AN UNSUPERVISED SEMANTIC SIMILARITY BASED METHOD FOR WORD SENSE DISAMBIGUATION

SEDAT ÇANKAYA

BOĞAZİÇİ UNIVERSITY

2010

AN UNSUPERVISED SEMANTIC SIMILARITY BASED METHOD FOR WORD SENSE DISAMBIGUATION

Thesis submitted to the

Institute for Graduate Studies in the Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Management Information Systems

by

Sedat Çankaya

Boğaziçi University

2010

Thesis Abstract

Sedat Çankaya, "An Unsupervised Semantic Similarity Based Method for Word Sense Disambiguation"

In this thesis, a semantic similarity based unsupervised method for word sense disambiguation is presented. The method tries to disambiguate a target word by calculating a similarity score between the words surrounding the target word and the words existing in the sense definition of the target word. The built-in semantic hierarchy and synset relations of WordNet, a machine readable thesauri, are used in similarity score calculations. The method is evaluated using SemCor data and the results are compared against other methods based on semantic similarity and unsupervised methods. Results show us that increasing the number of inputs by including the words in a word's sense into disambiguation process, improves precision rate of disambiguation process.

Tez Özeti

Sedat Çankaya, “Kelime Anlamı Berraklaştırma İçin Anlam Benzerliği Tabanlı Denetimsiz Bir Yöntem”

Bu tezde kelime anlamı berraklaştırma için anlam benzerliği tabanlı denetimsiz bir yöntem sunulmaktadır. Bu yöntem bir kelimenin doğru anlamını tespit edebilmek için, metin içerisinde o kelimenin yakınında bulunan diğer kelimeler ile hedef kelimenin sözlük anlamında geçen kelimeler arasındaki anlam benzerliği derecesini hesaplamaya çalışır. Bu hesaplama, makine tarafından okunabilir bir sözlük olan WordNet in içerisinde halihazırda mevcut bulunan anlam sıradüzeni ile eşanlam kümeleri arasındaki ilişkiler kullanılarak yapılmaktadır. Burada sunulan yöntem SemCor dan alınan girdiler ile denendikten sonra, elde edilen sonuçlar diğer anlam tabanlı ve denetimsiz yöntemlere ait sonuçlarla karşılaştırılmıştır. Deneyler sonucunda elde edilen sonuçlar, bir kelimenin sözlük anlamında geçen kelimeleri kelime anlamı berraklaştırma sürecine dahil ederek mevcut girdi sayısını arttırmanın sürecin başarı yüzdesini arttırdığını göstermiştir.

CONTENTS

CHAPTER I

INTRODUCTION

Problem Definition and the Motivation Behind

Words can have different senses although they are pronounced and written as same, and those senses may be similar or completely irrelevant which leads severe misunderstandings. More than 73% of the words in English are polysemous, in other words 73% of the words in English have at least 2 senses. Indeed the average number of different senses of a word is much more than 2, it is 6.55 (Mihalcea et al. 2001). Following example examines the word “crane” which is having two completely different senses:

- Sense 1: a lifting machine that can be used both to lift and lower materials and to move them horizontally.
- Sense 2: large long-necked wading bird of marshes and plains in many parts of the world.

in the two sentences below:

- Sentence 1: It's a large crane barge that was once used to offload ships in Aqaba.
- Sentence 2: This illustrated volume is devoted to the Japanese crane, one of the world's rarest and most strikingly beautiful birds.

We as human beings, can easily assign first sense to the first sentence and the second sense to the second sentence by looking at the context of the sentences. But it is not the case for computers, a computational model is needed for a computer to make a reasonable choice between those senses. Word Sense Disambiguation (WSD) is the ability to identify the correct meaning of words in context in a computational manner.

According to the fact that a word has an average of 6.55 senses, a random sense assignment to a word will give a success rate of 15%. This low value is the motivation behind the development of WSD methods.

Application Areas of WSD

WSD is not a main task itself, it is an intermediary task that can be used to improve the performance and reliability of a language processing application. The areas that WSD can be used are:

Machine Translation

Sense disambiguation can be used for proper machine translation of a text, especially for the words having completely irrelevant senses. The sentence below can be taken as an example:

"Our village lies on the southern bank of the Patna river."

In this sentence, the two different senses of the word “bank” and their corresponding Turkish equivalents can be considered as:

- Sense 1: a financial institution that accepts deposits and channels the money into lending activities: *banka*
- Sense 2: sloping land (especially the slope beside a body of water): *kıyı, yaka*

Realizing the word “river” in the sentence, a human being can easily grab the context and decide that the second sense of the word “bank” is appropriate in this sentence. And translates the sentence as:

Köyümüz Patna nehrinin güney kıyısı boyunca uzanır.

However, a machine translator without having a WSD procedure translates the sentence using the most common usage of the word “bank”; *banka* and breaks the meaining unity of the sentence:

Köyümüz Patna nehrinin güney bankası boyunca uzanır.

Information Retrieval

When searching for specific keywords, it is desirable to eliminate occurrences in documents where the word or words are used in an inappropriate sense (Ide et al.,

1998). For example when searching for the word “bass” for a musical research, the search results including the other meaning of bass (a kind of fish) should be eliminated. Hence documents should not be ranked based on words alone. To get relevant results while doing a search on web, documents should be ranked based on word senses, or based on a combination of word senses and words.

Speech Processing

WSD can be used in speech recognition and speech to text applications, while processing homophone words (i.e pronounced same but spelled different). For example, cell-sell, bear-bare, fair-fare, dear-deer etc. It is difficult for a speech to text application to differentiate those homophone words, as they are pronounced likely. WSD methods can help those speech to text applications, by selecting the correct word according to the context.

Text Processing

In text to speech applications, to differentiate the words that are spelled same but pronounced different WSD methods can be applied. For example, for two different senses of the word “lead”, the word is pronounced differently:

- Sense 1: a position of leadership
- Sense 2: a soft heavy toxic malleable metallic element

WSD can be added to the text processing applications as a submodule to identify the correct sense and thus correct spelling of those type of words.

CHAPTER II

WSD METHODS

Basic Framework

As WSD is the task of matching a word with its correct sense, there exists two main tasks in disambiguation process. The first task is gathering all different senses of a word and the second task is deciding the correct sense among them and associating that sense with the word (Ide et al., 1998).

Determination of Senses

As it is stated above, first step of WSD is determination of all the different senses for each word relevant to the text or discourse under consideration. Much recent work on WSD relies on pre-defined senses for this step. The sources used are:

- a list of senses such as those found in everyday dictionaries.
- a group of features, categories, or associated words (e.g., synonyms, as in a thesaurus).
- an entry in a transfer dictionary which includes translations in another language.

WordNet, which is a useful tool for computational linguistic and natural language processing, is a lexicon database that is used widely by WSD methods as a word sense

source. WordNet is a combination of a thesaurus and a dictionary which provides different sense definitions of a word, groups words into sets of synonyms called synsets and records various semantic relations between those synsets (Fellbaum, 1998). Those relations can be listed as:

- synonymy: two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value. X is synonym of Y, if X has an equivalent meaning with Y.
- antonymy: X is an antonym of Y, if X expresses an oppposite meaning of Y.
- hypernymy: Y is a hypernym of X if every X is a (kind of) Y (canine is a hypernym of dog)
- hyponymy: Y is a hyponym of X if every Y is a (kind of) X (dog is a hyponym of canine)
- coordinate terms: Y is a coordinate term of X if X and Y share a hypernym (wolf is a coordinate term of dog, and dog is a coordinate term of wolf)
- holonymy: Y is a holonym of X if X is a part of Y (building is a holonym of window)
- meronymy: Y is a meronym of X if Y is a part of X (window is a meronym of building)

WordNet presently contains about 155,000 different words organized into 117,000 synsets. The detailed statistics about WordNet can be found in Table 1.

Table 1. Statistical Data of WordNet

POS	Unique Strings	Synsets	Total Word-Sense Pairs
Noun	117,798	82,115	146,312
Verb	11,529	13,767	25,047
Adjective	21,479	18,156	30,002
Adverb	4,481	3,621	5,580
Totals	155,287	117,659	206,941

Assigning Senses to Words

The second step of WSD involves means to assign each occurrence of a word to an appropriate sense. The assignment of words to senses is accomplished by reliance on two major sources of information:

- The context of the word to be disambiguated. In the broad sense, this includes information contained within the text or discourse in which the word appears, together with extra-linguistic information about the text such as situation, etc.
- External knowledge sources, including lexical, encyclopedic, etc. resources, as well as hand-devised knowledge sources, which provide data useful to associate words with senses.

Basically all WSD methods' aim is to find the best match between the context in which the word is used and the information sources that are listed above. In the following sections those methods are presented in a categorized manner according to the approaches they are based on.

Knowledge Based Methods

Knowledge Based Methods exploit the information in a lexical knowledge base (LKB) such as a dictionary and thesauri, without using any corpus evidence. These methods are based on the hypothesis that context knowledge can be extracted from definitions of words. The performance of knowledge based methods are usually lower than corpus based methods, but they have the advantage of a wider coverage. As opposed to corpus based methods, knowledge based methods do not need an annotated text and are applicable to any text.

Lesk Algorithm

Lesk (1986) hypothesize that the words in a dictionary definition of a target word are related with that word and the words surrounding the target are relevant to those words in the definition. The algorithm tries to disambiguate a word by counting overlaps between the surrounding words and the words in the dictionary definitions of the various senses of the target word. The sense having the highest number of word overlaps is selected as the correct sense. Although Lesk algorithm is very simple in theory and easy to implement, it forms a sound basis for upcoming Knowledge Based Methods.

As an example, while performing disambiguation for the "pine cone" phrasal, according to the Oxford Advanced Learner’s Dictionary, the word "pine" has two senses:

- Sense 1: kind of evergreen tree with needle–shaped leaves.
- Sense 2: waste away through sorrow or illness.

The word "cone" has three senses:

- Sense 1: solid body which narrows to a point.
- Sense 2: something of this shape whether solid or hollow.
- Sense 3: fruit of a certain evergreen tree.

By comparing each of the two gloss senses of the word "pine" with each of the three senses of the word "cone", it is found that the words "evergreen tree" occurs in one sense in each of the two words. So, these two senses are then declared to be the most appropriate senses when the words "pine" and "cone" are used together. A general graphical representation of Lesk algorithm can be seen in Fig. 1 (Torres et al., 2009).

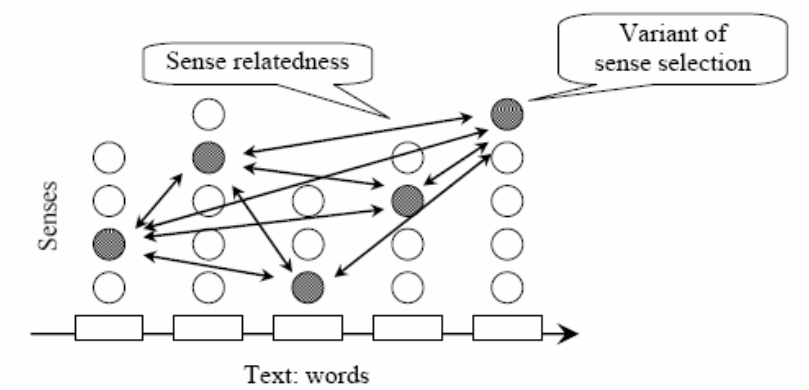


Fig. 1 Graphical representation of the Lesk algorithm

Semantic Similarity

Semantic similarity is based on the assumption that for a discourse to be coherent, the words in that discource must be related in meaning. Words in the same context are closely related in meaning, and the correct sense is the one having the minimum semantic distance or in other words the one having the maximum semantic relatedness.

There are several methods devised to quantify the semantic relatedness of two words. Several of these methods that work well with WordNet relations are described below. All of these methods accept two concepts as input and return a value indicating the semantic relatedness of these two concepts.

Leacock–Chodorow

Calculates the relatedness score according to the length of the shortest path between two synsets (Leacock et al., 1998). The resulting value is normalized by the depth of the taxonomy.

$$Similarity(C_1,C_2) = -log(Path(C_1,C_2) / 2D)$$

where Path(C_1,C_2) represents the length of the path, i.e. the number of arcs in the semantic network, between the concepts C_1 and C_2 and D is the overall depth of the taxonomy.

Hirst and Onge

In addition to the length of the path between two concepts, this method introduces the direction of the links connecting the path between those two concepts (Hirst et al., 1998). An upward direction link corresponds to generalization (hypernymy), a downward link indicates a specialization (hyponymy) and a horizontal link indicates a synonymy, antonymy or coordinate term. As it can be observed from the below formula, the relatedness score decreases as the direction changes:

$$Similarity(C_1,C_2) = C - (path\ length) - k*d,$$

where d is the number of changes of direction in the path, and C and k are constants.

The number of direction changes affects relatedness negatively, because change of direction in a semantic network constitute large semantic steps. Fig. 2 (Miller et al., 1993) illustrates the concept hierarchy in WordNet and directed links between those concepts indicating the relation type between two concepts.

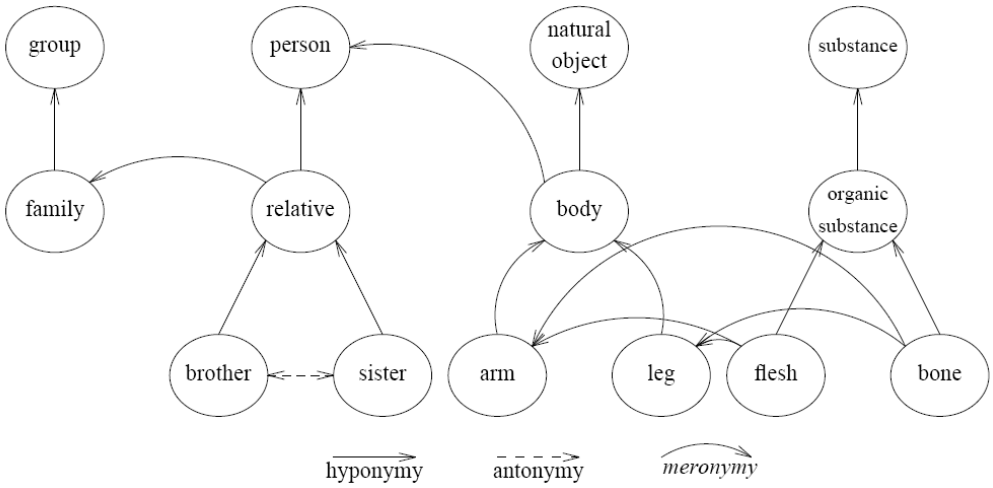


Fig. 2. Network representation of three semantic relations

Resnik

Resnik (1995) introduced a new concept, Information Concept (IC), which is a measure of the specifity of a concept and is based on its probability of occurence in a corpus. The method is based on the idea that the more specific the concept that subsumes two words is, the more semantically related are those two words. Thus the aim of this method is finding the lowest common concept of two words in the semantic hierarchy. The formulation of the method is:

$$Similarity(C_1,C_2) = IC(LCS(C_1,C_2)), IC(C) = -log(P(C))$$

In this formula, LCS is the least common subsumer of the two concepts C_1 and C_2, that is the first common node of the two concepts in the semantic hierarchy starting from the bottom of the hierarchy. And P(C) is the probability of finding the concept C in a

textual corpus. Therefore, it is obvious that P(C) value reaches its maximum as the concept is selected from the higher nodes of the semantic hierarchy.

Jiang and Conrath

As an alternative to Resnik, this method makes use of the difference in the Information Concept values of the two concepts, C_1 and C_2 (Jiang et al., 1997).

$$Similarity(C_1,C_2) = 2*IC(LCS(C_1,C_2)) - (IC(C_1) + IC(C_2))$$

Lin

Similar to Jiang and Conrath, Lin (1998) proposes another change in the original version of Resnik's formula:

$$Similarity(C_1,C_2) = (2*IC(LCS(C_1,C_2))) / (IC(C_1) + IC(C_2))$$

The semantic similarity measures presented above are tested against a Senseval-2 English lexical sample data. Window length is taken as one, i.e only one word from the left and one word from the right of the target word are included in the computations. The sense having the highest cumulative score is selected as the correct sense. And finally as a result of the experiments, it is found that among those measures Jiang and Conrath performed the best (Banerjee et al., 2003).

Selectional Preference

Selectional preferences or restrictions are constraints on the semantic type that a word sense imposes on the words with which it combines in sentences (usually through grammatical relationships) (Navigli, 2009). For example, knowing that one typically cooks food, one can disambiguate the word bass in "I am cooking bass" (i.e., it's not a musical instrument).

Although selectional preference is a simple method in theory, it is hard to put it in practice. The performance of WSD improves with the size of the selectional preferences that are defined and that is the main reason that makes it difficult to implement as it requires knowledge of the word senses involved in a candidate relation. There are methods devised to overcome this difficulty which are based on reducing the number of relations by classifying words and defining the selectional preferences among those classes instead of word-to-word relations.

The first method that Resnik (1993) suggests is defining selectional preferences between a word and a semantic class. The contribution of a semantic class is measured according to the number of concepts subsumed by that class. As the number of concepts subsumed by the semantic classes defined increases, the number of the selectional preferences to be identified decreases.

Second method, by Agirre et al. (2001), proposes a method to determine class-to-class selectional preferences which is a more generalized method than the first method suggesting word-to-class selectional preferences.

The performances of word-to-word, word-to-class and class-to-class models that are described above are evaluated by Agirre et al. (2001). According to their observations, class-to-class models performed significantly better than the other two models, word-to-word and word-to-class. The word-to-word selectional preferences give 95.9% precision and 26% coverage, word-to-class preferences decrease the precision to 66.9% and increase the coverage to 86.7%, and finally the class- to-class preferences have a precision of 66.6% and a coverage of 97.3%.

Supervised Methods

Supervised methods are based on the assumption that the context can provide enough evidence on its own to disambiguate words (Chen et al., 2009). These 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 learning techniques. In the following testing phase, a word is classified into senses. Currently supervised methods achieve the best disambiguation quality. However, these supervised methods are subject to a new knowledge acquisition bottleneck since they rely on substantial amounts of manually sense-tagged corpora for training, which are laborious and expensive to create.

Probabilistic Methods

Statistical methods usually estimate a set of probabilistic parameters that express the conditional or joint probability distributions of categories and contexts (described by features). These parameters can then be used to assign to each new example the particular category that maximizes the conditional probability of a category given the observed context features.

The Naive Bayes algorithm (Pedersen, 2000) is the simplest algorithm of this type, which uses the Bayes inversion rule and assumes the conditional independence of features given the class label. The algorithm is based on the calculation of the conditional probability of each senses of a word and the sense having the highest probability value is selected as the correct sense. An example Bayesian network can be found in Fig. 3 (Navigli, 2009).

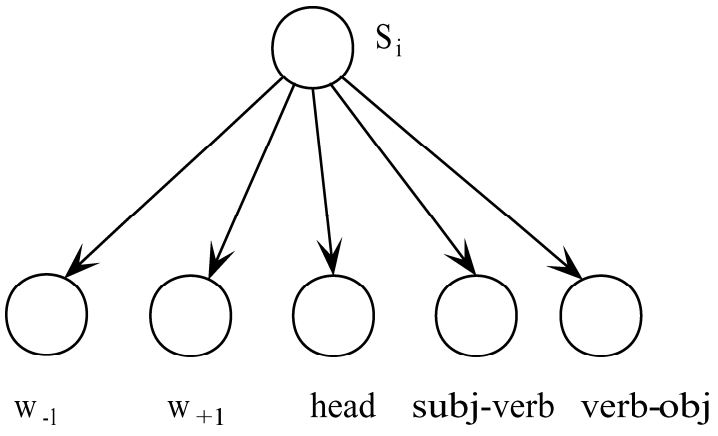


Fig. 3 An example Bayesian Network

Similarity Based Methods

The methods in this family perform disambiguation by taking into account a similarity metric. This can be done by comparing new examples to a set of learned vector prototypes (one for each word sense) and assigning the sense of the most similar prototype, or by searching in a stored base of annotated examples for the most similar examples and assigning the most frequent sense among them. The most widely used representative of this family of algorithms is the k-Nearest Neighbor (kNN) algorithm (Ng et al., 1996). In this algorithm the classification of a new example is performed by searching the set of the k most similar examples (or nearest neighbors) among a pre-stored set of labeled examples, and performing an average of their senses in order to make the prediction. In the simplest case, the training step reduces to storing all of the examples in memory and the generalization is postponed until each new example is classified. An example kNN classification is presented in Fig. 4 (Navigli, 2009).

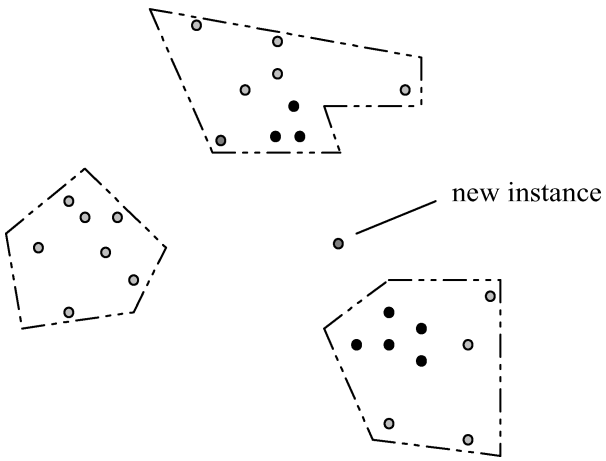


Fig. 4 An example kNN classification

Rule Based Methods

Decision lists and decision trees use selective rules associated with each word sense. The system selects one or more rules that are satisfied by the example features and assign a sense based on their predictions.

A decision list (DL), is an ordered list of rules of the form (*condition, class, weight*). An example decision list can be found in Table 2 (Navigli, 2009). According to Rivest (1987), decision lists can be considered as weighted if-then-else rules where the exceptional conditions appear at the beginning of the list (high weights), the general conditions appear at the bottom (low weights), and the last condition of the list is a "default" accepting all remaining cases. Weights are calculated with a scoring function describing the association between the condition and the particular class, and they are estimated from the training corpus. When classifying a new example, each rule in the list is tested sequentially and the class of the first rule whose condition matches the example is assigned as the result.

Table 2 An Example Decision List Table

Feature	Prediction	Score
account with bank	Bank/FINANCE	4.83
stand/V *on*/P ... bank	Bank/FINANCE	3.35
bank *of blood*	Bank/SUPPLY	2.48
work/V ... bank	Bank/FINANCE	2.33
the left/J bank	Bank/RIVER	1.12
of the bank	-	0.01

Yarowsky (1994) applied decision lists to WSD. In this work, each condition corresponds to a feature, the values are the word senses and the weights are calculated

by a log-likelihood measure indicating the plausibility of the sense given the feature value.

Combined Methods

In order to beat the weaknesses of supervised methods, a number of learning algorithms having significantly different characteristics are combined to form a single and much powerful learning algorithm.

AdaBoost (Schapire et al., 1999) is one of the most successful representatives of combined methods. AdaBoost is a general method for obtaining a highly accurate classification rule by combining many weak classifiers. A specified number of iterations are performed for each classifier. At the end of each iteration, according to the success of the classifier a weight is calculated and assigned for that classifier. For each sense of a word s_i, the weight values of each classifier whose prediction equals to s_i are summed and the s_i having the highest score is selected as the correct sense.

According to the results of the tests executed, which are listed in Table 3, AdaBoost has the best performance among the supervised methods mentioned above. On the other hand, the Naive Bayesian method performed significantly worse than the other 3 methods.

Table 3 Experiment Results of the Supervised Methods Mentioned Above

POS	Naive Bayesian (%)	kNN (%)	Decision List (%)	Ada Boost (%)
Noun	46.59	62.29	61.79	66.00
Verb	46.49	60.18	60.52	66.91
All	46.55	61.55	61.34	66.32

Semisupervised Methods

Semisupervised methods' aim is to learn sense classifiers from annotated data with minimal human supervision.

Bootstrapping

Bootstrapping methods make use of a small annotated corpus as seed data in a bootstrapping process, to overcome the knowledge acquisition bottleneck problem suffered by supervised methods. The bootstrapping approach starts from a small amount of seed data for each word: either manually-tagged training examples or a small number of decision rules. The seeds are used to train an initial classifier, using any supervised method. This classifier is then used on the untagged portion of the corpus to extract a larger training set, in which only the most confident classifications are included. The process repeats, each new classifier being trained on a successively larger training corpus, until the whole corpus is consumed, or until a given maximum number of iterations is reached. Fig. 5 (Navigli, 2009) illustrates how a small seed data evolves into a large training set.

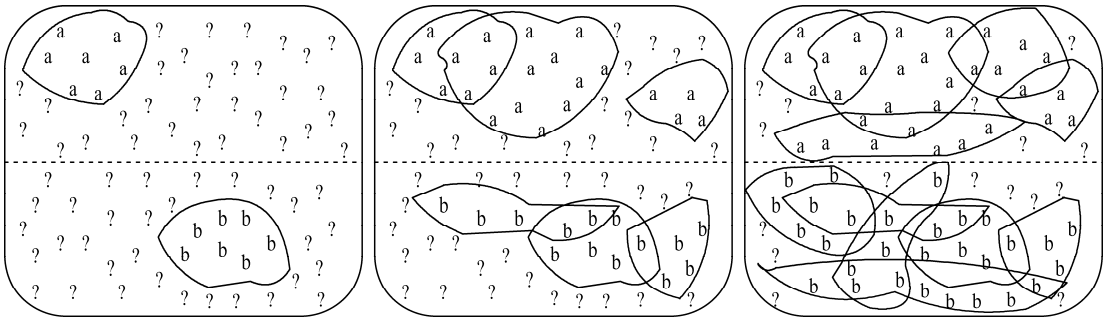


Fig. 5 Phases of Bootstrapping algorithm

The Yarowsky algorithm (Yarowsky, 1995) is one of most successful and popular applications of the bootstrapping approach. The algorithm is a simple iterative and incremental algorithm. It assumes a small seed set of labeled examples, which are representatives of each of the senses, a large set of examples to classify, and a supervised learning algorithm (decision lists). Initially, the learning algorithm is trained on the seed set and used to classify the entire set of unannotated examples. Only those examples that are classified with a confidence above a certain threshold are kept as additional training examples for the next iteration. The algorithm repeats this retraining and re-labeling procedure until convergence (i.e., when no changes are observed from the previous iteration). Yarowsky algorithm relies on two heuristics:

- one sense per collocation: nearby words strongly and consistently contribute to determine the sense of a word, based on their relative distance, order, and syntactic relationship.
- one sense per discourse: a word is consistently referred with the same sense within any given discourse or document.

The performance of Yarowsky algorithm is illustrated in Fig. 6. As it can be observed from the figure, the precision of the algorithm decreases as the recall value increases. In other words, if the number of to be disambiguated words increases, then the performance of the algorithm decreases.

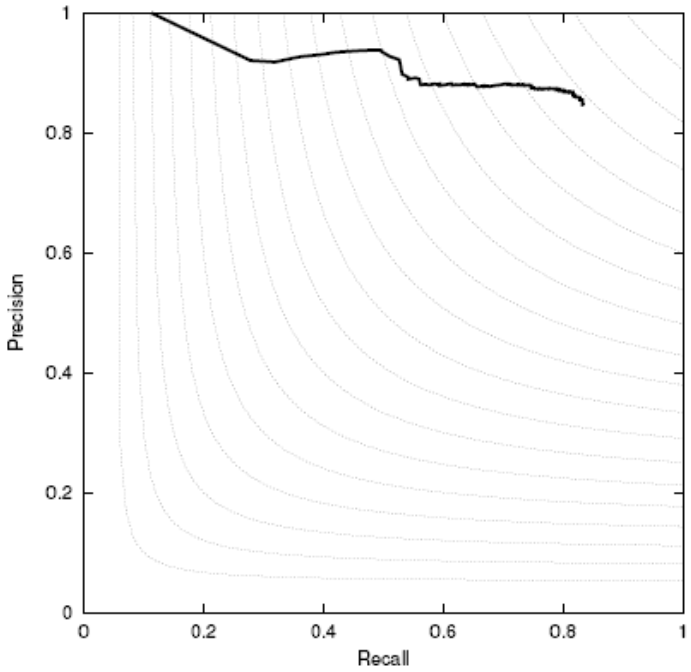


Fig. 6 Performance of the Yarowsky algorithm

Monosemous Relatives

Leacock et al. (1998) used the monosemous lexical relatives of a word sense as a key for finding training sentences in a corpus. For instance, looking for *business suit* as a monosemous hyponym of *suit* can give us training sentences for the appropriate sense of suit. Mihalcea et al. (1999) extend this idea and apply it to the Web as the target

corpus. The method works as follow: First, for a word w, all unique expressions for each sense definitions of w are determined. Then, for each expression, a search on web (through any search engine, e.g. Google) is performed and text fragments surrounding those expressions are retrieved from Web. Finally, a sense annotated corpus is created by tagging each text fragment with sense *S*.

An example regarding the three senses of church and a partial list of their corresponding monosemous relatives can be found below:

- church-1: church, Christian church, Christianity (a group of Christians; any group professing Christian doctrine or belief)
- church-2: church, church building (a place for public (especially Christian) worship)
- church-3: church service, church (a service conducted in a church)

The monosemous relatives of the corresponding senses listed above are:

- Synonyms: church building (sense 2), church service (sense 3), ...
- Direct hyponyms: Protestant Church (sense 1), Coptic Church (sense 1), ...
- Direct hypernyms: house of prayer (sense 2), religious service (sense 3), ...
- Distant hyponyms: Greek Church (sense 1), Western Church (sense 1), ...
- Siblings: Hebraism (sense 2), synagogue (sense 2), ...

Agirre et al. (2004) built a Web corpus, focusing on only the monosemous-relatives technique and applying additional filters. Monosemous relatives included in this work are synonyms, hyponyms, hypernyms, and siblings. In this work, it is shown that the monosemous relatives technique can be used to extract examples for all nouns in

WordNet. Overall, training a supervised WSD system with Web data provides better results than any unsupervised system participating in Senseval-2. Web data is shown to be very useful for WSD, but still does not match the results obtained with hand-tagged data.

Unsupervised Methods

Up to this point, knowledge based methods, supervised and semisupervised methods are reviewed. One of the main common properties among these methods is a need of manually created information source. Knowledge based methods need a machine readable dictionary, thesauri or a semantic taxonomy, while supervised and semisupervised methods employ a sense annotated text for learning phase. All these resources required to be built by human manually which is an expensive task. This situation leads to a knowledge acquisition bottleneck when dealing with a new language, new concepts/domains and large amounts of text.

Unsupervised methods have the potential to overcome this knowledge acquisition bottleneck as these methods do not rely on external knowldge sources. These methods acquire contextual information directly from unannotated raw text, and senses can be induced from text using some similarity measure. The assumption that unsupervised methods based on is that similar senses occur in similar contexts, and thus senses can be induced from text by clustering word occurrences using some measure of similarity of context. Then, new occurrences of the word can be classified into the closest induced clusters/senses. However, automatically acquired information is often

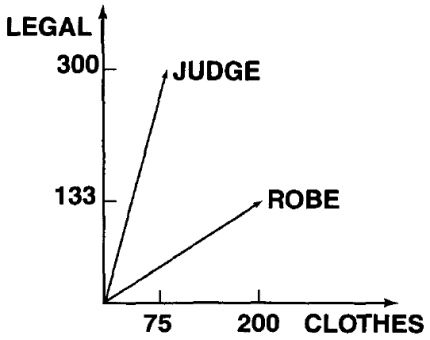
noisy or even erroneous and thus performance is lower than the other methods discussed in the above sections.

Context Clustering

Context Clustering method clusters together the contexts in which a given word occurs. The different senses of a word are represented by building a series of three vector spaces (Schutze, 1998). The first vector space is Word Space, which is a co-occurence matrix where each word is represented by a vector of co-occurence data. The words that make up the dimensions of this co-occurence matrix are determined either considering the local context that the target word occurs or the whole corpus. After the dimensions of the matrix is determined, the similarity between words could be calculated by measuring the cosine value between the word vectors. Then a context vector is created for each context that the target word occurs. The context vector is found by taking the average (centroid) of the vectors in the word space. An example word vector and a context vector is drawn in Fig. 7 (Schutze, 1998).

After all context vectors for a word are created, sense vectors are found by grouping the context vectors of a target word using a cluster algorithm (e.g. Buckshot clustering algorithm). These sense vectors represent the different senses of the target word.

Dimension	Vector *judge*	*robe*
legal	300	133
clothes	75	200



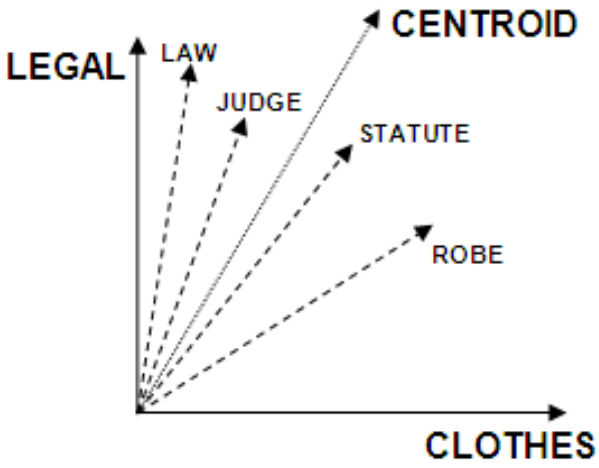


Fig. 7 The derivation of word vectors and context vectors

Word Clustering

Word Clustering methods create a representation of the different words in a corpus that attempts to capture their contextual similarity. These representations are usually based on counts of word co-occurrences or measures of association between words. Given such information about a word, it is possible to identify other words that have a similar profile and assumed to share related contexts and have similar meanings.

Lin (1998), proposed a method for the identification of words $W = (w_1, \ldots, w_k)$ similar to a target word w_0. The similarity between w_0 and w_i is determined based on the information content of their single features, given by the syntactic dependencies which occur in a corpus. The more dependencies the two words share, the higher the information content is. However, as for context vectors, the words in W will cover all senses of w_0. To discriminate between the senses, a word clustering algorithm is

applied. Let W be the list of similar words ordered by degree of similarity to w_0. A similarity tree T is initially created which consists of a single node w_0. Next, for each i $\in \{1, \ldots, k\}$, $w_i \in W$ is added as a child of w_j in the tree T such that w_j is the most similar word to w_i among $\{w_0, \ldots, w_{i-1}\}$. After a pruning step, each subtree rooted at w_0 is considered as a distinct sense of w_0.

Co-occurence Graphs

These approaches are based on the notion of a co-occurrence graph, that is, a graph G = (V, E) whose vertices V correspond to words in a text and edges E connect pairs of words which co-occur in a syntactic relation, in the same paragraph, or in a larger context.

Veronis (2004) proposed an ad hoc approach called HyperLex. First, a co-occurrence graph is built such that nodes are words occurring in the paragraphs of a text corpus in which a target word occurs, and an edge between a pair of words is added to the graph if they co-occur in the same paragraph. Each edge is assigned a weight according to the relative co-occurrence frequency of the two words connected by the edge. As a result, words with high frequency of co-occurrence are assigned a weight close to zero, whereas words which rarely occur together receive weights close to 1.

In the second step, the graph is converted into a tree through an iterative algorithm. This conversion is illustrated in Fig. 8 (Veronis, 2004). The different senses of the target word lies in the first level of the tree. Finally Minimum Spanning Tree

(MST) algorithm is used to identify the correct sense (the one having the highest weight score is selected).

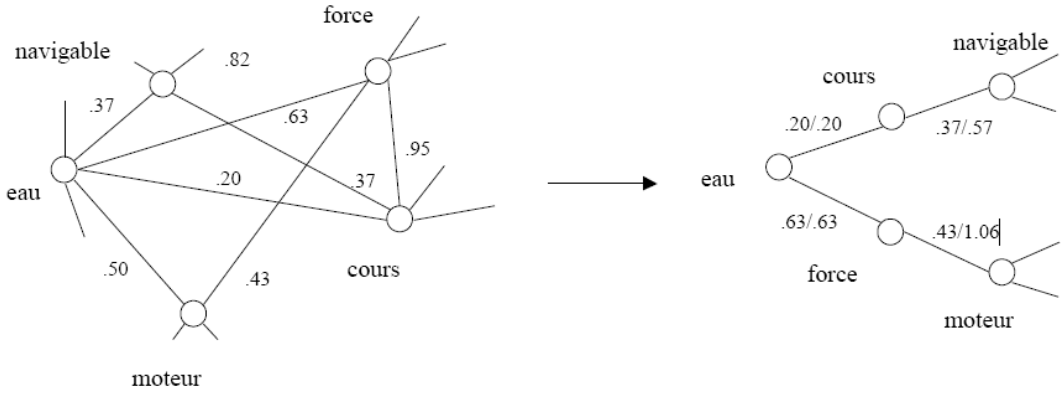


Fig. 8 Co-occurrence graph and its expansion to MST

CHAPTER III

METHODOLOGY

In this thesis, a new unsupervised word sense disambiguation method based on semantic similarity is introduced. This method is mainly based on the hypothesis that at least one synset of a target word and the synsets of the words surrounding the target word should be the same if the text is semantically coherent. A similarity score is calculated for each sense of a target word, between the words in the gloss of the target word and the words surrounding the target word. The sense having the maximum similarity score is accepted as the most relevant sense with the context and thus selected as the appropriate sense. A step by step sample execution trace of the method is presented in Appendix B.

Similarity Score Calculation

Following the hypothesis stated above, 3 measures are used to find a similarity score between two synsets: the number of semantic relation matches, the path length between the source concept and the target concept, and the distance of the surrounding word to the target word. Those 3 measures are explained in detail below.

Number of Matches

Lesk (1986) introduced the gloss overlap concept to perform word sense disambiguation. The Lesk algorithm tries to match the words in different senses of a target word with the words surrounding the target word. The sense having the maximum overlapped words is chosen as the correct sense.

Considering only the words in the sense definitions of a target word is a limitation of Lesk algorithm. Most of the sense definitions in a dictionary are short and lack of substantial amount of words required securing a reliable result. Banerjee et al. (2003) extend the classical Lesk algorithm and try to overcome the limitation by involving the concepts related with the target word and the words surrounding the target word. The concepts that can be used are the built-in semantic relations existing in WordNet which are mentioned in section 3.1. To find the appropriate sense, a score will be calculated for each sense of a target word by adding the Lesk score calculated by using sense definitions of the target word and the surrounding words to the Lesk score calculated by using the semantic relations between the target word and the surrounding words. Assuming we have two synsets A and B, the relatedness score calculation formula will be:

relatedness(A, B) = score(gloss(A), gloss(B)) + score(hypernym(A), hypernym(B))
+ score (hyponym(A), hyponym(B)) + score(gloss(A), hypernym(B))
+ score(hypernym(A), gloss(B))

Like Banerjee et al., this method extends Lesk algorithm by including the concepts of the target and surrounding words in to the process. Unlike Banarjeeet et al., only the concept itself is taken into account, not the gloss of the concept. Each match between the concepts of the surrounding words and the concepts of the words in the gloss of the target word is summed up to calculate a similarity score.

$$\begin{aligned} \text{similarity}(A, B) = {} & \text{score}(\text{hypernym}(\text{gloss}(A)), \text{hypernym}(B)) \\ & + \text{score}(\text{hyponym}(\text{gloss}(A)), \text{hyponym}(B)) \end{aligned}$$

Currently, WordNet groups nouns under 80.000 different concepts through 9 noun hierarchies and verbs under 13.500 concepts through 554 verb hierarchies. A sample concept hierarchy is illustrated in Fig. 9. This built-in hierarchical organization of concepts is useful for similarity based methods and will be used in the context of this thesis.

Path Length

As it is mentioned in Semantic Similarity section, similarity measures of Leacock et al. (1998) and Wu et al. (1994) are based on path lengths between concepts. Here what is meant by the path length is the number of concepts between the source and the target concepts. Leacock et al. find the shortest path between two concepts and scales that value by the maximum path length of those concepts. Wu et al. first find the most specific concept that both concepts share as parent concept and then finds the length of the path from this parent node to the root node.

Similar to the methods mentioned above, this method makes use of the path length between concepts to measure similarity between them. Like Wu et al., the closest parent concept that both source and target concepts share in common are found in the concept hierarchy. The level value that is going to be used in the calculation is the minimum path length from the source and the target concept to the parent concept. The relatedness between the concepts decreases as the path length increases, so the inverse of the path length is used. The formula is updated as:

similarity(A, B) =
(1/path(hypernym(gloss(A)), hypernym (B)))*score(hypernym(gloss(A)), hypernym(B))
+ (1/path(hyponym(gloss(A)), hyponym(B)))*score(hyponym(gloss(A)), hyponym(B))

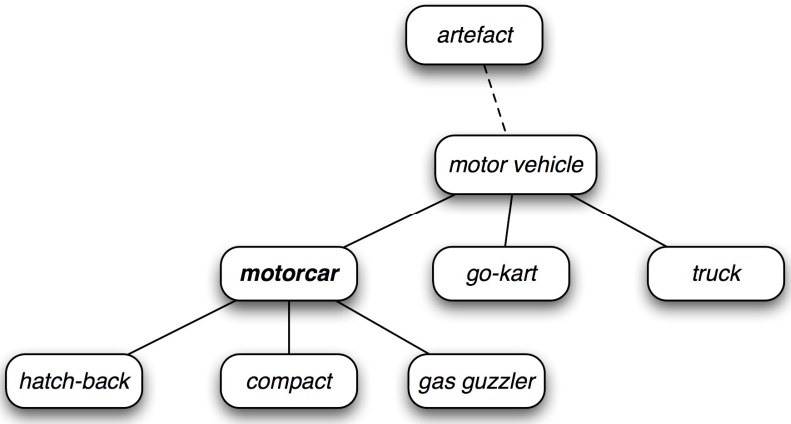


Fig. 9 Hierarchy of concepts in WordNet.

As it can be seen in Fig. 9, the length of the path between “hatch-back” and “compact” is 1 since they share the same parent concept, “motorcar”. The path length value

between “hatch-back” and “truck” is 2 as there is one sub level between “hatch-back” and “motor-vehicle”, “motorcar”.

Distance

In this method, the hypothesis which says similar words tend to occur in similar contexts is followed. The fact that, words surrounding a target word are much related with the target word than the words that are far from the target word, can be implied from this hypothesis. As a result it can be said that the distance of a word to the target word is inversely proportional to its relatedness with the target word. Regarding this, the above formula is updated as below:

```
similarity(A, B) = (1/distance(A,B)) *
      ((1/path(hypernym(gloss(A)), hypernym (B))) * score(hypernym(gloss(A)), hypernym(B))
    + (1/path(hyponym(gloss(A)), hyponym(B))) * score(hyponym(gloss(A)), hyponym(B)) )
```

As the relatedness of the words decrease with distance, the words having a distance above a certain value are assumed to be irrelevant with the target word and are not considered in the similarity score calculations. In other words, a window having a specified length is defined and only the words existing in that window are used in the calculations. The target word is always located at the center of the window as long as the target word is not at the beginning and at the end of the text. The window slides by one word to the right, as the iteration is moved to the next target word.

In the example shown in Fig. 10a, the text consists of 8 words. The target word is "administrator" and the window size is set to 4, so 2 words left to the target word and 2 words right to the target word are included in the similarity score calculation.

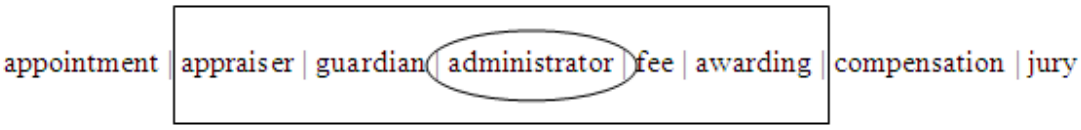


Fig. 10a Sliding window and target word

As the target word is set to the next word, "fee", the window slides by one word to the right. Now, the target word is "fee" and the words in the window are "guardian", "administrator", "awarding", "compensation". Fig. 10b illustrates this final state.

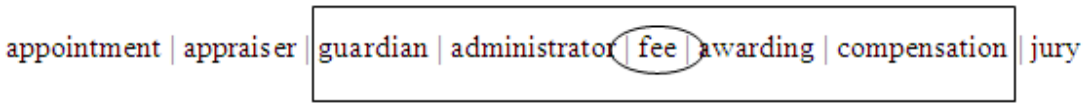


Fig. 10b Window and target word is slided one word right

Following the example shown in Fig. 10a and Fig. 10b, a distance tree can be constructed as shown in Fig. 11.

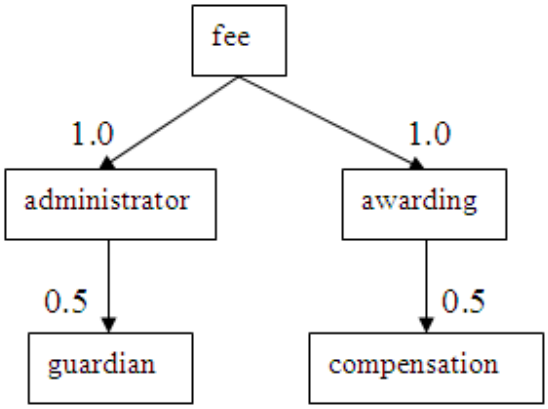


Fig. 11 Distance Tree

As all the 3 measures used in similarity score calculation are described above, a general formulation for this calculation can be presented:

Let W be our target word, t is the text and w_i denotes the other words in t, m_j is for the j^{th} sense of W in WordNet, s_{jk} stands for the k^{th} word in m_j, d_i is the unit distance of w_i to W and l denotes the synset level at which the matching is found (path length).

For each m_j, a matching score S_j is calculated as:

S_j += $(1/d_i)*(1/l_i)*(1/l_{jk})$, if a synset relation match is found between s_{jk} and w_i
S_j += 0, if there is no synset relation match between s_{jk} and w_i

After the iteration is over, the m_j having the highest score S_j is selected as the appropriate sense.

A corresponding pseudocode for the formulation above is:

```
For each sense definition do
    For each word wi in sense definition do
        For each word wj in window size do
            Calculate and sum the relation score between wi and wj
        End For
    End For
End For
```

Test Environment Settings

A test environment is set up to get the results of the method and evaluate the performance. The items required to perform the tests are: a test data to work on and a program which implements the method. As a test data, a ready to use tagged text is used. A brief explanation for this test data can be found under Semcor section. And a program is developed in Java which implements the method presented here that gets the test data as input, processes it according to the method and outputs the results for each part of speech (POS).

Test Data

For experiments, there are two kinds of test corpora that can be used as input:

- Lexical sample: the occurrences of a small sample of target words need to be disambiguated.
- All-words: all the words in a piece of running text need to be disambiguated.

All-words is a more realistic form of evaluation, but the corpus is more expensive to produce because human annotators have to read the definitions for each word in the sequence every time they need to make a tagging judgment, rather than once for a block of instances for the same target word (lexical sample). In order to define common evaluation datasets and procedures, public evaluation campaigns have been organized. Semantic Concordance (Semcor) is one of those campaigns.

Semcor

A semantic concordance is a textual corpus and a lexicon so combined that every substantive word in the text is linked to its appropriate sense in the lexicon. Thus it can be viewed either as a corpus in which words have been tagged syntactically and semantically, or as a lexicon in which example sentences can be found for many definitions. Texts that were used to create the semantic concordances were extracted from the Brown Corpus and then linked to senses in the WordNet lexicon. The semantic tagging was done by hand, using various tools to annotate the text with WordNet senses.

The "raw" data is segmented into paragraphs and sentences, and then sequentially numbered within the file. Each sentence is separated into word forms and punctuation. A semantic tag associated with a word form indicates one or more senses in the WordNet database that are appropriate for that word form in the textual context. An extract from a Semcor text file can be found in Appendix A.

Only nouns, verbs, adjectives, and adverbs (open class words) can be semantically tagged, as these are the only classes of words represented in WordNet. Strings of several words that form a collocation or phrase found in WordNet are joined into one word form in a semantically tagged file and tagged as a single unit.

Performance Criteria

For the evaluation of WSD methods, two main performance criteria are being used:

- Precision: the fraction of assignments made that are correct
- Recall: the fraction of total word instances correctly assigned

As the method presented here assigns a sense to each to-be-disambiguated word in the text, both precision and recall values will be equal.

CHAPTER IV

RESULTS AND EVALUATION

In this chapter, the results of the experiments performed using different versions of the method that will be explained in the Subjects of the Experiments section are presented. After the results are given, the found values are evaluated and compared against other methods.

In the experiments, Semcor 1.6, texts semantically annotated with WordNet 1.6 senses, dataset is used. Experiments are made on a text extracted from Brown Corpus, including 1024 to be disambiguated words in total. The domain of the sample text is politics and law. The to-be-disambiguated words in the text are quite polysemous and difficult to disambiguate, with an average polysemy count of 6.4, ranging from 1 to 14 senses.

Different versions of the method are implemented in Java, using Java WordNet Library (JWNL). JWNL is a Java API for accessing WordNet's relational dictionary and the relationships stored in WordNet.

The results are compiled separately for each POS, namely noun, verb, adjective and adverb. For each POS, the percentage of correct sense assignments are calculated and presented in the results section. Also, the average success rate of the method is found by taking the weighted average of each POS' success rate.

Subjects of the Experiments

In the methodology chapter, the formula of similarity score calculation which forms the basis of this method is presented. As it can be seen in that formula, similarity score calculation involves two main parameters, namely the size of the sliding window and the distance. To observe the effects of these parameters, different versions of the method are released and tested with varying parameter values.

In the first version of the method, distance is measured based on words. The distance of each word in the window is used in the similarity calculation. The window size is taken as 10 in this version. This is the original version of the method and released to exploit the relatedness of the target word with the words surrounding it.

Second version of the method does not make use of window and window size. Actually window still exists but not like the original method, the window is the sentence itself. In other words, all words in a sentence are used in the similarity score calculation of a target word residing in that sentence. The distance measure is not applied in this version; instead all the words in a sentence are assumed to have equal distance value. The hypothesis is that, for a sentence to be coherent, the senses of the words in that sentence should be coherent within each other. The aim of this version is to prove this hypothesis by measuring the effect of sentence coherence on the method.

The third version of the method introduces a slight modification over the second version. The distance and window size parameters that are cancelled in the second version are included in the similarity score calculations like the first version. Unlike the first version, those parameters are not word based but sentence based. The window size

is set as 2, which means that the sentences before and after the sentence including the target word are counted. This version is released to increase the size of input and to measure the degree of relatedness of the sentences in the neighborhood of the target word.

The final version, fourth version of the method is an unsophisticated version which does not make use of any distance parameter. This version is used to show the effect of the distance parameter on the original (basic) method.

Results of the Experiments

Each version presented in the Subjects of the Experiments section are tested against the same text taken from the SemCor data. Among all the versions, experiments show that the second version of the method has the best performance while the fourth version has the worst performance. Detailed results of the experiments can be found in Table 4.

Table 4 Test Results

	Success Rate (%)				
Version	Noun	Verb	Adjective	Adverb	Total
Word Based Distance and Window Size (First)	53.32	25.10	55.88	63.33	44.53
Sentence Based Window, Distance Ommitted (Second)	53.94	29.80	55.15	60.00	46.78
Sentence Based Window Size and Distance (Third)	54.56	29.80	54.41	56.67	45.80
Word Based Window Size, Distance Ommitted (Fourth)	40.50	23.96	39.65	41.25	38.87

As it can be observed in Table 4, the results of the first three versions are very close. This shows that setting the window size based on words or sentence does not affect the success rate much. On the other hand, as the fourth version performed the worst clearly, it can be said that adding distance factor to the similarity score calculation affect the results in a positive manner.

Comparison of the Results Obtained

This method performed slightly worse than the other first three methods, when the results of this method are compared with the results of the other methods listed in Table 5. However, the performance difference between the last method (Tree Match) and this method is much bigger. The main reason for this difference is that the Tree Match method is using a manually created dependency data. This dependency data helps the method to deal with the words directly, not over the concepts that the words belong to. Thus, Tree Match method makes use of a specialized data, while the method presented here is using a generalized data which increases the error rate.

The success rate of the verbs appears to constitute the main difference between the other methods. The root cause of this difference is the lack of concept hierarchy in WordNet for verbs. As it is stated in the methodology, concept hierarchy is the main knowledge source in similarity score calculation. Since it is not possible to exploit concept hierarchy for verbs, the success rate of this method decreases dramatically affecting also the overall performance of the method.

Table 5 Performance Comparison

	Success Rate (%)				
Method	Noun	Verb	Adjective	Adverb	Overall
Unsupervised Large-Vocabulary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling	58.8	37.9	57.6	71.9	55.3
Unsupervised All-words Word Sense Disambiguation with Grammatical Dependencies	63.3	32.7	56.8	59.1	52.7
Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity	61.1	43.3	53	100	53.4
A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge (Tree Match Method)	NA	NA	NA	NA	73.6
This Method	53.9	29.8	55.1	60	46.8

CHAPTER V

CONCLUSIONS AND FUTURE WORK

Conclusion

This thesis shows that the sense of a word and the words surrounding that target word has to share a common context if the text is semantically coherent. According to this hypothesis a similarity score formulation is derived and each sense of a target word is ranked according to that score.

It is assumed that there are two main parameters exist in this formulation. The first parameter is the path length, which measures the concept hierarchy level where a concept match is found. The level and thus the path length are inversely proportional to the relatedness. In other words the relatedness between the concepts decreases as the path length increases. The second parameter is the distance of the target word to the word residing in the window size. Like path length, distance is inversely proportional to the relatedness of the words. As the distance between the words increase, it is assumed that the relation between the words decreases.

Different versions of the base method are released and tested against same Semcor text, to see the effects of these parameters. Results show us that, without removing those parameters from the formula but changing the way they contribute to the overall value did not change the result dramatically. On the other hand, omitting distance factor decreases the performance of the method clearly. This fact proves the positive effect of taking distance into account. According to the results, the second

version performed best which shows that the sliding window used in the calculations should be set to the sentence itself. The words in a sentence proved to be much related than the words outside the sentence.

Future Work

There are several possible improvements to be made on this work. First of all, the JWNL API that is used to retrieve the concept hierarchy of WordNet, does not support coordinate term retrieval. Coordinate terms are an important element of concept hierarchy as they offer a first order relationship between the words. Finding a coordinate term match between words, changes the value of the similarity score obviously. Currently there is no API developed to exploit coordinate term search. However as JWNL is an open source project, an extension over this API can be developed for coordinate term retrieval.

The sense assignment in this method is an iterative process. Each word is processed one by one and a corresponding sense is found and assigned during the iteration. When processing a target word, the sense assignments made before are not used. This information could be useful in determining the context of the text. The common concepts shared by the previous sense assignments may help the method to increase the accuracy of the sense assignments to be made for the following words.

Last but not least, the sense assignment process could be made considering all senses of the words residing in the window together. Currently, as it is stated above, the method processes the words one by one iteratively. Instead, a weighted graph could be

formed in which different senses of a word constitutes the vertices of the graph. A similarity score between each sense is calculated and assigned as the weight of the edge between them. Finally a graph search algorithm can be applied over this graph to calculate the maximum value between each vertex. The sense assignment giving the maximum edge weight value is selected as the correct one. This improvement is the most complex one among the others but offers a significant improvement on the results.

APPENDIX A

AN EXTRACT FROM SEMCOR DATA

```
<?xml version="1.0" encoding="UTF-8"?>
<wf cmd="ignore" pos="IN">in</wf>
<wf cmd="ignore" pos="DT">the</wf>
<wf cmd="done" pos="NN" lemma="appointment" wnsn="1" lexsn="1:04:00::">appointment</wf>
<wf cmd="ignore" pos="IN">of</wf>
<wf cmd="done" pos="NN" lemma="appraiser" wnsn="1" lexsn="1:18:01::">appraisers</wf>
<punc>,</punc>
<wf cmd="done" pos="NN" lemma="guardian" wnsn="1" lexsn="1:18:00::">guardians</wf>
<wf cmd="ignore" pos="CC">and</wf>
<wf cmd="done" pos="NN" lemma="administrator" wnsn="1" lexsn="1:18:00::">administrators</wf>
<wf cmd="ignore" pos="CC">and</wf>
<wf cmd="ignore" pos="DT">the</wf>
<wf cmd="done" pos="NN" lemma="awarding" wnsn="1" lexsn="1:04:00::">awarding</wf>
<wf cmd="ignore" pos="IN">of</wf>
<wf cmd="done" pos="NN" lemma="fee" wnsn="1" lexsn="1:21:00::">fees</wf>
<wf cmd="ignore" pos="CC">and</wf>
<wf cmd="done" pos="NN" lemma="compensation" wnsn="1" lexsn="1:21:00::">compensation
</wf>
<punc>.</punc>
</s>
</p>
<p pnum="17">
    <s snum="19">
        <wf cmd="ignore" pos="DT">The</wf>
        <wf cmd="done" pos="NN" lemma="jury" wnsn="1" lexsn="1:14:00::">jury</wf>
        <wf cmd="done" pos="VB" lemma="say" wnsn="2" lexsn="2:32:01::">said</wf>
        <wf cmd="ignore" pos="PRP">it</wf>
        <wf cmd="done" pos="VB" lemma="find" wnsn="4" lexsn="2:32:00::">found</wf>
        <wf cmd="ignore" pos="DT">the</wf>
        <wf cmd="done" pos="NN" lemma="court" wnsn="1" lexsn="1:14:00::">court</wf>
        <punc>``</punc>
        <wf cmd="done" pos="VBZ" ot="notag">has</wf>
        <wf cmd="done" pos="VB" lemma="incorporate" wnsn="1" lexsn="2:30:00::">incorporated
        </wf>
        <wf cmd="ignore" pos="IN">into</wf>
    </s>
</p>
```

APPENDIX B

SAMPLE EXECUTION TRACE STEPS 1-3

Step 1: Identify the to be disambiguated words

Original passage:

....in the appointment of appraisers, guardians and administrators and the awarding of fees and compensation. The jury said it found the court has incorporated into....

appointment | appraiser | guardian | administrator | fee | awarding | compensation | jury | say | find | court | has | incorporate

Step 2: Select to be disambiguated word from the list one by one

appointment | appraiser | guardian | administrator | fee | awarding | compensation | jury | say | find | court | has | incorporate

Step 3: Identify the words inside the specified window size (=10 for our case)

appointment | appraiser | guardian | administrator | fee | awarding | compensation | jury | say | find | court | has | incorporate

Since only 4 words exist before “fee”, 6 words coming after that word are included to complete the window size to 10.

APPENDIX C

SAMPLE EXECUTION TRACE STEPS 4-5

Step 4: Get the sense definitions of the to be disambiguated word (fee) from WordNet

1) a fixed charge for a privilege or for professional services

2) an interest in land capable of being inherited

Step 5: Tokenize the sense definitions one by one

1) fixed | charge | privilege | professional | services

1) interest | land | capable | being | inherited

APPENDIX D

SAMPLE EXECUTION TRACE STEP 6

Step 6: Calculate Score for First Sense

fixed vs appointment, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs appraiser, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs guardian, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs administrator, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs awarding, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs compensation, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs jury, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs say, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs find, Relation Type: , Relation Level: 0, Relation Score: 0.0
fixed vs court, Relation Type: , Relation Level: 0, Relation Score: 0.0
charge vs appointment, Relation Type: hypernym, Relation Level: 4, Relation Score: 0.2
charge vs appraiser, Relation Type: hypernym, Relation Level: 8, Relation Score: 0.11111111
charge vs guardian, Relation Type: hypernym, Relation Level: 6, Relation Score: 0.14285715
charge vs administrator, Relation Type: hypernym, Relation Level: 6, Relation Score: 0.14285715
charge vs awarding, Relation Type: hypernym, Relation Level: 7, Relation Score: 0.125
charge vs compensation, Relation Type: hypernym, Relation Level: 4, Relation Score: 0.2
charge vs jury, Relation Type: hypernym, Relation Level: 9, Relation Score: 0.1
charge vs say, Relation Type: hypernym, Relation Level: 2, Relation Score: 0.33333334
charge vs find, Relation Type: hypernym, Relation Level: 4, Relation Score: 0.2
charge vs court, Relation Type: hypernym, Relation Level: 8, Relation Score: 0.11111111
.....
fee vs appointment, Relation Type: hypernym, Relation Level: 16, Relation Score: 0.05882353
fee vs appraiser, Relation Type: hypernym, Relation Level: 21, Relation Score: 0.045454547
fee vs guardian, Relation Type: hypernym, Relation Level: 19, Relation Score: 0.05
fee vs administrator, Relation Type: hypernym, Relation Level: 19, Relation Score: 0.05
fee vs awarding, Relation Type: hypernym, Relation Level: 17, Relation Score: 0.055555556
fee vs compensation, Relation Type: hypernym, Relation Level: 6, Relation Score: 0.14285715
fee vs jury, Relation Type: hypernym, Relation Level: 15, Relation Score: 0.0625
fee vs say, Relation Type: , Relation Level: 0, Relation Score: 0.0
fee vs find, Relation Type: , Relation Level: 0, Relation Score: 0.0
fee vs court, Relation Type: hypernym, Relation Level: 15, Relation Score: 0.0625

Total Score for first sense of fee is: 0.11915591

APPENDIX E

SAMPLE EXECUTION TRACE STEP 7-8

Step 7: Calculate Score for Second Sense

interest vs appointment, Relation Type: hypernym, Relation Level: 4, Relation Score: 0.2
interest vs appraiser, Relation Type: hypernym, Relation Level: 15, Relation Score: 0.0625
interest vs guardian, Relation Type: hypernym, Relation Level: 13, Relation Score: 0.071428575
interest vs administrator, Relation Type: hypernym, Relation Level: 13, Relation Score: 0.071428575
interest vs awarding, Relation Type: hypernym, Relation Level: 7, Relation Score: 0.125
interest vs compensation, Relation Type: hypernym, Relation Level: 6, Relation Score: 0.14285715
interest vs jury, Relation Type: hypernym, Relation Level: 9, Relation Score: 0.1
interest vs say, Relation Type: , Relation Level: 0, Relation Score: 0.0
interest vs find, Relation Type: hypernym, Relation Level: 5, Relation Score: 0.16666667
interest vs court, Relation Type: hypernym, Relation Level: 9, Relation Score: 0.1
land vs appointment, Relation Type: hypernym, Relation Level: 4, Relation Score: 0.2
land vs appraiser, Relation Type: hypernym, Relation Level: 7, Relation Score: 0.125
land vs guardian, Relation Type: hypernym, Relation Level: 5, Relation Score: 0.16666667
land vs administrator, Relation Type: hypernym, Relation Level: 5, Relation Score: 0.16666667
land vs awarding, Relation Type: hypernym, Relation Level: 10, Relation Score: 0.09090909
land vs compensation, Relation Type: hypernym, Relation Level: 7, Relation Score: 0.125
land vs jury, Relation Type: hypernym, Relation Level: 5, Relation Score: 0.16666667
land vs say, Relation Type: hypernym, Relation Level: 10, Relation Score: 0.09090909
land vs find, Relation Type: , Relation Level: 0, Relation Score: 0.0
land vs court, Relation Type: hypernym, Relation Level: 5, Relation Score: 0.16666667
.....
fee vs appointment, Relation Type: hypernym, Relation Level: 16, Relation Score: 0.05882353
fee vs appraiser, Relation Type: hypernym, Relation Level: 21, Relation Score: 0.045454547
fee vs guardian, Relation Type: hypernym, Relation Level: 19, Relation Score: 0.05
fee vs administrator, Relation Type: hypernym, Relation Level: 19, Relation Score: 0.05
fee vs awarding, Relation Type: hypernym, Relation Level: 17, Relation Score: 0.055555556
fee vs compensation, Relation Type: hypernym, Relation Level: 6, Relation Score: 0.14285715
fee vs jury, Relation Type: hypernym, Relation Level: 15, Relation Score: 0.0625
fee vs say, Relation Type: , Relation Level: 0, Relation Score: 0.0
fee vs find, Relation Type: , Relation Level: 0, Relation Score: 0.0
fee vs court, Relation Type: hypernym, Relation Level: 15, Relation Score: 0.0625

Total score for second sense of fee is: 0.105383344

Step 8: Set the correct sense

Compare the score values of each sense and select the sense having maximum score:

1st sense score > 2nd sense score, so 1st sense is selected and it is the correct one.

REFERENCES

Mihalcea, R. & Moldovan, D. I. (2001). Ez.wordnet: Principles for automatic generation of a coarse grained WordNet.

Ide, N. & Veronis, J. (1998). Word Sense Disambiguation: The State of the Art.

Fellbaum, C. (1998). WordNet an Electronic Database, Cambridge: MIT Press.

WordNet Statistics (2010), Retrieved March 20, 2010, from http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone.

Leacock, C., Chodorow, M. & Miller, G. A. (1998). Using corpus statistics and WordNet relations for sense identification.

Hirst, G. & St-Onge, D. (1998). Lexical chains as representations of context in the detection and correction of malaproprisms.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. (1993). Introduction to WordNet: An On-line Lexical Database.

Resnik, P. (1995). Using information content to evaluate semantic similarity.
Jiang, Jian & Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy.

Lin, Dekang. (1998). An information theoretic definition of similarity.

Banerjee, S., Patwardhan, S. & Pedersen, T. (2003). Using measures of semantic relatedeness for word sense disambiguation.

Navigli, R. (2009). Word sense disambiguation: A survey.

Resnik, P. (1993). Selection and information: A class-based approach to lexical relationships.

Agirre, E. & Martínez, D. (2001). Learning class-to-class selectional preferences.

Chen, P., Ding, W., Bowes, C. & Brown, D. (2009). A fully unsupervised word sense disambiguation method using dependency knowledge.

Pedersen, T. (2000). A simple approach to building ensembles of Naive Bayesian classifiers for word sense disambiguation.

Ng, H. T. & Lee, H. B. (1996). Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach.

Rivest, R. (1987). Learning decision lists.

Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution

Schapire, R. E. & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods.

Leacock, C., Chodorow, M & Miller, G. (1998). Using corpus statistics and WordNet relations for sense identification.

Mihalcea, R. & Moldovan, D. (1999). An automatic method for generating sense tagged corpora.

Agirre, E. & Martínez, D. (2004). Unsupervised WSD based on automatically retrieved examples.

Schutze, H. (1998). Automatic word sense discrimination.

Lin, D. (1998). Automatic retrieval and clustering of similar words.

Veronis, J. (2004). Hyperlex: Lexical cartography for information retrieval.

Torres, S. & Gelbukh, A. (2009). Comparing Similarity Measures for Original WSD Lesk Algorithm.

Wu, Z. & Palmer, M. (1994). Verb semantics and lexical selection.