Semantic Relatedness Applied to

All Words Sense Disambiguation

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Jason Michelizzi

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

June 2005

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master's thesis by

JASON MICHELIZZI

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Ted Pedersen

Name of Faculty Adviser

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Measures of semantic relatedness have many practical applications including word sense disambiguation. This thesis presents an algorithm for all words sense disambiguation that incorporates measures of semantic relatedness.

This thesis explicates a variety of measures of semantic relatedness and semantic similarity that have been proposed. These are founded on a number of different principles. There are measures that rely on path length in a taxonomy, information content, finding lexical chains, and comparing the similarity of dictionary glosses. The strengths and weaknesses of the measures are discussed in detail. A method of handling multiple inheritance in a taxonomy is developed.

A method of word sense disambiguation that employs measures of semantic relatedness for an all words task is formulated. This algorithm is unique because it does not require training and is capable of disambiguating arbitrary text.

The algorithm has been evaluated using an assortment sense-tagged corpora, including SemCor and the test data for SENSEVAL-2 and SENSEVAL-3. The experimental results are compared to the performance of other existing all words sense disambiguation systems. The results are very promising, and the system presented in this thesis is highly competitive with other systems.

1 Introduction

In Natural Language Processing, it is often useful to quantify how related two words are. More precisely, we often wish to measure the relatedness of two *word senses*, where a word sense is a specific meaning of a particular words. For example, the word *ball* has several senses, including a round object used in games, a formal dance, and a pitch in baseball that is not a strike. A precise method for quantifying how similar two word senses are is called a measure of semantic relatedness.

Measures of semantic relatedness have been used for a variety of practical applications. Hirst & St-Onge [3] proposed a measure of semantic relatedness and used it for detecting and correcting malapropisms. They hypothesized that a spell checker could be enhanced to identify words that are spelled correctly but are used incorrectly.

They theorized that each word in a sentences should be semantically related to the nearby words. If a word is unrelated to its neighbors, then it is likely to be a malapropism.

They proposed an algorithm for detecting malapropisms where if a word is unrelated to the other words in a sentence, a search is undertaken for similarly spelled words. If one of those similarly spelled words is semantically related to the nearby words, then the algorithm suggests that word as a replacement for the original word, which may be a malapropism.

Another application of measures of semantic relatedness is *word sense disambiguation*. Word sense disambiguation is the process of determining which sense of a word is the intended sense in a particular context. For example, in the sentence, "John took his wife to the annual ball", a human can easily understand which sense of *ball* is intended. It would often be useful if software could also detect which sense of ball was the intended sense. Word sense disambiguation involves picking the desired sense of a word for a pre-defined set of words, which is typically a machine-readable dictionary, such as WordNet.

Lesk [6] presented a method of word sense disambiguation that compared the definitions of the ambiguous word to the definitions of the words near it. This method has formed the basis of the Adapted Lesk measure of semantic relatedness discussed in the next section and has also influenced the development of the word sense disambiguation algorithm discussed later in this thesis.

Closely related to word sense disambiguation is *word sense discrimination*. In word sense discrimination, uses of a word in different contexts are grouped together with the intent of grouping similar or synonymous uses of the word in the same sets. Word sense discrimination is different than word sense disambiguation because there is no pre-defined set of word senses in word sense discrimination.

Word sense disambiguation itself has many practical applications. One of the earliest motivations for studying word sense disambiguation was machine translation. It is often the case that different meanings of a single word in one language can correspond to multiple words in a different language. For example, the English word *port* can be translated as several different words in French depending upon its use. The sense of port that it somewhat synonymous with harbor is *port* in French, an opening in a ship is *sabord*, the left side of a vessel or aircraft is *bâbord*, and the red dessert wine is *Porto*.

Word sense disambiguation is itself often divided into distinct tasks. Sometimes, a word sense disambiguation method is concerned only with disambiguating a single ambiguous word in a given context. Other methods will attempt to disambiguate all the words in a text. This thesis focuses on “all words” disambiguation but also highlights “target word” methods that have been influential in the research presented here.

One popular approach to word sense disambiguation is to train a supervised machine learning algorithm (such as a Naive Bayes Classifier, or a Decision Tree) on a set of training data. A supervised machine learning approach has some significant disadvantages. A supervised approach requires a significant amount of manually annotated training data. Often, some training data is required for every word. Since there is relatively little manually annotated data available, supervised machine learning is often not practical for an all words word sense disambiguation system.

The approach in this thesis is considerably different than the supervised machine learning approach. The algorithm presented is not based directly upon statistics. Instead, it uses measures of semantic relatedness. The algorithm does not require any training data.

This thesis presents a number of contributions to existing work in measuring semantic relatedness and word sense disambiguation.

1. This thesis compares and contrasts various methods of measuring semantic relatedness by discussing the strengths and weaknesses of the different measures.

2. A method of handling of multiple inheritance by semantic similarity measures based on WordNet

is developed. Previous discussions of semantic similarity measures have given little treatment to multiple inheritance handling.

3. A novel method of "all words" disambiguation that employs measures of semantic relatedness is presented.

4. Freely available[1] word sense disambiguation software, WordNet::SenseRelate::AllWords, has been developed for this thesis. The experimental results presented in this thesis were obtained with this software, and the experiments can easily be re-created.

5. The word sense disambiguation algorithm is evaluated using nearly all relevant sense tagged corpora.

[1] The software is distributed free of charge and may be modified and/or redistributed under the terms of the GNU General Public License. The software is available from `http://senserelate.sourceforge.net`.

2 Measuring Semantic Relatedness

2.1 WordNet

WordNet [2] is a machine readable dictionary created at the Cognitive Science Laboratory at Princeton University. Unlike most dictionaries, WordNet contains only open-class words (nouns, verbs, adjectives, and adverbs). WordNet does not contain closed-class words such as pronouns, conjunctions, and prepositions.

WordNet groups sets of synonymous word senses into synonym sets or *synsets*. A word sense is a particular meaning of a word. For example, the word *walk* has several meanings; as a noun, it can refer to traveling by foot or it can refer to a base on balls in baseball. A synset contains one or more synonymous words senses. For example, {base on balls, walk, pass} is the synset for the second sense of the noun walk. The synset is the basic organizational unit in WordNet.

Each synset has a gloss (definition) associated with it. The gloss for the synset {base_on_balls, walk, pass} is "(baseball) an advance to first base by a batter who receives four balls." Many synsets also have an example in addition to the gloss. For example, "he worked the pitcher for a base on balls."

The sense numbers in WordNet are assigned according to the frequency with which the word sense occurs in the SemCor corpus (i.e., the first sense of a word is usually more common than the second). Word senses that do not appear in SemCor are assigned sense numbers in a random order. Word senses can be represented as strings in a specific format, using the word form, a single letter representing the part of speech, and a sense number, such as walk#n#2, which represents the second sense of the noun walk. The part of speech letter is *n* for nouns, *v* for verbs, *a* for adjectives, and *r* for adverbs. Terms consisting of more than one word are often joined by underscores instead of spaces. The synset in the previous paragraph can be written as {base_on_balls#n#1, walk#n#2, pass#n#1}.

Version 2.0 of WordNet contains approximately 152,000 unique words that together have a total of about 203,000 unique senses. These word senses are organized into over 115,000 different synsets. Of these synsets, almost 80,000 are noun synsets, 13,500 are verb synsets, 18,500 are adjective synsets, and 3,700 are adverb synsets.

WordNet defines relations between synsets and relations between word senses. A relation between synsets is a semantic relation, and a relation between word senses is a lexical relation. The distinction between lexical relations and semantic relations is somewhat subtle. The difference is that lexical relations are relations between *members* of two different synsets, but semantic relations are relations between two *whole* synsets.

Some examples of semantic relations are the hypernym, hyponym, meronym, and holonym relations. A hypernym of a synset is a generalization of that synset. The hyponym relation is the inverse of the hypernym relation. The hypernym and hyponym relations represent *is a* relationships between nouns. If A and B are nouns, and an A is a B, then A is a hyponym of B and B is a hypernym of A. For example, {organism, being} is the hypernym of {plant, fora} because a plant is an organism (cf. figure 1). Since the hyponym relation is the opposite of the hypernym relation, {plant, fora} is a hyponym of {organism, being}.

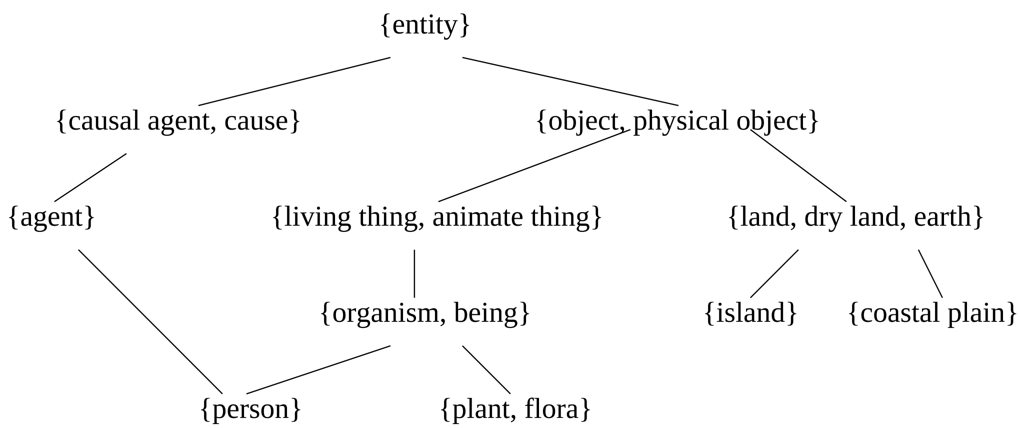


Figure 1: WordNet hypernyms

For verbs, the term troponym is used instead of hyponym. A troponym is a way of doing something else. For example, {soar} is a troponym of {fly, wing} because soaring is a way of flying (see figure 2). WordNet still uses the term hypernym as the inverse of tryponym; therefore, {fly, wing} is a hyponym of {soar}.

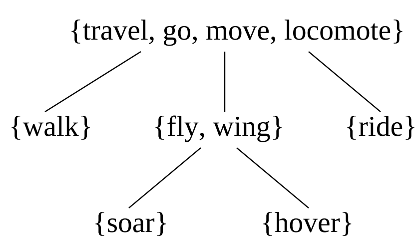


Figure 2: WordNet troponyms

A meronym is a word that is a part of whole (e.g., {wheel} is a meronym of {wheeled vehicle}). The inverse is the holonym relation (e.g., {wheeled vehicle} is a holonym of {wheel}).

The hypernym, hyponym, and troponym relations are particularly interesting. These relations form is-a taxonomies with the noun and verb synsets. These taxonomies have a tree-like structure. All noun and verb synsets belong to at least one taxonomy. Because multiple inheritance is allowed, some synsets belong to more than one taxonomy. There is no unique root node that links all noun synsets together or all verb synsets together. Instead, there are multiple taxonomies. In WordNet 2.0, there are nine noun taxonomies and 554 verb taxonomies.

Some examples of lexical relations are the antonym relation and the derived form relation. For example, the antonym of the tenth sense of the noun light (light#n#10) in WordNet is the first sense of the noun dark (dark#n#1). The synset to which light#n#10 belongs is {light#n#10, lighting#n#1}. Clearly it makes sense that light#n#10 is an antonym of dark#n#1, but lighting#n#1 is not a antonym of dark#n#1; therefore, the antonym relation needs to be a lexical relation, not a semantic relation because the relationship is between individual word senses, not whole synsets.

2.2 Path Length Similarity Measures

When the concepts expressed by two word senses are alike, the senses are said to be semantically similar. A measure of semantic similarity quantifies the degree of alikeness of two word senses. WordNet::Similarity [12] is a set of Perl modules that implements a number of measures of semantic similarity. The measures implemented are discussed in detail in the remainder of this chapter. The measures can be divided roughly into two groups: measures of semantic similarity and measures of semantic relatedness. Semantic similarity measures typically work with noun-noun or verb-verb pairs only nouns and verbs can be easily classified into is-a hierarchies. Semantic relatedness measures generally work on all (open class) parts of speech because they are not limited to is-a hierarchies.

In an is a taxonomy such as WordNet, a simple approach to measuring similarity is to treat the taxonomy as an undirected graph and use the distance in terms of path length between the two synsets as a measure of similarity. The greater the distance between two synsets, the less similar they are. In figure 1, for example, the synset {island} is closer to {land, dryland, earth} than it is to {living thing, animate thing}, so it is considered to be more similar to {land, dryland, earth} than {living thing, animate thing}.

The distance between two synsets can be measured using either either *edge counting* or *node counting*. In edge counting, the distance between two synsets is the number of links between the two synsets. In node counting, the distance between two synsets is the number of nodes along the shortest path between the two synsets, *including the end nodes representing the two synsets*. In figure 1, the distance between {plant, flora} and {living thing, animate thing} is two using edge counting and is three using node counting.

The depth of a synset is similar to the distance between two synsets. The depth of a synset is simply the distance between that synset and the root of the taxonomy in which the synset is located. The depth can be measured using either edge counting or node counting.

A slightly different notion of depth is the depth of a taxonomy. The depth of a taxonomy is the distance between the root of a taxonomy and a leaf node such that the distance is maximized.

WordNet presents an additional difficulty because some synsets have more than one hypernym. The synset containing the first sense of person, for example, has two hypernyms: {organism, being} and {causal agent, cause, causal agency} (see figure 1).

A shared parent of two synset is called a *subsumer*. The least common subsumer of two synset is the subsumer that does not have any children that are also subsumers of the two synsets. In figure 1, the subsumers of {living thing, animate thing} and {land, dry land, earth} are {object, physical object} and {entity}. The Least Common Subsumer (LCS) of two synsets is the most specific subsumer of the two synsets. The LCS of {living thing, animate thing} and {land, dry land, earth} is {object, physical object} since {object, physical object} is more specific than {entity}.

By default, the measures in the WordNet::Similarity package join all the noun taxonomies and all the verb taxonomies into one taxonomy for nouns and one for verbs by introducing an unique root node for each part of speech. However, this behavior can be "turned off" so that there is no unique root node for nouns or verbs.

When a unique root node is not being used, it is possible for two synsets from the same part of speech to have no common subsumer. In this case, the similarity measures cannot give a similarity score. Even when a unique root node is used, the measures cannot give a score for synsets from different parts of speech.

2.2.1 Path

The Path measure in WordNet::Similarity is a simple measure that uses the path length distance to measure the similarity of synsets. Within the package it is called WordNet::Similarity::path. The distance between two synsets is measured using node counting. Similarity is defined as

$$Sim_{path}(s_1, s_2) = \frac{1}{dist_{node}(s_1, s_2)} \tag{1}$$

such that $dist_{node}(s_1, s_2)$ is the distance between synset s_1 and synset s_2 using node counting. In figure 1, the distance between $\{$person$\}$ and $\{$living thing, animal$\}$ is three, so the similarity score is $1/3$.

Since node counting is used, the distance between two synsets is always greater-than or equal-to 1. For example, the distance between $\{$person$\}$ and $\{$person$\}$ is 1.

If a unique root node is being used, then there will always exist a path between any two noun synsets or any two verb synsets. If, however, a unique root node is not being used, then it is possible and in the case of verbs likely that there will not be a path between two synsets. In such a case, the similarity of the two synsets is undefined.

Since some synsets in WordNet have more than one hypernym, all measures of semantic similarity must take into account multiple inheritance. In the case of this measure, when there is more than one path between two synsets as a result of multiple inheritance, the shortest path is used. In figure 1, there is more than one path between $\{$person$\}$ and $\{$entity$\}$, but the shortest path is through $\{$agent$\}$ rather than $\{$organism, being$\}$.

One perceived drawback of a simple edge or node counting measure is that links in a taxonomy like WordNet can represent different distances between synsets. Some links may represent a large difference in meaning, while other links may represent only a small refinement in meaning. Typically, links that are high in a taxonomy (i.e., closer to the root), represent a greater semantic distance, and links low in a taxonomy represent a smaller semantic distance. There is a greater semantic distance $\{$object, physical object$\}$ and $\{$land, dry land, earth$\}$ than there is between $\{$island$\}$ and $\{$land, dry land, earth$\}$.

2.2.2 Wu & Palmer

Wu & Palmer [16] present a measure of semantic similarity based on distances and depths in a taxonomy. The measure takes into account the distance between each of two synsets, s_1 and s_2, and their LCS, s_3, as well as the distance between the LCS and the root of the taxonomy in which the synsets reside.

As originally formulated, the similarity of the two synsets is

$$Sim_{wup}(s_1, s_2) = \frac{2 \cdot dist_{node}(s_3, Root)}{dist_{node}(s_1, s_3) + dist_{node}(s_2, s_3) + 2 \cdot dist_{node}(s_3, Root)} \tag{2}$$

where $dist_{node}(s_1, s_3)$ is the distance between synset s_1 and synset s_3 using *node* counting.

Resnik [14] re-formulated this measure slightly:

$$Sim_{rwup}(s_1, s_2) = \frac{2 \cdot depth_{edge}(s_3)}{depth_{edge}(s_1) + depth_{edge}(s_2)} \tag{3}$$

where $depth_{edge}(s)$ is the depth of the synset s using *edge* counting.

The WordNet::Similarity package implements a slight reformulation of the Resnik reformulation by counting *nodes* instead of edges.

$$Sim_{adapted\_wup}(s_1, s_2) = \frac{2 \cdot depth_{node}(s_3)}{depth_{node}(s_1) + depth_{node}(s_2)} \tag{4}$$

Using node counting maintains consistency within the WordNet::Similarity package since the two previous measures use node counting. This measure is named WordNet::Similarity::wup.

In figure 1, the node depths of {person} and {plant, flora} are both 5, and the depth of their LCS, {organism, being}, is 4; therefore, their similarity score is $\frac{2 \cdot 4}{5+5} = 0.8$ using the adapted Wu & Palmer formulation.

In all three forms of the Wu & Palmer the maximum score is one. A score of one occurs only when the two synsets are the same (*i.e.*, self-similarity). A score of zero is possible in the first two formulations when the LCS is the root node. A score of zero is never possible in the third formulation because the depth of the root node using node counting is one. There is no clear minimum score in this formulation, but the minimum approaches zero. The advantage of this third formulation is that the score will always be greater than zero whenever there is a path connecting two synsets. Intuitively, if there is a path connecting two synsets, then the two synsets are not completely unrelated.

In the case of multiple inheritance, there will more than one possible similarity score. The measure uses the LCS that is deepest in the taxonomy (i.e., the LCS that is furthest from the root of the taxonomy). Choosing the LCS maximizes the value of the numerator in equation (4) and generally maximizes the overall similarity score in the same equation.

2.2.3 Leacock & Chodorow

Another measure of similarity based on distances and depths was proposed by Leacock & Chodorow [5]. It is implemented as the WordNet::Similarity::lch measure. The measure considers the depth of the taxonomy in which the synsets are located. They define similarity as

$$Sim_{lch} = -log(\frac{dist_{node}(s_1, s_2)}{2 \cdot D}) \tag{5}$$

where D is the depth of the taxonomy in which the synsets lie. The depth of a taxonomy is defined as the length of the shortest path between the root of the taxonomy and synset s_d such that there is no synset that has a greater shortest path length between it and the root than the length between the root and s_d.

Considering again the example of $\{$person$\}$ and $\{$plant, flora$\}$, the distance between the two synsets in figure 1 is 3, and the maximum depth is 5, the similarity score is $-log(3/(2 \cdot 5)) = -log(0.3) = 1.20$.

The measure of Leacock & Chodorow is profoundly affected by the presence or absence of a unique root node. If there is a unique root node, then there are only two taxonomies: one for nouns and one for verbs. The maximum depth of the noun taxonomy will be 18, and the maximum depth of the verb taxonomy will be 14 for WordNet 2.0. If, on the other hand, a unique root node is not used, then there are nine noun taxonomies and over 560 verb taxonomies. The maximum depths of the taxonomies varies considerably. The shallowest of the noun taxonomies is 9 and the deepest is 18. While the deepest verb taxonomy is has a depth of 14, the mean depth is 2.4 and the median depth is 2. Clearly the noun taxonomies are usually much deeper than the verb taxonomies.

2.3 Information Content Similarity Measures

Several measures of semantic similarity have been developed that use *information content*. Information content is a measure of *specificity*. The information content of a concept is inversely proportional to the frequency with which the concept is expected to occur. A concept that rarely occurs would have a high information content, and a concept that frequently occurs would have a low information content. A common word such as *be* would have a low information content, but a rare word such as *aliquot* would have a high information content.

Mathematically, the information content of a concept is

$$IC(c) = -logP(c) \tag{6}$$

where $P(c)$ is the probability of the concept c. A high information content means that the concept conveys a lot of meaning when it occurs in a text. A concept with a high information content is very specific, but a concept with a low information content is very general; therefore, information content is corresponds to specificity.

In semantic similarity measures, a concept is a synset, and the probability of a concept is the frequency of the concept divided by the number of concepts occurring in a corpus:

$$P(c) = frequency(c)/N \tag{7}$$

such that N is the number of concepts in the corpus from which the frequency counts were extracted. Frequency counts are propagated up the hierarchies so that the count of a concept is equal to the sum of the counts of its hyponyms plus the count of the concept itself.

A word can have multiple senses, and each sense of a word will belong to a different synset. If a corpus of sense-tagged text such as SemCor is available, then counting the frequencies of concepts is straightforward: for each sense-tagged word in the corpus, increment the count of the synset to which the word sense belongs. If sense-tagged text is not available, which is the usual case, then the situation is more difficult. Resnik [14] proposed that when a word with multiple senses occurs, the count for that word be divided equally among its senses. For example, if a word w has five senses, then the synsets corresponding to those five senses will have their frequency counts incremented by $1/5$. In WordNet::Similarity, the default behavior is to increment each synset by one. If w has five senses, the frequency count for each synset would be incremented by one. It is also possible to use Resnik's counting method in WordNet::Similarity.

The information content of a concept is zero only when the probability is one. This situation occurs only when the frequency of the concept is equal to the frequency of all the concepts in the taxonomy, which only happens when the concept is the unique root node of its taxonomy. If a taxonomy does not have a unique root node, then the information content can never be zero.

When the frequency count of a concept is zero (*i.e.,* when it does not occur in the corpus and none of its hyponyms occurs in the corpus) information content is undefined because the probability is zero. When the frequency count is undefined, the similarity measures that use information content also return an undefined value.

It is possible to avoid undefined information content values by using a smoothing scheme. WordNet::Similarity will optionally perform Laplace smoothing, where the frequency count of every possible event is incremented by one. In WordNet::Similarity, a every synset is a possible event; therefore, the frequency count of each synset is incremented by one.

The advantage of Laplace smoothing is that it avoids the possibility of undefined information content values. The disadvantage is that it can radically change the distribution of frequencies. Many, if not most, of the synsets in WordNet are unlikely to occur in any given corpus, and a small number of synsets are likely to occur repeatedly. Laplace smoothing can often shift too much of the overall probability mass to synsets that are rarely found in a corpus.

2.3.1 Resnik

Resnik [14] proposed a simple information content-based measure that considers the information content of the least common subsumer of the two synsets:

$$Sim_{res}(s_1, s_2) = IC(LCS(s_1, s_2)) \tag{8}$$

In cases where there is more than one subsumer of s_1 and s_2, the LCS is defined as the common subsumer with the greatest information content. The implementation of Resnik's measure used for this thesis is WordNet::Similarity::res.

The maximum similarity value for the Resnik measure occurs when the frequency of an LCS is one. When the frequency is one, the information content of the LCS is $logN$, where N is the sum of the frequencies of all the top-level nodes of the given part of speech.

One characteristic of the Resnik measure is that it is a rather coarse-grained measure. All pairs of synsets that with the same LCS will have the same similarity score. For example, {object, physical object} is the LCS of many synset pairs in figure 1, including {plant, flora} and {island}, {plant, flora} and {land, dry land, earth}, and {plant, flora} and {object, physical object}. Since these these pairs have the same LCS, by equation (8) they will have the exactly the same similarity score.

2.3.2 Lin

Another similarity measure developed by Lin [7] takes an information-theoretic approach based on there assumptions. Firstly, the more similar two concepts are, the more they will have in common. Secondly, the less two concepts have in common, the less similar they are. Thirdly, maximum similarity occurs when two concepts are identical. The measure of similarity defined by Lin meets these assumptions.

$$Sim_{lin}(s_1, s_2) = \frac{2 * IC(LCS(s_1, s_2))}{IC(s_1) + IC(s_2)} \tag{9}$$

The information content of the LCS will always be less-than or equal-to the information content of both s_1 and s_2; therefore, the similarity score can be at most one. The score is zero only if the information content of the LCS is zero. The score is undefined if the information contents of s_1 and s_2 are zero.

2.3.3 Jiang & Conrath

Jiang & Conrath [4] proposed a measure of semantic distance that uses information content.

$$Dist_{jcn}(s_1, s_2) = IC(s_1) + IC(s_2) - 2 * IC(LCS(s_1, s_2)) \tag{10}$$

This distance measure can be converted to a similarity measure by taking the multiplicative inverse of it:

$$Sim_{jcn}(s_1, s_2) = 1/Dist(s_1, s_2) \tag{11}$$

The WordNet::Similarity::jcn measure implements equation (11).

By defining a similarity measure as in equation (11), the distribution of scores is modified. Consider a case where the synset s_1 has the following distances from synsets s_2, s_3, and s_4.

		distance
s_1	s_2	2
s_1	s_3	3
s_1	s_4	4

The corresponding similarity scores are

		similarity
s_1	s_2	0.50
s_1	s_3	0.33
s_1	s_4	0.25

Note that difference between the distance scores for s_1-s_2 and s_1-s_3 is the same as s_1-s_3 and s_1-s_4; however, the same is not true of the similarity scores. This is not necessarily a problem. The best way to interpret the distance scores is that s_2 is closer to s_1 than s_3 is to s_1, and s_2 and s_3 are both closer to s_1 than s_4 is to s_1. Likewise for the similarity scores, s_2 is more similar to s_1 than s_2 is similar to s_1, etc.

An alternative method of converting the distance score to a similarity score would be to define similarity as

$$Sim'_{jcn}(s_1, s_2) = maxdistance - Dist(s_1, s_2) \tag{12}$$

however, the difficulty is determining a suitable value for $maxdistance$.

The maximum distance possible using equation 10 is not simple to calculate, and, in any case, would depend heavily upon the corpus used to calculate the information content values. Every corpus would have a different maximum distance.

2.4 Semantic Relatedness Measures

Semantic relatedness is a much broader notion than semantic similarity. When two word senses are semantically related, there exists some type of relationship between the word senses. For example, *tire* is related to *car*, but the two are not very similar since a tire is not a type of a car nor is a car a type of tire. Semantic similarity is a special case of semantic relatedness where the only relationship considered is the is-a relationship.

A measure of semantic relatedness quantifies the strength of the relationship between two word senses. WordNet::Similarity implements three measures of semantic relatedness.

2.4.1 Extended Gloss Overlaps (Adapted Lesk)

Lesk [6] proposed counting overlapping words in glosses to perform word sense disambiguation. An adaptation [1] of the Lesk measure finds overlaps in WordNet glosses to measure semantic relatedness. This adaptation is the WordNet::Similarity::lesk measure. The idea behind the measure is that the more words that are common in two glosses, the more related the corresponding synsets are. The implementation uses not only the glosses of the synsets but also uses the relations in WordNet to compare the glosses of closely related synsets.

When computing the relatedness of two synsets, s_1 and s_2, relation functions are used to determine which glosses are to be compared. The default relation functions used in WordNet::Similarity is shown in table 1. Table 2.4.1 explains what each function name means.

Each pair of functions specifies the glosses that are to be searched for overlaps. For example, the pair hype-hype means the the gloss for the hypernym of s_1 and the gloss for the hypernym of s_2 are searched for overlaps. The pair hype-hypo means that the gloss of the hypernym of s_1 is compared to the gloss of the hyponym(s) of s_2. If there is more than one hyponym for s_2, then the glosses for each hyponym are concatenated into a single gloss. The pair glos-hype means that the gloss of s_1 is compared to the gloss of the hypernym of s_2.

Each pair of relation functions generates a score, and the overall relatedness score is the sum of the scores for each relation function pair. The scoring mechanism takes into account both the number of words in the overlaps and the length of the overlaps. The motivation is that a four-word overlap (i.e., an overlap consisting of four consecutive words in both glosses) is more significant than four one-word overlaps. The score for a single relation function pair is the sum of the squares of the length of each overlap found:

$$pairscore = \sum_{i}^{\#overlaps} length^2(overlap_i) \tag{13}$$

The overall relatedness score is simply the sum of each of these pairwise scores:

$$Relatedness(s_1, s_2) = \sum_{j}^{\#pairscores} pairscore_j \tag{14}$$

Table 1: Default relations used with gloss measures

also	also	glos	sim	mero	attr
also	attr	glos	syns	mero	glos
also	glos	holo	also	mero	holo
also	holo	holo	attr	mero	hype
also	hype	holo	glos	mero	hypo
also	hypo	holo	holo	mero	mero
also	mero	holo	hype	mero	pert
also	pert	holo	hypo	mero	sim
also	sim	holo	mero	pert	also
attr	also	holo	pert	pert	attr
attr	attr	holo	sim	pert	glos
attr	glos	hype	also	pert	holo
attr	holo	hype	attr	pert	hype
attr	hype	hype	glos	pert	hypo
attr	hypo	hype	holo	pert	mero
attr	mero	hype	hype	pert	pert
attr	pert	hype	hypo	pert	sim
attr	sim	hype	mero	sim	also
example	example	hype	pert	sim	attr
example	glos	hype	sim	sim	glos
example	syns	hypo	also	sim	holo
glos	also	hypo	attr	sim	hype
glos	attr	hypo	glos	sim	hypo
glos	example	hypo	holo	sim	mero
glos	glos	hypo	hype	sim	pert
glos	holo	hypo	hypo	sim	sim
glos	hype	hypo	mero	syns	example
glos	hypo	hypo	pert	syns	glos
glos	mero	hypo	sim		
glos	pert	mero	also		

Using this formulation, one can observe that a single four-word overlap would receive a score of $4^2 = 16$, while four one-word overlaps would only receive a score of four.

Table 2: Abbreviations for Relation Functions

also	Also see relation
attr	Attribute relation
example	The example that is often found in a gloss
glos	The gloss without the example part
glosexample	The full gloss (both the definition and the example)
holo	Holonym relation
hype	Hypernym relation
hypo	Hyponym relation
mero	Meronym relation
pert	Pertainym relation
sim	Similar relation
syns	All the words in a synset

As a simple example of how the scoring works, consider the word senses congress#n#3 and parliament#n#1. Assume that only one relation function pair is used: glos-glos. The gloss for congress#n#3 is "a national legislative assembly." The gloss for parliament#n#1 is "a legislative assembly in certain countries." There is one overlap of two consecutive words, "legislative assembly." The relatedness score would be four.

2.4.2 Context Vectors

Another measure that uses similarities between glosses was proposed by Patwardhan [10] and is implemented as the WordNet::Similarity::vector_pairs measure; however, the approach used is quite different than the Adapted Lesk measure in its details.

Context vectors are often used in Information Retrieval for measuring the similarity of documents. A context vector is essentially a vector representation of the occurrences of a word in different contexts.

The vector measure of Patwardhan measures semantic relatedness of two word senses s_1 and s_2 by comparing the gloss vectors the corresponding to s_1 and s_2. The gloss vector for s_1 or s_2 is formed by adding together the word vectors of every word in the gloss for s_1 or s_2.

The measure works by creating a word vector for every content word found in the any of the WordNet glosses. A word vector for a word w represents the co-occurrence of every word in a WordNet gloss that appears with w. For example, the word *diabatic* occurs in just one gloss with the words *involve, transfer, heat,* and *process.* The word vector for diabatic may look like

$$\vec{diabatic} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

The non-zero entries in the vector represent the frequencies with which the words involve, transfer, heat, and process co-occur with diabatic. In this case, the each word only co-occurs once. The zero entries represent words that do not co-occur with diabatic. A real word vector would have thousands of dimensions and would be very sparse.

A gloss vector is formed by adding together the word vectors for each word found in a gloss. For example, the gloss vector for diabatic#a#1 is formed by adding together the word vectors for the content words that occur in the gloss for diabatic#a#1. The gloss for diabatic#a#1 is "involving a transfer of head; 'a diabatic process"'. The gloss vector is then

$$\vec{diabatic\#a\#1} = \vec{involve} + \vec{transfer} + \vec{heat} + \vec{diabatic} + \vec{process}$$

Finally, the relatedness of two word senses s_1 and s_2 is the cosine of the angle between the gloss vectors $\vec{s_1}$ and $\vec{s_2}$, which is equivalent to

$$Relatedness(s_1, s_2) = \frac{\vec{s_1} \cdot \vec{s_2}}{|\vec{s_1}||\vec{s_2}|} \tag{15}$$

One advantage of the gloss vector approach over the gloss overlap approach is that the vector method is not limited to finding exact matches between glosses.

The gloss vector measure is enhanced by the ability to consider glosses of closely related word senses. The pairs of relation functions listed in table 1 can be used to in finding the relatedness of word senses. Gloss vectors are formed for each individual pair of functions and a score is computed using equation (15) for each pair. The overall relatedness score is the mean of the individual function pair scores.

2.4.3 Hirst & St-Onge

Hirst and St-Onge [3] proposed a measure of relatedness based on finding lexical chains between synsets. The assumption is that a closely related pair of words will be linked by a chain that neither is too long nor has too changes in direction. The measure of Hirst & St-Onge has been implemented as the WordNet::Similarity::hso measure.

Unlike the three path-based measures of similarity discussed previously (Path, Leacock & Chodorow, and Wu & Palmer), the measure of Hirst & St-Onge considers many relations in addition to the hypernym-hyponym relations. The relations considered, along with their directions, are listed in table 3.

The measure considers relatedness to be either *extra-strong*, *strong*, or *medium-strong*. Extra-strong relatedness occurs only when two word senses are identical. A strong relation occurs when the two words senses are synonymous, when there is a single horizontal link between the synsets, or when one word sense is a compound of the other word sense and there exists any link between the synsets (e.g., dog and attack_dog). Medium-strong relatedness can have different weights and is defined as

$$weight = C - path\_length - k \cdot \#changes\_in\_direction \tag{16}$$

where C and k are constants. In WordNet::Similarity, the values for C and k are 8 and 1 respectively.

To illustrate how this measure of relatedness works, consider the word senses car#n#1 (an automobile) and jet#n#1 (an airplane with jet engines). There is a lexical chain in WordNet that links these two word senses as shown in figure 3. The word sense hood#n#5 is a meronym of car#n#1 since a hood is part of a car. Because an airplane can contain a hood, airplane#n#1 is a holonym of hood#n#5. Since a jet is a type of airplane, jet#n#1 is a hyponym of airplane#n#1.

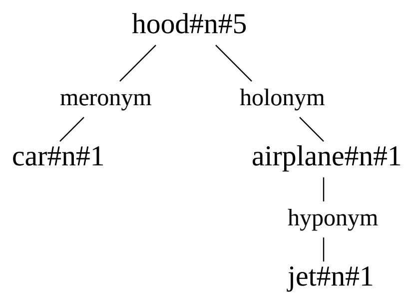


Figure 3: Example of Lexical Chains

The length of the chain linking car#n#1 and jet#n#1 is 3 (counting the relations that link the word senses). The meronym relation is an upward relation, and the holonym and hyponym relations are downward relations (see table 3); therefore, there is one change in direction. The relatedness score using equation (16) is

$$score = 8 - 3 - 1 \cdot 1 = 4$$

Table 3: Directions of WordNet Relations [3]

Relation	Direction
Also see	Horizontal
Antonymy	Horizontal
Attribute	Horizontal
Pertinence	Horizontal
Similarity	Horizontal
Cause	Down
Entailment	Down
Holonymy	Down
Hyponymy	Down
Hypernymy	Up
Meronymy	Up

2.5 Summary

The measures implemented in WordNet::Similarity represent a diverse approach to measuring semantic similarity and semantic relatedness. Table 2.5 illustrates the type of scores that can be expected from each measure in a few different circumstances.

Table 4: Comparison of measure results

	Self similarity		Immediate parent		More distant	
measure	dog#n#1	dog#n#1	dog#n#1	canine#n#2	dog#n#1	rodent#n#1
res	8.61		8.50		6.28	
lin	1.00		.994		.696	
jcn	$2.96 \cdot 10^7$		9.49		.183	
path	1		.5		.2	
wup	1		.960		.833	
lch	3.58		2.89		1.97	
lesk	5105		905		220	
vector_pairs	.785		.252		.135	
hso	16		4		3	

3 Word Sense Disambiguation

Words often have multiple senses. Human beings generally are very good at selecting which sense of a word is intended in a particular context. For example, the word *ball* can mean several things, including a round object used in games or a formal dance. When ball occurs in a sentence, humans can usually comprehend which sense is intended (e.g., “The batter hit the ball” vs. “He dressed up for the ball”).

It is sometimes useful if software could distinguish between different senses of a word. Selecting the correct sense of a word in a given context is word sense disambiguation (WSD).

Patwardhan *et al.* [11] describe a method of disambiguating target words using the WordNet::Similarity measures of semantic relatedness. The method described is for “lexical sample” situation, where only one target word is disambiguated in a context. The target word and its context is often annotated using XML, SGML, or another similar format.

A more general approach is to disambiguate all of the open-class words in a context. An “all words” approach is often more useful for certain natural language tasks. An extension of the Patwardhan *et al.* approach is discussed here. The algorithm has been implemented as a set of Perl modules and scripts, and is distributed as WordNet::SenseRelate::AllWords. The name is consistent with the convention for naming Perl packages and reflects that it implements an extension of the SenseRelate algorithm for an all words approach and incorporates the WordNet knowledge base.

3.1 Algorithm

Each word in the input text is disambiguated separately, starting with the first word and working left to right. At each stage, the word being disambiguated is called the target word, and the surrounding words form the context window.

The algorithm computes a score for each sense of the target word. To find the score for a sense of the target word (i.e., target sense), the target sense is compared to the senses of the context words using a semantic relatedness measure. The algorithm finds the sense of each context word that is most related to the target sense. The score for each target sense is sum of the relatedness scores between the target sense and each most related context sense. Once the algorithm finds a score for each sense of the target word, the sense with the greatest score is assigned to the target word.

Equation (17) presents the algorithm in a compact mathematical form. For each target word, the algorithm finds

$$best\_sense = \mathrm{argmax}_i \sum_{j=t-c_l}^{t+c_r} \mathrm{max}_k relatedness(s_{ti}, s_{jk}) \tag{17}$$

where s_{ti} is sense i of word t (i.e., the target word), and s_{jk} is sense k of word j (i.e., a context word). The context window consists of the c_l words to the left of the target word and the c_r words to the right of the target word.

The pseudo-code for the algorithm is in figure 1. The $relatedness()$ function on line 14 is a call to one of the previously described semantic relatedness or semantic similarity measures. Any of the measures can be used for the algorithm, but some measures may be better suited than others.

The threshold in line 18 can be used to require that all relatedness scores contributing to the overall score for a candidate word sense be above a certain value. This threshold may be useful in eliminating noise.

3.1.1 The Context Window

A window size of N means that there are a total of N words in the context window, including the target word. Since the target word is not compared to itself, a window of size N means that $N - 1$ words will be compared to the target word. The N words are chosen so that there are an equal number of context words on each side of the target word if N is an odd number. If N is an even number, there will be one more word on the left side of the target word than on the right. For example, if the window size is four, there will be two words to the left of the target word and one word to the right.

It is believed that the words preceding a specific word are more likely to influence the meaning of a word than the words following the word; therefore, when the window size is even and the number of words on each side of the target must be unequal, the algorithm chooses an extra word to the left.

If the target word occurs near the beginning or end of a sentence, the total number of words in the window may actually be less than the specified window size. For example, if the target word is the first word in a sentence, there will be no words to the left of the target word in the window. No attempt is made to expand the window to the right to compensate for the lack of words to the left.

Algorithm 1 Word Sense Disambiguation algorithm

function disambiguate-all-words
for all w_t in input **do**
  best-sense $\leftarrow$ disambiguate-single-word (w_t)
  display best-sense
end for
end procedure

function disambiguate-single-word (w_t) returns sense s
for all s_{ti} of target word w_t **do** $\{s_{ti}$ is the ith sense of the target word $w_t\}$
  $score_i \leftarrow 0$
  for all w_j in context window **do**
    if $j = t$ **then**
      next w_j
    end if
    for all s_{jk} of w_j **do** $\{s_{jk}$ is the kth sense of word $w_j\}$
      $temp\text{-}score_k \leftarrow$ relatedness (s_{ti}, s_{jk})
    end for
    $best\text{-}score \leftarrow$ max $temp\text{-}score$
    if $best\text{-}score > threshold$ **then**
      $score_i \leftarrow score_i + best\text{-}score$
    end if
  end for
end for
return $sense_i$ s.t. $score_i > score_j$ for all j in $\{s_{t0}, \cdots s_{tN}\}$
end function

When the window size is two and the target word is the first word in the sentence, the algorithm would normally be unable to assign a sense number to that word since the context window is empty. In this case, we assign sense number one to that word. In WordNet, the sense numbers of words are ranked by their frequency in SemCor; therefore, the first sense of a word is usually the most likely sense of the word. Senses of a word that do not appear in SemCor are assigned sense numbers in a random order.

Adjusting the size of the context window will affect how the algorithm performs. A large context window (e.g., more than two words on each side of the target word) will result in more words being considered for each sense of the target word. This may increase the chances of finding a sense of the target word that is closely related to one or more context word senses. A small context window (e.g., one word on each side of the target word) will result in very few words being considered for each sense of the target word. This might result in the algorithm finding better matches. Words farther way from the target word are less likely to be related to words close to the target word; therefore, using a small window may result in fewer unrelated words being used. Additionally, using a small context window will result in the algorithm running much faster since there are fewer comparisons to perform.

3.1.2 Fixing Word Senses

In the preceding algorithm, when a sense number is assigned to a word, other senses of the word are still considered when disambiguation subsequent words in a sentence. In the sentence, "dogs hate cats", if dogs is assigned sense number 1, other senses of dogs would normally still be considered when disambiguating hate and cats.

In WordNet::SenseRelate::AllWords, there is an option to fix the senses as they are assigned so that other senses of a word are not considered once a word has been disambiguated. Not only is this scheme logically pleasing, but it also is more efficient because fewer relatedness values need to be computed.

3.1.3 Part of Speech Tags

The WordNet::SenseRelate::AllWords package is capable of accepting text that is part of speech tagged and is also capable of accepting untagged text. Part of speech tagged text is text where the words have been assigned a tag indicating the part of speech to which they belong. One popular tag set is the Penn

Treebank tag set. When using the Penn Treebank tags, each word is followed by a slash and then one or more characters indicating the part of speech. For example, in the sentence, “The/DT dog/NN ran/VBD home/NN”, the *DT* tag indicates a determiner, the *NN* tag denotes a singular noun, and the *VBD* tag is used for a verb in the past tense.

Using part of speech tagged text can improve the performance of the algorithm in terms of speed and precision. When part of speech tagged text is used fewer senses of the target words will need to be considered when a word can has senses belonging to more than one part of speech. The word *walk*, for example, can be either a noun or a verb. Since fewer senses will be considered, using tagged text will result in the algorithm running faster. Precision will often increase because there are fewer incorrect senses of a target word that the algorithm could incorrectly project as the correct sense.

There are several good automatic part of speech taggers. One of the most popular is the rule based tagger of Eric Brill.

4 Experimental Data

Evaluating the performance of Word Sense Disambiguation algorithms requires manually sense-tagged corpora. A sense-tagged corpus is a body of text where the words have been assigned sense tags using a dictionary (or a similar knowledge base) as a source of sense information. A program that implements an algorithm is used to disambiguate the words in a corpus, ignoring the sense tags. The sense tags assigned by the program are compared to the manually-assigned sense tags.

There are relatively few sense-tagged corpora available. The largest is SemCor [9], which is a subset of the Brown Corpus. There are also corpora for the SENSEVAL-2 and SENSEVAL-3 competitions.

Although hand-tagging text is a reliable (albeit time-consuming) method of assigning senses to words in a corpus, it is not perfect. Humans often disagree about which sense of a word is intended in a text. Words are often used ambiguously or even incorrectly. In the case of puns, the ambiguity is intentional. Since one human's judgments are unlikely to correspond exactly to another's, no word sense disambiguation algorithm can be expected to perform perfectly.

4.1 SemCor

SemCor consists of texts selected from the Brown Corpus and consists of about 360,000 words of which about 221,000 are sense tagged. The untagged words are not closed class words and do not appear in WordNet. The Brown Corpus itself contains over a million words. The Brown Corpus was created at Brown University in 1964 and contains a wide selection of general English texts. SemCor is broken into 186 separate files.

Since the Brown Corpus was created in 1964, the corpus does not reflect changes in the English language during the past forty years. WordNet, on the other hand, was created in the 1990s and is updated frequently to reflect changes in the language.

The words in SemCor were tagged using WordNet version 1.6. WordNet has undergone changes since version 1.6. The version used in this thesis is version 2.0. The sense tags were mapped from the original WordNet 1.6 senses to WordNet 2.0 sense; however, many senses have been added and deleted since WordNet 1.6. In cases where the original sense from 1.6 could not be mapped to a 2.0 sense, the word is omitted from the experiments in this thesis.

The experiments in this thesis used a subset of SemCor. Five files were selected for use in the experiments.[2] The five files contain 9,442 words; 5,800 of those words are open class words, but only 5,012 words could be mapped to valid senses in WordNet 2.0.

Table 4.1 shows the distribution of the open class words in SemCor by part of speech. The table also shows the breakdown of the subset of SemCor used in the experiments. Note that only the open class words that could be mapped to WordNet 2.0 senses are included in the table.

Table 5: Breakdown of Test Data by Part of Speech

	nouns	verbs	adjectives	adverbs	total
SemCor	87,422	47,701	34,835	19,709	189,667
SemCor5	2,408	1,407	760	437	5,012
SENSEVAL-2	1,061	510	456	288	2,315
SENSEVAL-3	892	725	348	14	1,979

4.2 SENSEVAL

The goal of SENSEVAL is to evaluate the strengths and weaknesses of various Word Sense Disambiguation systems. There have been three competitions held so far, and a fourth is planned for the near future. The competitions include a number of different tasks, and the SENSEVAL organization develops a set of test data to use for evaluating the systems. There are several different types of data sets creating, including one for a *lexical sample* task and one for an *all words* task. This thesis uses the all words data from SENSEVAL-2 and SENSEVAL-3.

The SENSEVAL-2 data used in this thesis is the English all words data. The data is a small subset of the Penn Treebank corpus and consists of 4,873 words taken from *Wall Street Journal* articles. 2,473 of the 4,873 words are open-class words, and 2,315 of the open-class words are found in WordNet 2.0.

The SENSEVAL-3 data used is the English all words data. Like the SENSEVAL-2 all words data, this data is a small subset of the Penn Treebank corpus. The data is split into three separate files. Two of the files are

[2] The files are br-a01, br-a02, br-k18, br-m02, br-r05.

from *Wall Street Journal* articles and one is from a work of fiction. Of the 4,883 words in the set, 2,081 are open-class words and 1,979 of the open class words are found in WordNet 2.0.

Table 4.1 shows the distribution of the open class words in the SENSEVAL-2 and SENSEVAL-3 data by part of speech. Only words with valid WordNet 2.0 senses are included in the table. Notice the SENSEVAL-3 data has a significantly higher percentage of verbs than the other data sets.

5 Experimental Results

The experiments presented in this thesis were conducted using the data sets presented in the previous section. The results are reported using precision, recall, and the F Measure. Precision is the number of correct responses divided by the number of attempts. Recall is the number of correct responses divided by the total number of words. The F Measure combines precision and recall into a single quantity.

$$F = \frac{1}{\alpha\frac{1}{p} + (1-\alpha)\frac{1}{r}} \tag{18}$$

The constant α allows the precision and recall to be weighted. To weight precision more heavily than recall, α is chosen to be greater than 0.5. To weight recall more heavily than precision, α is chosen to be less than 0.5. If α is exactly 0.5, then precision and recall are weighted equally, and the previous equation simplifies to

$$F = \frac{2 \cdot p \cdot r}{p + r} \tag{19}$$

In this case, the value of the F Measure is simply the harmonic mean of p and r. In this thesis, precision and recall are weighted equally; therefore, the values for the F Measure that are reported use equation (19).

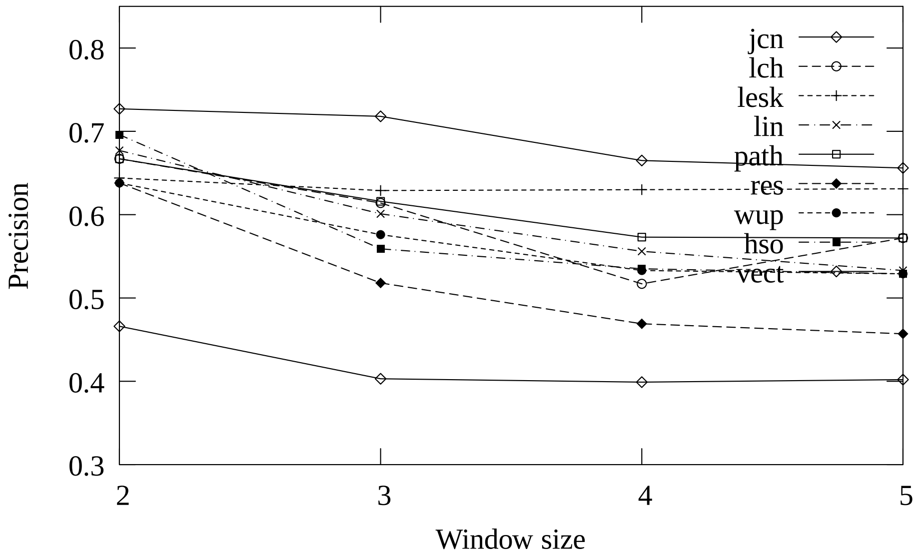
5.1 SemCor Results

The experiments conducted used a subset subset of the SemCor corpus. The subset consists of five randomly selected files from the corpus. The entire corpus consists of 186 files.

When conducting experiments, values need to be selected for several parameters. The most universal, and perhaps the most significant, is the window size. The window can consist of two or more words. The window size is the total number of words in the window. A window of one would only have a target word but no context words; therefore, the minimum size used is two.

In general, a small window size results in high precision but low recall. Words close to the target word are more likely to be related to the target words than words that are farther away.

Figure 4: Precision for SemCor results



Experiments were conducted using a window size of 2, 3, 4, and 5. The results are shown in table 5.1. The results are shown in terms of precision (P), recall (R), and the F Measure (F).

The random entry serves as a baseline for analyzing the results. Under the random scheme, a word sense is chosen at random from the possible senses for a word. If a word has ten senses, then there will be a 0.1 percent chance of guessing the correct sense. A value of 0.414 indicates that the mean number of senses for a word in WordNet is about 2.4. The random guessing scheme does not use a window; however, the values are shown under the columns for the different window sizes for comparison purposes.

The entry labeled *sense1* in table 5.1 corresponds to guessing sense number one for each word. The words in WordNet are numbered (approximately) according to frequency. The first sense of a word in WordNet is usually represents the most frequent use of that word; therefore, always guessing the first sense of a word is an approximation to guessing the most frequent sense of a word. It can be observed that the algorithm never does better than guessing the first sense of a word. Like the random guessing scheme, the sense1 scheme does not use a context window; however, the values are shown under the columns for the different window sizes for comparison purposes.

Figure 4 is a graph of precision versus window size for each of the measures. The graph illustrates a slight decrease in precision for most of the measures when window size is increased. Figure 5 is a graph of recall

Figure 5: Recall for SemCor results

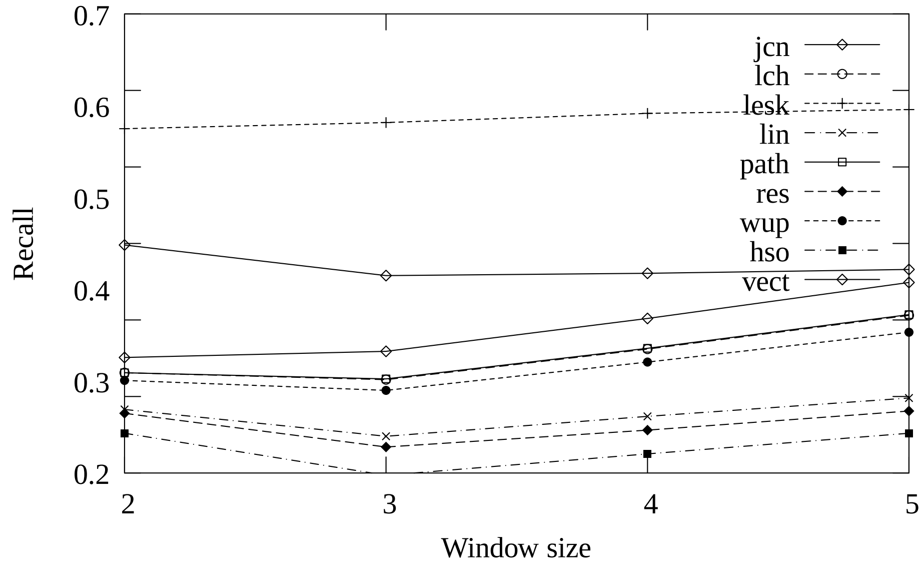


Figure 6: F Measure for SemCor results

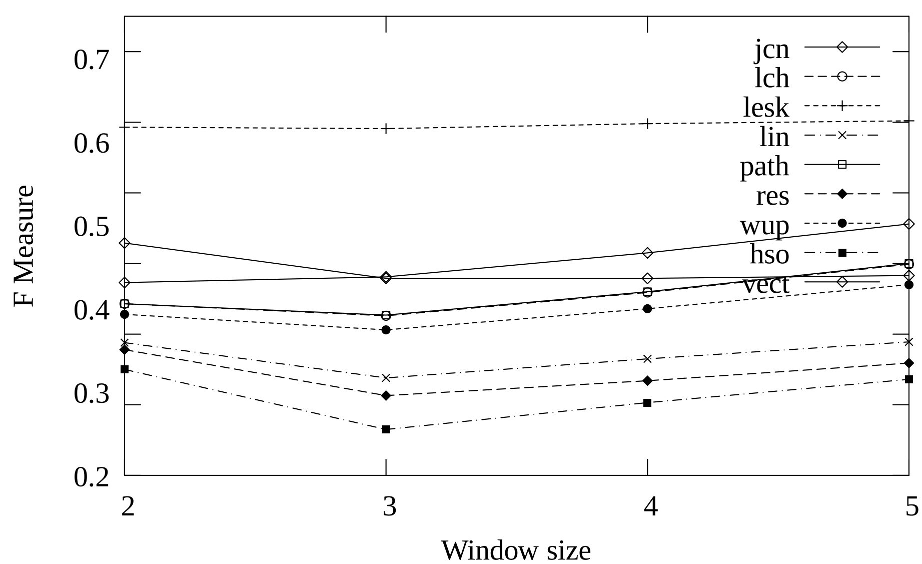


Table 6: SemCor results

	window 2			window 3			window 4			window 5		
measure	P	R	F	P	R	F	P	R	F	P	R	F
hso	.696	.152	.250	.559	.097	.165	.535	.125	.203	.529	.152	.236
jcn	.727	.251	.373	.718	.259	.381	.665	.302	.415	.656	.349	.456
lch	.667	.231	.343	.614	.222	.326	.571	.262	.359	.572	.306	.399
lesk	.644	.550	.593	.629	.558	.591	.630	.570	.598	.631	.575	.602
lin	.677	.183	.288	.601	.148	.238	.556	.174	.265	.533	.198	.289
path	.667	.231	.343	.616	.223	.327	.573	.263	.360	.572	.307	.400
random	.414	.414	.414	.414	.414	.414	.414	.414	.414	.414	.414	.414
res	.638	.178	.278	.518	.134	.213	.469	.156	.234	.457	.181	.259
sense1	.764	.764	.764	.764	.764	.764	.764	.764	.764	.764	.764	.764
vector_pairs	.466	.398	.429	.403	.358	.379	.399	.361	.379	.402	.366	.383
wup	.638	.221	.328	.576	.208	.306	.533	.245	.336	.529	.284	.370

versus window size for each measure. Most of the measures experience an increase in recall as window size increases. Finally, figure 6 is a graph of the F Measure versus window size for all measures. Most of the measures show an increase in the value of the F Measure when the window size is increased.

5.2 SENSEVAL-2 Results

Similar experiments were conducted using the SENSEVAL-2 data. The results are shown in table 5.2. The results are similar to the results using SemCor, but the values tend to be somewhat lower.

Like the SemCor results, the algorithm never does better than the *sense1* method. The *sense1* results are lower than the results for SemCor, but the results using the random-guessing method are about the same.

5.3 SENSEVAL-3 Results

Similar experiments were also conducted on SENSEVAL-3. The results are shown in table 5.3. The results using the SENSEVAL-3 data are similar to the SENSEVAL-2 results. The values using the sense1 and random methods are almost the same. The recall values for the six similar measures are noticeably better and are

Table 7: SENSEVAL-2 results

	window 2			window 3			window 4			window 5		
measure	P	R	F	P	R	F	P	R	F	P	R	F
hso	.609	.108	.183	.473	.061	.108	.455	.086	.145	.466	.111	.180
jcn	.644	.162	.259	.621	.162	.257	.574	.191	.287	.578	.233	.332
lch	.587	.168	.261	.527	.161	.247	.513	.199	.287	.519	.237	.325
lesk	.617	.516	.562	.610	.537	.571	.604	.543	.572	.608	.552	.579
lin	.606	.110	.186	.481	.066	.116	.456	.089	.148	.461	.113	.182
path	.587	.168	.261	.528	.162	.248	.515	.200	.288	.517	.236	.324
random	.432	.432	.432	.432	.432	.432	.432	.432	.432	.432	.432	.432
res	.573	.127	.208	.439	.088	.147	.431	.116	.183	.437	.143	.215
sense1	.679	.679	.679	.679	.679	.679	.679	.679	.679	.679	.679	.679
vector_pairs	.500	.419	.456	.472	.417	.443	.475	.427	.450	.471	.429	.449
wup	.582	.166	.258	.504	.154	.236	.492	.191	.275	.497	.227	.317

similar to the recall values for SemCor.

5.4 Results by Part of Speech

It is known that the semantic relatedness measures work better with some parts of speech than others. The six similarity measures work only with nouns and verbs and do not work at all with adjectives and adverbs. Even the other measures of relatedness are known to work better with certain parts of speech.

Results using only nouns as the target word in the SENSEVAL-2 data are in table 5.4. In comparison to the results for all parts of speech in table 5.2, the numbers are noticeably higher for all three metrics: precision, recall, and the F Measure. In particular, the recall is significantly better for the six similarity measures.

The six similarity measure have much better recall when only nouns are scored because those measures only work with nouns and verbs. Because these measures do not work with adjectives and adverbs, the recall is automatically reduced when using these six measures.

Table 5.4 shows results using only verbs. The results are noticeably worse than the results for nouns. The recall for the six similarity measures is particularily bad. The similarity measures rely on having a good hierarchical sturcture linking the synsets. The verb hierarchies in WordNet are very different than the noun

Table 8: SENSEVAL-3 results

measure	window 2 P	window 2 R	window 2 F	window 3 P	window 3 R	window 3 F	window 4 P	window 4 R	window 4 F	window 5 P	window 5 R	window 5 F
hso	.559	.131	.212	.338	.056	.096	.342	.081	.131	.360	.108	.166
jcn	.604	.213	.315	.569	.201	.297	.532	.245	.335	.526	.290	.374
lch	.565	.206	.302	.501	.186	.271	.496	.244	.327	.497	.291	.367
lesk	.581	.499	.537	.556	.490	.521	.561	.502	.530	.569	.515	.541
lin	.561	.152	.239	.439	.094	.155	.409	.120	.186	.404	.147	.216
path	.565	.206	.302	.501	.186	.271	.491	.241	.323	.494	.289	.365
random	.434	.434	.434	.434	.434	.434	.434	.434	.434	.434	.434	.434
res	.526	.152	.236	.385	.096	.154	.367	.125	.186	.368	.154	.217
sense1	.693	.693	.693	.693	.693	.693	.693	.693	.693	.693	.693	.693
vector_pairs	.462	.398	.428	.395	.349	.370	.399	.357	.377	.396	.359	.376
wup	.539	.196	.287	.473	.176	.256	.456	.224	.300	.469	.274	.346

Table 9: SENSEVAL-2 results for nouns

measure	window 2 P	window 2 R	window 2 F	window 3 P	window 3 R	window 3 F	window 4 P	window 4 R	window 4 F	window 5 P	window 5 R	window 5 F
hso	.596	.132	.216	.500	.113	.184	.499	.159	.241	.518	.206	.295
jcn	.681	.239	.354	.671	.306	.420	.633	.363	.461	.632	.430	.512
lch	.579	.246	.345	.556	.308	.396	.550	.380	.449	.551	.437	.487
lesk	.689	.597	.640	.678	.649	.663	.667	.661	.664	.670	.668	.669
lin	.601	.140	.227	.524	.133	.212	.510	.185	.272	.510	.231	.318
path	.579	.246	.345	.558	.309	.398	.553	.382	.452	.550	.436	.486
random	.467	.467	.467	.467	.467	.467	.467	.467	.467	.467	.467	.467
res	.560	.177	.269	.472	.180	.261	.473	.238	.317	.478	.289	.360
sense1	.739	.739	.739	.739	.739	.739	.739	.739	.739	.739	.739	.739
vector_pairs	.545	.471	.505	.515	.493	.504	.520	.515	.517	.504	.502	.503
wup	.579	.246	.345	.537	.298	.383	.535	.369	.437	.534	.424	.473

Table 10: SENSEVAL-2 results for verbs

measure	window 2			window 3			window 4			window 5		
	P	R	F	P	R	F	P	R	F	P	R	F
hso	.444	.063	.110	.280	.027	.049	.212	.033	.057	.229	.047	.078
jcn	.418	.090	.148	.415	.096	.156	.354	.114	.172	.393	.165	.232
lch	.468	.102	.167	.383	.090	.146	.349	.114	.172	.394	.167	.234
lesk	.385	.373	.379	.387	.380	.383	.378	.373	.375	.380	.378	.379
lin	.419	.061	.106	.245	.024	.044	.157	.022	.038	.193	.033	.056
path	.468	.102	.167	.383	.090	.146	.349	.114	.172	.389	.165	.232
random	.271	.271	.271	.271	.271	.271	.271	.271	.271	.271	.271	.271
res	.397	.061	.106	.211	.024	.043	.172	.029	.050	.202	.045	.074
sense1	.455	.455	.455	.455	.455	.455	.455	.455	.455	.455	.455	.455
vector_pairs	.314	.306	.310	.281	.276	.278	.274	.271	.272	.278	.276	.277
wup	.441	.096	.158	.342	.080	.130	.301	.098	.148	.352	.149	.209

hierarchies. There are only nine noun hierarchies, and they tend to be very deep. There are hundereds of verb hierarchies, and most of them are only a few concepts deep. The poor quality of the verb hierarchies severely limites the effectiveness of the similarity measures.

Table 5.4 shows results using only the adjectives. The results for the six similarity measures are not shown. For those measures, the recall would always be zero and the precision would be undefined.

Table 5.4 shows results using only the adverbs. The results for the six similarity measures are not shown. For those measures, the recall would always be zero and the precision would be undefined.

5.5 Other Parameters

The Adapted Lesk measure can use a stoplist to remove stopwords from the superglosses before they are searched for overlaps. Removing stopwords may reduce the likelihood of superfluous matches.

Results of using the Adapted Lesk measure on the SENSEVAL-2 data with and without a stoplist are shown in table 13. The results indicate that using a stoplist reduces the quality of the results. Reduced recall is not unexpected since reducing the number of words in the superglosses will reduce the probability of finding an overlap; however, reduced precision is somewhat unexpected. One explanation is that the most frequent

Table 11: SENSEVAL-2 Results for Adjectives

measure	window	P	R	F
hso	2	.917	.087	.159
hso	3	.833	.020	.039
hso	4	.714	.040	.076
hso	5	.565	.052	.095
lesk	2	.789	.434	.560
lesk	3	.806	.439	.568
lesk	4	.807	.441	.570
lesk	5	.805	.443	.571
random	-	.447	.447	.447
sense1	-	.728	.728	.728
vector_pairs	2	.575	.318	.410
vector_pairs	3	.581	.316	.409
vector_pairs	4	.594	.325	.420
vector_pairs	5	.598	.329	.424

Table 12: SENSEVAL-2 Results for Adverbs

measure	window	P	R	F
hso	2	.778	.170	.279
hso	3	1.00	.007	.014
hso	4	1.00	.007	.014
hso	5	.667	.007	.014
lesk	2	.641	.601	.620
lesk	3	.588	.559	.573
lesk	4	.596	.573	.584
lesk	5	.613	.604	.608
random	-	.562	.562	.562
sense1	-	.774	.774	.774
vector_pairs	2	.620	.590	.605
vector_pairs	3	.567	.545	.556
vector_pairs	4	.561	.545	.553
vector_pairs	5	.585	.583	.584

senses of a word also tend to have the longest glosses. By removing stop words, the bias in favor of the most frequent sense is reduced.

Table 13: SENSEVAL-2 using Lesk with and without a stoplist

	window	P	R	F
lesk	2	.617	.516	.562
lesk	3	.610	.537	.571
lesk	4	.604	.543	.572
lesk	5	.608	.552	.579
lesk.stop	2	.582	.410	.481
lesk.stop	3	.582	.462	.515
lesk.stop	4	.578	.489	.530
lesk.stop	5	.588	.514	.548

The Lesk measure is also affected by the relations used in forming superglosses. The default set of relations used is shown in table 1. Table 15 shows a reduced set of relations also used in experiments. The results of these experiments are shown in table 14.

Reducing the set up relations used reduces both precision and recall. Using the reduced set of relations results in smaller superglosses. Having smaller superglosses means that the Lesk measure may run faster because less text needs to be searched for overlaps, but the difference in speed appears to be negligible.

Table 14: SENSEVAL-2 using Lesk with different relations

	window	P	R	F
lesk	2	.617	.516	.562
lesk	3	.610	.537	.571
lesk	4	.604	.543	.572
lesk	5	.608	.552	.579
lesk.reduced	2	.547	.457	.498
lesk.reduced	3	.529	.466	.496
lesk.reduced	4	.521	.468	.493
lesk.reduced	5	.515	.468	.490

Three similarity measures use information content (Jiang & Conrath, Resnik, and Lin). The information content values derived from a corpus. The default information content data is derived from SemCor using the sense tags. Another corpus is the British National Corpus. Unlike SemCor, the British National Corpus is not sense-tagged.

Table 16 shows the results of using information content data derived from the British National Corpus with the Jiang & Conrath measure on the five SemCor files mentioned previously. There are small differences in the results when using Laplace (Add1) smoothing or Resnik counting, and the use of smoothing or Resnik counting does seem to increase both precision and recall, but the differences are not large.

Table 17 shows the results of using the same information content data with the Lin measure on the same five SemCor files. The differences in this case are smaller than the differences with the Jiang & Conrath measure.

5.6 Fixed Mode

5.6.1 SemCor

The experimental results for the five SemCor files are shown in table 5.6.1. The results are very similar to the results in table 5.1 on page 34; although, the numbers are consistently slightly lower in fixed mode.

5.6.2 SENSEVAL-2

The results using the SENSEVAL-2 data in the fixed more are shown in table 5.6.2. The results are quite similar to the results without fixed mode (*cf.* table 5.2). Most of the measures show a very slight decrease for all three metrics.

5.6.3 SENSEVAL-3

Table 5.6.3 shows the results using the SENSEVAL-3 data in fixed mode. Like the SemCor and SENSEVAL-2 results, the numbers are slighly lower in most cases. Overall, the use of fixed mode does not appear to make a significant difference.

Table 15: Lesk's reduced relation set

also	also	glos	hype	mero	holo
also	attr	glos	mero	mero	hype
also	glos	glos	pert	mero	mero
also	holo	glos	sim	mero	pert
also	hype	glos	syns	mero	sim
also	mero	holo	also	pert	also
also	pert	holo	attr	pert	attr
also	sim	holo	glos	pert	glos
attr	also	holo	holo	pert	holo
attr	attr	holo	hype	pert	hype
attr	glos	holo	mero	pert	mero
attr	holo	holo	pert	pert	pert
attr	hype	holo	sim	pert	sim
attr	mero	hype	also	sim	also
attr	pert	hype	attr	sim	attr
attr	sim	hype	glos	sim	glos
example	example	hype	holo	sim	holo
example	glos	hype	hype	sim	hype
example	syns	hype	mero	sim	mero
glos	also	hype	pert	sim	pert
glos	attr	hype	sim	sim	sim
glos	example	mero	also	syns	example
glos	glos	mero	attr	syns	glos
glos	holo	mero	glos		

Table 16: Results using the BNC with the Jiang & Conrath Measure

Counting	Smoothing	window	precision	recall	F Measure
Normal	None	2	.664	.228	.339
Normal	Add1	2	.665	.230	.341
Resnik	None	2	.668	.229	.341
Resnik	Add1	2	.670	.232	.345
Normal	None	3	.619	.221	.326
Normal	Add1	3	.624	.226	.332
Resnik	None	3	.627	.224	.330
Resnik	Add1	3	.632	.228	.335
Normal	None	4	.572	.260	.358
Normal	Add1	4	.576	.264	.362
Resnik	None	4	.582	.264	.363
Resnik	Add1	4	.572	.260	.358
Normal	None	5	.567	.301	.393
Normal	Add1	5	.572	.307	.400
Resnik	None	5	.570	.303	.396
Resnik	Add1	5	.567	.301	.393

Table 17: Results using the BNC with the Lin Measure

Counting	Smoothing	window	precision	recall	F Measure
Normal	None	2	.639	.177	.277
Normal	Add1	2	.641	.179	.280
Resnik	None	2	.638	.177	.277
Resnik	Add1	2	.640	.180	.280
Normal	None	3	.539	.138	.220
Normal	Add1	3	.543	.141	.224
Resnik	None	3	.540	.138	.220
Resnik	Add1	3	.545	.141	.224
Normal	None	4	.492	.162	.244
Normal	Add1	4	.497	.166	.249
Resnik	None	4	.496	.164	.246
Resnik	Add1	4	.492	.162	.244
Normal	None	5	.474	.185	.266
Normal	Add1	5	.481	.190	.272
Resnik	None	5	.482	.188	.270
Resnik	Add1	5	.474	.185	.266

Table 18: Results using the BNC with the Resnik Measure

Counting	Smoothing	window	precision	recall	F Measure
Normal	Add1	2	.638	.178	.278
Normal	Add1	3	.521	.135	.214
Normal	Add1	4	.475	.158	.237
Normal	Add1	5	.462	.138	.262
Resnik	Add1	2	.639	.179	.280
Resnik	Add1	3	.522	.135	.214
Resnik	Add1	4	.475	.158	.237
Resnik	Add1	5	.462	.183	.262
Resnik	None	2	.639	.179	.280
Resnik	None	3	.522	.135	.214
Resnik	None	4	.472	.157	.236
Resnik	None	5	.462	.183	.262
Normal	None	2	.638	.178	.278
Normal	None	3	.521	.135	.214
Normal	None	4	.475	.158	.237
Normal	None	5	.462	.183	.262

Table 19: Results using SemCor in fixed mode

	window 2			window 3			window 4			window 5		
measure	P	R	F	P	R	F	P	R	F	P	R	F
hso	.703	.147	.243	.559	.095	.162	.537	.119	.195	.527	.148	.231
jcn	.721	.248	.369	.726	.262	.385	.654	.279	.391	.648	.331	.438
lch	.652	.225	.334	.637	.231	.339	.570	.244	.342	.569	.291	.385
lesk	.638	.543	.587	.627	.557	.590	.628	.569	.567	.629	.573	.600
lin	.675	.174	.277	.612	.143	.232	.555	.155	.242	.533	.178	.267
path	.652	.225	.334	.637	.231	.339	.571	.245	.343	.574	.294	.389
res	.649	.172	.272	.539	.131	.211	.486	.144	.222	.454	.161	.238
vector_pairs	.464	.397	.428	.403	.358	.379	.398	.360	.378	.395	.360	.377
wup	.630	.218	.324	.602	.218	.320	.538	.231	.323	.530	.271	.359

Table 20: Results using SENSEVAL-2 in fixed mode

	window 2			window 3			window 4			window 5		
measure	P	R	F	P	R	F	P	R	F	P	R	F
hso	.617	.106	.181	.468	.060	.106	.455	.081	.138	.464	.108	.175
jcn	.638	.159	.254	.629	.163	.259	.581	.182	.277	.593	.231	.332
lch	.582	.166	.258	.549	.168	.257	.515	.186	.273	.529	.229	.320
lesk	.621	.517	.564	.611	.537	.572	.602	.540	.569	.609	.552	.579
lin	.616	.108	.184	.497	.063	.112	.489	.085	.145	.494	.110	.180
path	.582	.166	.258	.549	.168	.257	.511	.185	.272	.528	.229	.319
res	.585	.125	.206	.467	.085	.144	.435	.103	.166	.457	.134	.207
vector_pairs	.499	.419	.456	.477	.421	.447	.472	.425	.447	.473	.430	.450
wup	.574	.164	.255	.529	.162	.248	.496	.179	.263	.515	.223	.311

Table 21: Results using SENSEVAL-3 in fixed mode

	window 2			window 3			window 4			window 5		
measure	P	R	F	P	R	F	P	R	F	P	R	F
hso	.568	.127	. 208	.331	.054	.093	.330	.073	.120	.357	.103	.160
jcn	.610	.214	.317	.588	.208	.307	.524	.222	.312	.511	.263	.347
lch	.566	.206	.302	.522	.194	.283	.495	.224	.308	.478	.263	.339
lesk	.587	.504	.542	.557	.491	.522	.560	.501	.529	.569	.515	.541
lin	.573	.147	.234	.455	.091	.152	.406	.106	.168	.416	.136	.205
path	.566	.206	.302	.523	.195	.284	.494	.223	.307	.476	.261	.337
res	.546	.148	.233	.393	.090	.146	.359	.106	.164	.366	.135	.198
vector_pairs	.451	.388	.417	.390	.344	.366	.395	.354	.373	.395	.358	.376
wup	.540	.196	.288	.493	.183	.267	.458	.207	.285	.456	.251	.324

6 Related Work

This thesis has presented a novel method of all words word sense disambiguation. This section describes the work on which this thesis is founded and describes related methods of all words disambiguation.

6.1 Gloss Overlaps for Word Sense Disambiguation

Much of the foundation of the work presented in this thesis was laid by Lesk [6]. Lesk proposed a method of word sense disambiguation that compared the glosses of an ambiguous word to the glosses of the surrounding context. This method is the foundation of the Adapted Lesk semantic relatedness measure.

Lesk's method of word sense disambiguation was founded upon the assumption that the correct sense of an ambiguous word is likely to have a similar gloss to the glosses of the surrounding words. Each sense of the ambiguous word receives a score based on the number of words in the gloss of each sense that are also found in the glosses of the surrounding words.

As an example of how his algorithm works, consider the word *engine*. Engine has three sense in WordNet.

1. motor that converts thermal energy to mechanical work
2. something used to achieve a purpose; "an engine of change"
3. a wheeled vehicle consisting of a self-propelled engine that is used to draw trains along railway tracks

If the word *railcar* occurs near the word engine, then the gloss of railcar can be compared to the gloss of engine (there is only one sense of railcar in WordNet). In WordNet, the gloss of railcar is

> a wheeled vehicle adapted to the rails of railroad; "three cars had jumped the rails"

The word *vehicle* is found in both the gloss of railcar and the third sense of engine. The word *a* occurs in the gloss of railcar and in the glosses for the second and third senses of engine.

The score for the first sense of engine would be one since there is one common word, *to*, between its gloss and the gloss of railcar. The score for the second sense of engine would be one since there is one common word, *a*. The score for the third sense of engine is two since there are two shared words, *vehicle* and *a*[3]. Since the third sense of engine has the highest score, it is most likely to be the correct sense.

One limitation on the original Lesk algorithm is that glosses tend to be very short, and even if two words are closely related, it is often unlikely that there will be a shared word in the two glosses.

6.2 Target Word Semantic Relatedness Disambiguation

Significant work [10][11][13] has been done in using the WordNet::Similarity measures for target word disambiguation. The algorithm used in this previous work is the basis of the algorithm presented in chapter 3 of this thesis.

Target word disambiguation differs from all words disambiguation in several important ways. In target word disambiguation, there is only one ambiguous word under consideration. The word is often set apart in a text using a markup language such as XML. An all words system must be able to handle generic text, often without any special formatting.

In target word disambiguation, the target word is rarely, if ever, at the beginning or end of a sentence (or other window of context). An all words system needs to be able to disambiguate words at the beginning or end of a context window.

While a target word system fits many uses quite well, an all words system is usually more generic and flexible.

6.3 Set-based Semantic Relatedness Disambiguation

Another situation where word sense disambiguation is useful is when a single ambiguous word is associated with a set of words. The set of words can form a context to use for disambiguating the ambiguous word. For example, the word *Java* has several meanings. It can refer to an island in Indonesia, coffee, or a programming language.

If an instance of the word Java were associated with a set of words like {Jakarta, Indian Ocean, island},

[3] In practical applications, stop words such as *a*, *the*, *that*, etc. would often be removed before the glosses are compared. The scores would then be 0, 0, and 1 for the three senses.

we known that the first sense of Java would be most appropriate. If the set of words were {object-oriented, class, platform-independent, program}, we would know that the third sense would be most appropriate.

The algorithm used in this thesis can be easily adapted to the situation described here. In this situation, there is no window of context surrounding the target word. Instead, the context for the target word is the associated set of words. The sense of target word that is projected to be the correct sense is the sense that is most related to the words in the associated set. Equation (17) can be reformulated slightly. If there are N context words in the associated set of words, the best sense of the target word w is

$$best\_sense = \mathrm{argmax}_i \sum_{j=1}^{N} \mathrm{max}_k relatedness(t_i, s_{jk}) \tag{20}$$

where t_i is a sense of the target word and s_{jk} is sense k of the jth context word.

A set of Perl modules called WordNet::SenseRelate::WordToSet has been developed that implements equation (20) as an algorithm. This software, like WordNet::SenseRelate::AllWords, is freely available.[4]

.

6.4 Identifying the Predominant Sense

McCarthy, *et al.* [8] proposed a method of discovering the predominant sense of nouns using the semantic relatedness measures. They note that the first sense method described in the previous section often performs better than other word sense disambiguation systems that consider the context in which a word occurs.

While the sense numbers assigned to words in WordNet are ordered according to their frequency in SemCor, many words do not occur in SemCor. In addition, it may be useful to "tune" the first sense heuristic to a specific domain, such as medical text.

The authors use the semantic relatedness measures in WordNet::Similarity as a part of their algorithm to identify predominant word senses. The algorithm is based upon a prevalence score that takes into account a *distributional similarity score* between a target word and a neighbor of the target word as well as the semantic similarity score between a sense of the target word and a sense of a neighbor.

[4]The software is distributed free of chard and may be modifi ed and/or redistributed under the terms of the GNU General Public License. The software is available from `http://senserelate.sourceforge.net`

The prevalence score for a sense of the target word is

$$pscore = \sum_{n_j \in N_w} dss(w, n_j) \cdot \frac{msrs(ws_i, n_j)}{\sum_{ws'_i \in senses(w)} msrs(ws'_i, n_j)} \tag{21}$$

where w is the target word, n_j is neighbor word j, and ws_i is sense i of the target word. The function $senses(w)$ gives all the senses of word w, $dss(w, n)$ is the distributional similarity score of w and n, and $msrs(ws_i, n_j)$ is the maximum semantic relatedness score between sense i of the target word and any sense of the neighbor word n_j (i.e., the relatedness score between ws_i and the sense of n_j to which ws_i is most related).

The denominator of the fraction is essentially the sum of all the maximum semantic relatedness scores for a given word sense.

The distributional similarity scores are based on the occurrence of words in a selected corpus. Different types of corpora can be selected in order to tune the method to different domains. For example, if the method were to be applied to identifying predominant senses in a medical domain, a corpus of medical text could be used to determine the distributional similarity scores.

The distributional similarity score measures how often a word w and its neighbor n occur in the same context. It is defined as

$$dss(w, n) = \frac{\sum_{(r,x) \in T(w) \cap T(n)} I(w, r, x) + I(n, r, x)}{\sum_{(r,x) \in T(w)} I(w, r, x) + \sum_{(r,x) \in T(n)} I(n, r, x)} \tag{22}$$

where

$$I(w, r, x) = log \frac{P(x|n \cap r)}{P(x|r)} \tag{23}$$

where r is a grammatical relation between words w and x. The tuple $< w, r, x >$ represents the occurrence of w and x in the relation r in a corpus. For example, r could represent a direct object relationship such that w is the direct object of the verb x. $T(w)$ is a set of co-occurrence types (r, x) where w occurs in relation r with x.

6.5 SENSEVAL-2 English All Words

Twenty-two systems competed in the SENSEVAL-2 English All Words competition. Table 6.5 presents the results from the systems.[5] Also shown in the same table are two results from table 5.2. The results of the algorithm in this thesis are competitive with the performance of the entries in SENSEVAL-2. The precision of the algorithm when used the the Jiang & Conrath measure and a window size of two is .644, which is better than all but two of the systems. The F Measure score when used with the Adapted Lesk measure and a window size of 5 is .579, which would have been the fourth best at SENSEVAL-2.

6.6 SENSEVAL-3 English All Words

Sixteen teams submitted a total of twenty-six systems for the SENSEVAL-3 competition. The results were reported by Snyder & Palmer [15]. The performance of the systems is shown in table 6.6. Once again, the performance of the SenseRelate algorithm is shown under two different configurations, and the results for the SenseRelate algorithm are reproduced from table 5.3. The precision when using the Jiang & Conrath measure is better than fifteen of the twenty-six systems, and the overall performance of the algorithm when using the Adapted Lesk measure is better than about half of the systems.

[5] The results are available from `http://www.sle.sharp.co.uk/senseval2/`.

Table 22: Comparison of SENSEVAL-2 Results

System name	precision	recall	F Measure
SMUaw	.690	.690	.690
CNTS-Antwerp	.636	.636	.636
Sinequa-LIA - HMM	.618	.618	.618
SenseRelate-lesk-w5	**.608**	**.552**	**.579**
UNED - AW-U2	.575	.569	.572
UNED - AW-U	.556	.550	.553
UCLA - gchao2	.475	.454	.464
UCLA - gchao3	.474	.453	.463
CL Research - DIMAP	.416	.451	.433
CL Research - DIMAP (R)	.451	.451	.451
UCLA - gchao	.500	.449	.473
Universiti Sains Malaysia 2	.360	.360	.360
IRST	.748	.357	.483
Universiti Sains Malaysia 1	.345	.338	.341
Universiti Sains Malaysia 3	.336	.336	.336
BCU - ehu-dlist-all	.572	.291	.386
Sheffield	.440	.200	.275
Sussex - sel-ospd	.566	.169	.260
SenseRelate-jcn-w2	**.644**	**.162**	**.259**
Sussex - sel-ospd-ana	.545	.169	.258
Sussex - sel	.598	.140	.227
IIT 2	.328	.038	.068
IIT 3	.294	.034	.061
IIT 1	.287	.033	.059

Table 23: Comparison of SENSEVAL-3 Results

System name	precision	recall	F Measure
GAMBL-AW-S	.651	.651	.651
SenseLearner-S	.651	.642	.646
Koc University-S	.648	.639	.643
R2D2: English-all-words	.626	.626	.626
Meaning-allwords-S	.625	.623	.624
Meaning-simple-S	.611	.610	.610
LCCaw	.614	.606	.610
upv-shmm-eaw-S	.616	.605	.610
UJAEN-S	.601	.588	.594
IRST-DDD-00-U	.583	.582	.582
University of Sussex-Prob5	.585	.568	.576
Univeristy of Sussex-Prob4	.575	.550	.562
University of Sussex-Prob3	.573	.547	.560
DFA-Unsup-AW-U	.557	.546	.551
SenseRelate-lesk-w5	**.569**	**.515**	**.541**
KUNLP-Eng-All-U	.510	.496	.503
IRST-DDD-LSI-U	.661	.496	.567
upv-unige-CIAOSENSO-eaw-U	.581	.480	.526
merl.system3	.467	.456	.461
upv-unige-CIAOSENSO2-eaw-U	.608	.451	.517
merl-system1	.459	.447	.453
IRST-DDD-09-U	.729	.441	.550
autoPS-U	.490	.433	.460
clr04-aw	.506	.431	.465
autoPSNVs-U	.563	.354	.435
merl.system2	.480	.352	.406
SenseRelate-jcn-w2	**.604**	**.213**	**.315**
DLSI-UA-all-Nosu	.343	.275	.305

7 Conclusions

The experimental results indicate that the algorithm presented in this thesis is highly competitive with other all words word sense disambiguation systems. Most other high-performing systems are based on supervised machine learning algorithms, but the algorithm presented here takes a much different approach since it is founded on a knowledge base (i.e., WordNet).

The results show that varying the window size in the algorithm does not have a clear, consistent effect. With the six semantic similarity measures, a small window size tends to result in high precision but low recall, and increasing the window size reduces precision but boosts recall. Overall, the F Measure tends to increase slightly as the window size increases for those six measures. These results indicate that a few surrounding words provide sufficient context.

For the Adapted Lesk measure, there is a similar trend, but the differences are much smaller. A smaller window does tend to result in higher precision than a larger window, but the differences are small. Similarly, a larger window results in better recall, but the difference is not large.

For the Vector and Hirst & St-Onge measures, there is no consistent trend. Context vectors have been used successfully when the vectors are formed from very large corpora. The corpora used for the Vector measure is the glosses in WordNet, and there may not be enough data for the Vector measure to work optimally. The measure of Hirst & St-Onge relies on finding lexical chains linking synsets. It is very likely that there are not enough relations in WordNet to find chains between related word senses.

For some applications, high precision may be more desirable than having good recall. In such a case, a small window size should be used with the six semantic similarity measures and the Adapted Lesk measure. In other applications, the overall effect may be more important than having high precision. In that case, a larger window size should be used with those seven measures.

In applications where high precision is required, the best course may be to use the Jiang & Conrath measure with a very small window size; however, the Jiang & Conrath measure is only effective with nouns.

If an application demands a balance of precision and recall, then the Adapted Lesk measure should be used. The Adapted Lesk measure is also best with words that are not nouns.

The six similarity measures have a very clear advantage in terms of execution time. For the SENSEVAL-2 data, a single run of the algorithm using a similarity measure took less than three minutes.[6] A single run of the algorithm with the Adapted Lesk measure took between twenty and ninety minutes. A run using the Hirst & St-Onge measure typically took between four and nine hours. Finally, a run using the Vector measure took the longest and required between twelve and twenty-five hours. In all cases, the window size had a significant impact on execution time because a larger window equates to a significant increase in the number of semantic relatedness calculations required.

The use of the "fixed mode" has a slight negative effect on performance but may have a slight advantage by reducing execution time. The fixed mode requires fewer calculations of a semantic relatedness score.

This thesis has discussed a number of measures of semantic relatedness and semantic similarity and highlighted the strengths and weaknesses of each one. A method of handling multiple inheritance in a taxonomy for measures of semantic similarity has been presented.

This thesis has presented a method of all words disambiguation that uses measures of semantic relatedness and requires no training. Software has been developed for this thesis that implements the word sense disambiguation algorithm, and the software has been used to evaluate the algorithm using almost all known applicable sense tagged corpora.

[6] The experiments in this thesis were run on two different machines. Both machines are Linux systems using the 2.4.21 kernel. One machine has dual Intel Xeon 2 GHz processors and 2 GB of main memory. The second machine has dual Intel Xeon 3 GHz processors and 3 GB of main memory

8 Future Work

There is room for improvement in measuring semantic relatedness. As noted earlier in this thesis, the original measure proposed by Jiang & Conrath was a distance measure, not a similarity measure. WordNet::Similarity converts the distance measure into a similarity measure by taking the multiplicative inverse of the distance (cf. equation (11)). Equation (12) illustrates an alternative method of converting the original distance measure into a relatedness measure.

A better smoothing scheme for the information content measures is possible. Instead of using Laplace smoothing, there are a few other possibilities. One method of smoothing is to Lidstone's law. Lidstone smoothing is similar to Laplace smoothing, but instead of adding one to the frequency count of each possible event, a value λ is added, where $0 < \lambda < 1$. Often the value for λ is 0.5. One difficulty of this method of smoothing is choosing an appropriate value for λ.

From the experimental results, it is known that some measures work better than others for certain situations in word sense disambiguation. The similarity measures, and the Jiang & Conrath measure in particular, work well when the target word is a noun and there are other nouns in the context window. The Adapted Lesk measure works with all parts of speech, but the precision is not as high as the precision with the Jiang & Conrath measure.

A possible improvement would be to allow two or more measures to be used in conjunction to exploit the strengths of each one. One difficulty is that the measures give very different types of scores, and a score of 1 from one measure has a very different meaning than a score of 1 from another measure (cf. table 2.5)

Another improvement to the word sense disambiguation algorithm would be to make the selection of context words more flexible. Instead of simply using the words directly to the right and left, the algorithm could allow certain types of words to be selected without strict regard to their position. For example, if the target word were a noun, the algorithm could select other nouns in a sentence as the context to use when disambiguating the target word.

A possible use for the measures of semantic relatedness and the word sense disambiguation algorithm is measuring the similarity of text documents. Considerable work has been done in measuring document similarity; however, methods for measuring document similarity often rely solely on literal, surface matching of text. That is, the methods only search for instances where the identical word is found in two different documents. These methods miss cases where similar words are used in different documents. For example, one document may contain the word *automobile* while another may contain *car*.

The word sense disambiguation algorithm could be applied to text documents to first disambiguate the words in the document, then the semantic relatedness measures could be used to find similar word senses that appear in different documents.

References

[1] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 805–810, Acapulco, Mexico, August 2003.

[2] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[3] Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, chapter 13, pages 305–332. MIT Press, 1998.

[4] Jay Jiang and David Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computational Linguistics (ROCLING X)*, Taiwan, 1997.

[5] Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, chapter 11, pages 265–283. MIT Press, 1998.

[6] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the Special Interest Group for Design of Communications Conference*, pages 24–26, Toronto, Ontario, 1986.

[7] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 296–304, Madison, Wisconsin, 1998.

[8] Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL '04), Main Volume*, pages 279–286, Barcelona, Spain, July 2004.

[9] George Miller, Claudia Leacock, Randee Tengi, and Ross Bunker. A semantic concordance. In *Proceedings of the Third DARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, New Jersey, 1993.

[10] Siddharth Patwardhan. Incorporating dictionary and corpus information into a context vector measure of semantic relatedness. Master's thesis, University of Minnesota Duluth, 2003.

[11] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, Mexico City, February 2003.

[12] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet::Similarity – measuring the relatedness of concepts. In *Proceedings of the Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 267–270, Boston, Massachusetts, 2004.

[13] Ted Pedersen, Banerjee Satanjeev, and Siddharth Patwardhan. Maximizing semantic relatedness to perform word sense disambiguation. Technical Report UMSI 2005/25, University of Minnesota Supercomputing Institute, March 2005.

[14] Philip Resnik. Semantic similarity in a taxonomy: An information-based measure and its applications to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130, 1999.

[15] Benjamin Snyder and Martha Palmer. The english all-words task. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[16] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, Las Cruces, New Mexico, 1994.