shows us that there is an improvement in performance that we can attain over the state-of-the-art, if we can create a classifier for even a very coarse level of senses, with sufficiently high accuracy. We believe that the chances for such a high accuracy in a coarse-grained sense classifier is better, for several reasons:

- previously reported good performance for coarse grained systems (Yarowsky, 1992)
- better availability of data, due to the possibility of reusing data created for different words. For instance, labelled data for the noun ‘crane’ is not found in SEMCOR corpus at all, but there are more than 1000 sample instances for the
1.3 Overall approach

Basically, we assume that we can learn the ‘concepts’, in terms of WORDNET unique beginners, using a set of data labelled for these concepts, regardless of the actual word that is labelled. Hence, we can use a generic data set that is large enough, where various words provide training examples for these concepts, instead of relying upon data from the examples of same word as the word being classified.

Unfortunately, simply labelling each instance with its semantic class and then using standard supervised learning algorithms did not work well. This is probably because the effectiveness of the feature patterns often depend on the actual word being disambiguated and not just its semantic class. For example, the phrase ‘run the newspaper’ effectively indicates that ‘newspaper’ belongs to the semantic class GROUP. But ‘run the tape’ indicates that ‘tape’ belongs to the semantic class ARTEFACT. The collocation ‘run the’ is effective for indicating the GROUP sense only for ‘newspaper’ and closely related words such as ‘department’ or ‘school’.

In this paper, we use the k-nearest neighbor classifier. In order to allow training examples of different words from the same semantic class to effectively provide information for each other, we modify the distance between instances in a way that makes the distance between instances of similar words smaller. This is described in Section 3.

The rest of the paper is organized as follows: In section 2, we discuss several related work. We proceed on to a detailed description of our system in section 3, and discuss the empirical results in section 4, showing that our representation can yield state of the art performance.

2 Related Work

Although there is not much related work done in recent WSD research on systems specifically designed for learning semantic classes, some of the work in last decade show theoretical similarity to our approach.

Using generic classes as word senses has been done several times in WSD, in various contexts. Yarowsky (1992) used Roget’s thesaurus categories as classes for word senses. He used samples from Grolier’s Encyclopedia to learn representative contexts for these classes, and assumed that different senses of a word occur in recognizably different contexts. By using this technique, he reported accuracy figures above 90% for 12 polysemous words.

Resnik (1997) used a similar set of conceptual classes for word senses. As the means of sense disambiguation, he employed selectional preferences, based on the idea that certain linguistic predicates constrain the semantic interpretation of underlying words. He proposed a method for automatically acquiring these constraints from a raw corpus. Accuracy levels he reported are in the range of 35-45% for fine-grained senses.

WSD System presented by Crestan et al. (2001) in SENSEVAL-2 classified words in to WORDNET unique beginners. However, their approach did not use the fact that the primes are common for words, and training data can hence be reused. Their method considered all semantic classes of a word regardless of its part of speech.

Using related words such as hypernyms and hyponyms in order to increase the amount of data has been reported several times. Among these systems, Mihalcea (2002) reported an iterative bootstrapping algorithm, which could perform best at SENSEVAL-2 evaluation exercise in English all-words task.

3 Basic Design of the System

The system consists of three classifiers, built using local context, part of speech and syntax-based relationships respectively, and combined with the most-frequent sense classifier by using weighted majority voting. Our experiments (section 4.3) show that building separate classifiers from different subset of features and combining them works better than building one classifier by concatenating the features together. Possibly this is because the feature subsets are mutually quite independent given the class.

For training and testing, we used publicly available data sets, namely SEMCOR corpus and SENSE-
English all-words task data. In order to evaluate the system’s performance in vivo, we mapped the outputs of our classifier to the answers given in the key. Although we face a penalty here due to the loss of granularity, this approach allows a direct comparison of actual usability of our system.

### 3.1 Data

As training corpus, we used Brown-1 and Brown-2 parts of **SEM**COR corpus; these parts have all of their open-class words tagged with corresponding **WORD**NET senses. A part of the training corpus was set aside as the development corpus. This part was selected by randomly selecting a portion of multi-class words from the prepared data set. As labels, the semantic class (lexicographic file number) was extracted from the sense key of each instance. Testing data sets from **SENSEVAL**-2 and **SENSEVAL**-3 English all-words tasks were used as testing corpora.

### 3.2 Features

The feature set we selected was fairly simple; As we understood from our initial experiments, wide-window context features and topical context were not of much use for learning semantic classes from a multi-word training data set. Instead of generalizing, wider context windows add to noise, as seen from validation experiments with held-out data.

#### 3.2.1 Local context

This is a window of \( n \) words to the left, and \( n \) words to the right, where \( n \in \{1, 2, 3\} \) is a parameter we selected via cross validation. The actual word under consideration is excluded, so the vector has \( 2n \) elements.\(^1\) Punctuation marks were removed and all words were converted in to lower case. The feature vector was calculated the same way for both nouns and verbs. The window did not exceed the boundaries of a sentence; when there were not enough words to either side of the word within the window, the value NULL was used to fill the remaining positions.

For instance, for the verb ‘companion’ in sentence (given with POS tags)

\[
\text{'Henry/NNP peered/VBD doubtfully/RB at/IN his/PRP$ drinking/NN companion/NN through/IN bleary/JJ ,/, tear-filled/JJ eyes/NNS ./'}
\]

the local context feature vector is \([\text{at, his, drinking, through, bleary, tear-filled}], \) for window size \( n = 3 \). Notice that we did not consider the hyphenated words as two words, when the data files had them annotated as a single token.

#### 3.2.2 Part of speech

This consists of parts of speech for a window of \( n \) words to both sides of word (excluding the word itself), with quotation signs and punctuation marks ignored. For **SEM**COR, existing parts of speech were used; for **SENSEVAL** data files, parts of speech from the accompanying Penn-Treebank parsed data files were aligned with the XML data files. The value vector is calculated the same way as the local context, with the same constraint on sentence boundaries, replacing vacancies with NULL.

As an example, for the sentence we used in the previous example, the part-of-speech vector with context size \( n = 3 \) for the verb **peered** is \([\text{NULL, NULL, NNP, RB, IN, PRP}$].

#### 3.2.3 Syntactic relations with the word

The words that hold several kinds of syntactic relations with the word under consideration were selected. We used Link Grammar parser due to Sleator and Temperley (1991) because of the information-rich parse results produced by it.

Both **SEM**COR corpus files and the **SENSEVAL** files were parsed with Link parser, and words were aligned with links. A given instance of a word can

| Feature                      | Example                      | Value   |
|------------------------------|------------------------------|---------|
| Subject - verb               | He [sells] his art           | be      |
| Verb - object                | He [sells] his art           | art     |
| Adjectival modifier          | He will [sell] his art       | he      |
| Prepositional connectors     | He can [paint] well          | well    |
| Post-nominal modifiers       | to boldly [go]               |         |
| Adverbial modifier           | He can [paint] well          | well    |
| Words in split infinitives   | to boldly [go]               |         |

Table 1: Syntactic relations used as features. The target word is shown inside [brackets]
have more than one syntactic features present. Each of these features was considered as a binary feature, and a vector of binary values was constructed, of which each element denoted a unique feature found in the test set of the word.

Each syntactic pattern feature falls in to either of two types collocation or relation:

**Collocation features** Collocation features are such features that connect the word under consideration to another word, with a preposition or infinitive in between — for instance, the phrase ‘art of change-ringing’ for the word art. For these features, the feature value consists of two words, which are connected to the given word either from left or from right, in a given order. When encoding these features, we specified the connected phrase, and whether it connects to the word from left or right. For the above example, the feature value is [∼.of.change-ringing], where ∼ denotes the placeholder for word under consideration.

**Relational features** Relational features are the features that connect a word to the word under consideration with simpler and more direct relationships, such as subject-verb or noun-adjective. When encoding the feature value, we specified the relation type, and the value of the feature in the given instance. For instance, in the phrase ‘Henry peered doubtfully’, the adverbial modifier feature for the verb ‘peered’ is encoded as [adverb-mod doubtfully].

A description of the relations for each part of speech are given in the table 1.

### 3.3 Classifier and instance weighting

The classifier we used was TiMBL, a memory based learner due to Daelemans et al. (2003). One reason for this choice was that memory based learning, (also known as example-based, similarity-based, case-based, analogical, or instance-based learning) has shown to perform well in previous word sense disambiguation tasks, including some best performers in SENSEVAL, such as (Hoste et al., 2001; Decadt et al., 2004; Mihalcea and Faruque, 2004). Another reason is that it supported exemplar weights, a necessary feature in our system as we describe in the next section.

One of the salient feature of our system is that it does not take all the examples as similarly important. Due to the fact that training instances from different instances can provide confusing examples as shown in section 1.3, such an approach cannot be trusted to give good performance; we verified this by our own findings through empirical evaluations as shown in section 4.2.

#### 3.3.1 Weighting instances with similarity

We use a similarity based measure to assign weights to training examples. In the method we use, these weights are used to adjust the distances between the test instance and the example instances. The distances are adjusted according to the formula

\[ \Delta^E(X, Y) = \frac{\Delta(X, Y)}{ew_X + \epsilon}, \]

where \( \Delta^E(X, Y) \) is the adjusted distance between instance \( Y \) and example \( X \), \( \Delta(X, Y) \) is the original distance, \( ew_X \) is the exemplar weight of instance \( X \). The small constant \( \epsilon \) is added to avoid division by zero.

There are various schemes used to measure intersense similarity. Our experiments showed that the measure defined by Jiang and Conrath (1997) (JCn) is the best-performing. Results for various weighting schemes are discussed in section 4.2.

#### 3.3.2 Instance weighting

The exemplar weights were derived from the following method:

1. pick a labelled example \( e \), and extract it’s sense \( s_e \) and semantic class \( c_e \).

2. if the class \( c_e \) is a candidate for the current test word \( w \), i.e. the candidate word has any senses that fall in to \( c_e \), find out the most frequent sense of \( w \), \( s^w_{c_e} \), within \( c_e \). We define the most frequent sense within a class as the sense that has the lowest WORDNET sense number within that class. If none of \( w \)’s senses fall in to \( c_e \), we ignore that example.

3. calculate the relatedness measure between \( s_e \) and \( s^w_{c_e} \), using whatever the similarity metric being considered. This is the exemplar weight for example \( e \).
An Example  Consider the word ‘dog’: this word falls in to three semantic classes, namely ANIMAL, PERSON, and ARTEFACT. Suppose in the training corpus, we have an instance ‘I have a pet cat’, labelled with sense cat/1: this particular sense of cat falls in to the class ANIMAL. Since this is a candidate prime class for dog, we query for the smallest sense number of dog which falls in to class ANIMAL, which happens to be sense 1. Then a straightforward lookup for JCn similarity between senses cat/1 and dog/1 gives us a similarity value of 0.546. This is the weight of the example in the training set for word dog.  

In the implementation, we used freely available WordNet::Similarity package, implemented by Ted Pedersen and colleagues, to calculate sense similarities.  

3.4 Classifier optimization  
A part of SEMCOR corpus (600 instances for each part of speech) was used as a validation set; this part was selected by random sampling of instances of words that fall into more than one semantic class. The rest of the multiple class words and all single class word instances were used as training data in validation phase. In the preliminary experiments, it was seen that the generally recommended classifier options yield good enough performance, although variations of switches could improve performance slightly in certain cases. Classifier options were selected by a search over the available option space for only three basic classifier parameters, namely, number of nearest neighbors, distance metric and feature weighting scheme.  

4 Results  
In what follows, we present the results of our experiments in various test cases. We combined the three classifiers and the WORDNET first-sense classifier through simple majority voting. For evaluating the systems with SENSEVAL data sets, we mapped the outputs of our classifiers to WORDNET senses by picking the most-frequent sense (the one with the lowest sense number) within each of the class. 

| Classifier     | Senseval-2 | Senseval-3 |
|----------------|------------|------------|
| Baseline       | 0.617      | 0.627      |
| POS            | 0.616      | 0.614      |
| Local context  | 0.627      | 0.633      |
| Synt. Pat      | 0.620      | 0.612      |
| Concatenated   | 0.609      | 0.611      |
| **Combined**   | **0.631**  | **0.643**  |

Table 2: Results of baseline, individual, and combined classifiers: recall measures for nouns and verbs combined.

mapping was used in all tests. For all evaluations, we used SENSEVAL official scorer. We could use the setting only for nouns and verbs, because the similarity measures we used were not defined for adjectives or adverbs, due to the fact that hypernyms are not defined for these two parts of speech. So we list the initial results for only nouns and verbs. 

4.1 Individual classifiers vs. combination  
We evaluated the results of the individual classifiers before combination. Only local context classifier could outperform the baseline in general, although there’s a slight improvement with the syntactic pattern classifier on SENSEVAL-2 data. The results are given in the table 2, together with the results of voted combination, and baseline WORDNET first sense. Classifier shown as ‘concatenated’ is a single classifier trained from all of these feature vectors concatenated to make a single vector. Concatenating features this way does not seem to improve the performance. 

4.2 Effect of similarity measure  
Table 3 shows the effect of JCn and Resnik similarity measures, along with no similarity weighting, for the combined classifier. It’s clear that proper similarity measure has a major impact on the performance, with Resnik measure performing worse than the baseline. 

4.3 Optimizing the voting process  
Several voting schemes were tried for combining classifiers. Simple majority voting improves performance over baseline. However, previously reported results such as (Hoste et al., 2001) and (Decadt et al.,

\footnote{WordNet::Similarity is a perl package available freely under GNU General Public Licence. http://wn-similarity.sourceforge.net.}
Table 3: Effect of different similarity schemes on recall, combined results for nouns and verbs

| Similarity Scheme | Senseval-2 | Senseval-3 |
|------------------|------------|------------|
| No similarity used | 0.608 | 0.599 |
| Resnik | 0.540 | 0.522 |
| JCn | 0.631 | 0.643 |

Table 4: Improvement of performance with classifier weighting. Combined results for nouns and verbs with voting schemes Simple Majority (SM), Global classifier weights (GW) and local weights (LW).

| Voting Scheme | Senseval-2 | Senseval-3 |
|---------------|------------|------------|
| SM | 0.631 | 0.643 |
| GW | 0.634 | 0.649 |
| LW | 0.641 | 0.650 |

2004) have shown that optimizing the voting process helps improve the results. We used a variation of Weighted Majority Algorithm (Littlestone and Warmuth, 1994). The original algorithm was formulated for binary classification tasks; however, our use of it for multi-class case proved to be successful.

We used the held-out development data set for adjusting classifier weights. Originally, all classifiers have the same weight of 1. With each test instance, the classifier builds the final output considering the weights. If this output turns out to be wrong, the classifiers that contributed to the wrong answer get their weights reduced by some factor. We could adjust the weights locally or globally; in global setting, the weights were adjusted using a random sample of held-out data, which contained different words. These weights were used for classifying all words in the actual test set. In local setting, each classifier weight setting was optimized for individual words that were present in test sets, by picking up random samples of the same word from SEMCOR.\(^3\) Table 4 shows the improvements with each setting.

4.4 Final results in Senseval

Here, we list the performance of the system with adjectives and adverbs added for ease of comparison. Due to the facts mentioned at the beginning of this section, our system was not applicable for these parts of speech, and we classified all instances of these two POS types with their most frequent sense. We also identified the multi-word phrases from the test documents. These phrases generally have a unique sense in WORDNET; we marked all of them with their first sense without classifying them. All the multiple-sense instances of nouns and verbs were classified and converted to WORDNET senses by the method described above, with locally optimized classifier voting.

The results of the systems are shown in tables 6 and 7. Our system’s results in both cases are listed as SimilPrime, along with the baseline WORDNET first sense (including multi-word phrases and ‘U’ answers), and the two best performers’ results reported.\(^4\) These results compare favorably with the official results reported in both tasks.

Significance of results To verify the significance of these results, we used one-tailed paired t-test, using results of baseline WORDNET first sense and our system as pairs. Tests were done both at micro-average level and macro-average level, (considering test data set as a whole and considering per-word average). Null hypothesis was that there is no significant improvement over the baseline. Both settings yield good significance levels, as shown in table 5.

\(^3\)Words for which there were no samples in SEMCOR were classified using a weight of 1 for all classifiers.

\(^4\)The differences of the baseline figures from the previously reported figures are clearly due to different handling of multi-word phrases, hyphenated words, and unknown words in each system. We observed by analyzing the answer keys that even better baseline figures are technically possible, with better techniques to identify these special cases.

Table 5: One tailed paired t-test significance levels of results: \(P(T \leq t)\)

| System | Recall |
|--------|--------|
| SMUaw (Mihalcea, 2002) | 0.690 |
| Simil-Prime | 0.664 |
| Baseline (WORDNET first sense) | 0.648 |
| CNTS-Antwerp (Hoste et al., 2001) | 0.636 |

Table 6: Results for Senseval-2 English all words data for all parts of speech and fine grained scoring.

| System | Recall |
|--------|--------|
| Micro Average | < 0.0001 |
| Macro Average | 0.0073 | 0.0252 |
| System                        | Recall |
|-----------------------------|--------|
| Simil-Prime                 | 0.661  |
| GAMBL-AW-S (Decadt et al., 2004) | 0.652  |
| SenseLearner (Mihalcea and Faruque, 2004) | 0.646  |
| Baseline (WORDNET first sense) | 0.642  |

Table 7: Results for SENSEVAL-3 English all words data for all parts of speech and fine grained scoring.

5 Conclusion and Future Work

We analyzed the problem of Knowledge Acquisition Bottleneck in WSD, and proposed using a general set of semantic classes as a trade-off, and discussed why such a system is promising. Our formulation allowed us to use training examples from words different from the actual word being classified. This makes the available labelled data reusable for different words, relieving the above problem. In order to facilitate learning, we introduced a technique based on word sense similarity.

The generic classes we learned can be mapped to finer grained senses with simple heuristics. Through empirical findings, we showed that our system can attain state of the art performance, when applied to standard fine-grained WSD evaluation tasks.

In the future, we hope to improve on these results: Instead of using WORDNET unique beginners, using more natural semantic classes based on word usage would possibly improve the accuracy, and finding such classes would be a worthwhile area of research. As seen from our results, selecting correct similarity measure has an impact on the final outcome. We hope to work on similarity measures that are more applicable in our task.

References

E. Crestan, M. El-Bèze, and C. De Loupy. 2001. Improving wsd with multi-level view of context monitored by similarity measure. In Proceeding of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems, Toulouse, France.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2003. TiMBL: Tilburg memory based learner, version 5.0, reference guide. Technical report, ILK 03-10.

Bart Decadt, Véronique Hoste, Walter Daelemans, and Antal Van den Bosch. 2004. GAMBL, genetic algorithm optimization of memory-based wsd. In Senseval-3: Third Intl. Workshop on the Evaluation of Systems for the Semantic Analysis of Text.

P. Edmonds and S. Cotton. 2001. Senseval-2: Overview. In Proc. of the Second Intl. Workshop on Evaluating Word Sense Disambiguation Systems (Senseval-2).

C. Fellbaum. 1998. WordNet: An Electronic Lexical Database. The MIT Press, Cambridge, MA.

Véronique Hoste, Anne Kool, and Walter Daelmans. 2001. Classifier optimization and combination in english all words task. In Proceeding of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems.

J. Jiang and D. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings of International Conference on Research in Computational Linguistics.

N Littlestone and M.K. Warmuth. 1994. The weighted majority algorithm. Information and Computation, 108(2):212–261.

Rada Mihalcea and Tim Chklovski. 2003. Open mind word expert: Creating large annotated data collections with web users’ help. In Proceedings of the EACL 2003 Workshop on Linguistically Annotated Corpora.

Rada Mihalcea and Ehsanul Faruque. 2004. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In Senseval-3: Third Intl. Workshop on the Evaluation of Systems for the Semantic Analysis of Text.

Rada Mihalcea. 2002. Bootstrapping large sense tagged corpora. In Proc. of the 3rd Intl. Conference on Languages Resources and Evaluations.

Hwee Tou Ng. 1997. Getting serious about word sense disambiguation. In Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?, pages 1–7.

P. Resnik. 1997. Selectional preference and sense disambiguation. In Proc. of ACL Siglex Workshop on Tagging Text with Lexical Semantics, Why, What and How?

D. Sleator and D. Temperley. 1991. Parsing english with a link grammar. Technical report, Carnegie Mellon University Computer Science CMU-CS-91-196.

B. Snyder and M. Palmer. 2004. The english all-words task. In Senseval-3: Third Intl. Workshop on the Evaluation of Systems for the Semantic Analysis of Text.

David Yarowsky. 1992. Word-sense disambiguation using statistical models of Roget’s categories trained on large corpora. In Proceedings of COLING-92, pages 454–460.