GRAPH-BASED CENTRALITY ALGORITHMS FOR UNSUPERVISED WORD SENSE DISAMBIGUATION

Ravi Som Sinha

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

December 2008

APPROVED:

Rada Mihalcea, Major Professor
Paul Tarau, Committee Member
Miguel Ruiz, Committee Member
Armin Mikler, Graduate Coordinator of the Department of Computer Science and Engineering
Krishna Kavi, Chair of the Department of Computer Science and Engineering
Costas Tsatsoulis, Dean of the College of Engineering
Sandra L. Terrell, Dean of the Robert B. Toulouse School of Graduate Studies

Sinha, Ravi Som. *Graph-based Centrality Algorithms for Unsupervised Word Sense Disambiguation*. Master of Science (Computer Science), December 2008, 48 pp., 9 tables, 4 figures, 42 references.

This thesis introduces an innovative methodology of combining some traditional dictionary based approaches to word sense disambiguation (semantic similarity measures and overlap of word glosses, both based on WordNet) with some graph-based centrality methods, namely the degree of the vertices, Pagerank, closeness, and betweenness. The approach is completely unsupervised, and is based on creating graphs for the words to be disambiguated. We experiment with several possible combinations of the semantic similarity measures as the first stage in our experiments. The next stage attempts to score individual vertices in the graphs previously created based on several graph connectivity measures. During the final stage, several voting schemes are applied on the results obtained from the different centrality algorithms.

The most important contributions of this work are not only that it is a novel approach and it works well, but also that it has great potential in overcoming the new-knowledge-acquisition bottleneck which has apparently brought research in supervised WSD as an explicit application to a plateau. The type of research reported in this thesis, which does not require manually annotated data, holds promise of a lot of new and interesting things, and our work is one of the first steps, despite being a small one, in this direction.

The complete system is built and tested on standard benchmarks, and is comparable with work done on graph-based word sense disambiguation as well as lexical chains. The evaluation indicates that the right combination of the above mentioned metrics can be used to develop an unsupervised disambiguation engine as powerful as the state-of-the-art in WSD.

ACKNOWLEDGEMENTS

First and foremost, my heartfelt gratitude goes toward my major professor and advisor, Dr. Rada Mihalcea, for her patience, her energetic, supportive and continuous guidance, and her valuable advice and feedback without which this work would not have been possible.

Next, I would like to thank the members of the committee at my defense, Dr. Paul Tarau and Dr. Miguel Ruiz, whose interest and feedback are deeply appreciated. The faculty and staff at the Department of Computer Science have also been very supportive and helpful throughout the process of this work.

Last but not least I would like to thank my parents and all my friends for their moral support and continuous encouragement.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1. A Word on Meaning

Languages are inherently ambiguous. Most words in a language usually have a multitude of meanings, and usually only one of its several possible meanings fits in a given context. We encounter this issue all around us - from lighthearted puns to misinterpreted statements to serious research, it is hard not to marvel at this phenomenon.

These meanings, or "senses" in natural language processing (NLP) terms, can further be classified as coarse-grained or fine-grained. As an example of the coarse-grained sense distinction, the word *"plant"* could refer to either the *green plant* sense of the word or the *factory* sense. As another example, the word *"bank"* could refer to any of the several senses of the word - *river bank, a place where we put our money, or the action of a plane while making a turn.* As far as coarse-grained senses are concerned, usually for humans as well as machines it is easy to be able to tell from context which sense is being referred to. All we need to do is read a sentence, and in extreme cases look at the preceding and succeeding sentences, and we know what the word really means.

Problems start arising when we look at the other way in which senses can be divided – namely fine-grained distinctions. An extreme case of finer distinctions in senses is that of looking at the word *"bank"*, and trying to determine whether it is meant as *the building where the bank is located*, *the financial institution in itself*, or *the bank as the company.* Sometimes it is difficult even for humans to distinguish such fine-grained words. However, for the most part, disambiguating the sense of a word given enough context like this is a very easy task for human beings, owing to common sense and other faculties of the mind that we are born

with. On the other hand, it is hard for a machine to be able to tell the finer-grained sense distinctions correctly.

The very notion of *meaning* is somewhat suspect from both philosophical as well as practical points of view, and several thoughts and alternative views have been put forward in this regard. While sometimes it is easier to divide words into different clusters of meanings, oftentimes one finds examples in which meanings are loose, overlapping or extended, as mentioned by Agirre and Edmonds in [1]. The complexity of word meanings is at its early stages of research and has not yet been satisfactorily explained. For the purpose of this thesis, however, word meanings would refer to a set of fixed senses as available in the WordNet sense repository.

To elaborate a little bit on the coarse-grained and fine-grained sense distinctions mentioned above, coarse-grained senses are very few - probably just 2 or 3 for a word - and they are totally unrelated in every way. They are often called *homographs* and they are considered almost a solved problem (refer to Chapter 2). Fine-grained senses are obtained by dividing up the coarse-grained senses further and breaking them down into a complex structure of interrelated senses (Agirre and Edmonds, [1].) Ide and Wilks, in the same book [1] have gone so far as to argue that both machines and humans are only able to reliably disambiguate coarse-grained senses. However, active research using WordNet as a repository has shown that machines are capable of disambiguating much finer levels of meanings to a reasonable degree.

1.2. Word Sense Disambiguation

Word sense disambiguation (WSD) is the task of resolving ambiguities in text resulting from the inherent polysemous nature of language. Starting in the 1940s as a sub-problem of the much wider field of artificial intelligence (AI), and related in particular to machine translation (MT), through the decades WSD has evolved in what it represents as well as in its potential applications. Researchers have developed several approaches that attempt to solve

this problem of *lexical disambiguation* (for a more complete history and the state-of-the-art in WSD, please refer to Chapter 2.)

One of the most important resources used in WSD research are sense inventories. Sense inventories provide a fixed set of meanings, thus offering a standard upon which all researchers can test their systems' performance. This introduces uniformity and scientific rigor, and provides a common ground on which different systems can be built and compared with one another. There are several sense inventories available today, as seen in the list below. Some of these have been listed with their pros and cons (please refer to Agirre and Edmonds [1]):

(i) Longman's Dictionary of Contemporary English: has subject codes, but is not available freely

(ii) WordNet: is freely available, but for many applications its senses are too fine-grained

(iii) Roget's Thesaurus

(iv) Hector: low coverage despite being detailed

(v) Oxford Advanced Learner's Dictionary

1.3. Applications of WSD

One of the most important reasons propelling this thesis work on WSD is the potential use that WSD can be put to under certain conditions. In general, WSD finds several applications in the following research areas within natural language processing, both as an explicit system as well as an implicit one (Agirre and Edmonds [1]):

(i) Machine Translation: Machine translation (MT) was the original problem that gave rise to WSD. Since different senses of a word often have completely different translations across languages, all machine translation systems implement some sort of WSD. Often the senses of a word in the source language are represented directly as the translated words in the other language. Even though most MT systems don't use an explicit WSD system because of the structure of the MT algorithms, still there is scope for a change in the algorithms so that WSD can be used explicitly.

A highly accurate WSD system is likely to enhance the performance of automatic translators to a great degree.

(ii) Information Retrieval: Information Retrieval (IR) often requires that ambiguities be resolved before certain queries can be performed. For example, consider the following example by Agirre and Edmonds from [1]: for the query item *depression*, the user querying the search engine might be looking for information related to *economics, illness* or *weather systems*. The same kind of issue needs to be addressed by a search engine for proper nouns - if there are two or more people with the same name the search engine doesn't know which results to show first. Current systems depend on the user to provide some context, e.g. *tropical depression* in order to provide the exact results. While this may seem relatively unimportant, recently WSD has been shown to improve performance in cross-lingual IR and document classification. Furthermore, subdivisions of IR like news recommendation, alerting, topic tracking and automatic advertisement placement can also benefit from a good WSD system.

(iii) Information Extraction and Text Mining: Any accurate analysis of a text requires WSD. Some application might be required to keep track of all information on the Internet regarding *tropical depression* rather than any of the other senses of *depression*. Correct analysis of text also finds a great deal of application in bioinformatics, where it might be required to collect genes and gene proteins separately, especially because genes and gene proteins often have the same names. The new field of the Semantic Web needs automatic annotation of documents. Acronym expansion and named-entity classification are other applications in need of a good WSD system.

(iv) Lexicography: WSD might be used to provide a rough higher-level grouping of senses which can help lexicographers provide better sense inventories and corpora back to the WSD system, thus creating a cycle where both components benefit from each other.

1.4. Utility of WSD

As explored in the above discussion, WSD can be used in various ways to benefit various existing applications. Even though systems resulting from research in WSD as an explicit problem have not shown any significant improvement in any real applications, the following points make it clear why there is still a lot of scope in WSD research:

(i) WSD as of today is not accurate enough for fine-grained sense distinctions. Improving this performance could change the way WSD bears down upon other applications.

(ii) No proper way has been found to integrate explicit WSD into existing systems.

(iii) Even though supervised corpus-based WSD suffers from the knowledge-acquisition bottleneck, and it is not feasible to collect sufficient manually tagged corpora for all possible domains, unsupervised WSD still has promise, and when explored adequately, it is expected to significantly improve the impact of WSD on all other research in NLP - because it can be be inexpensive and extremely scalable.

1.5. Problem Statement and Outline of Approach

Inspired by the above motivation, we attempt to develop a completely unsupervised word sense disambiguation system, making use of several available methodologies and combining them in an innovative way. The gist of this thesis work can be summarized as follows:

Given a sequence of sentences of words, containing content words (nouns, verbs, adjectives and adverbs) with several possible senses according to WordNet, the algorithm seeks to disambiguate all the content words in the sequence, by aiming to ultimately assign one sense to each content word. This is accomplished one word at a time, by building a graph around the word to be disambiguated.

By looking at the senses of neighboring words within different window-sizes (the size of the window ultimately used being an experimental issue,) the algorithm finds relationships amongst all vertices in such a graph, using combinations of different measures of semantic similarity based on the WordNet hierarchy. Which measure is used for which pair of vertices

is again an empirically determined parameter. This is followed by running several graph centrality algorithms on the (fairly dense) graph thus constructed. This step assigns scores to each vertex in the graph, thus facilitating selection of one, most appropriate, sense for each word.

This approach therefore tends to be an extension of some of the traditional *local* approaches of graph-based WSD (please refer to Chapter 2) - we look at the words and their neighbors, we determine how similar they are, and we use local measures of graph connectivity on top of that to assign final scores. Finally the algorithm also performs a voting amongst the results from the different graph algorithms.

1.6. Contribution of the Thesis

As apprehended in the SensEval-3 evaluation (refer to Chapter 2), supervised WSD as an explicit application has reached a point where we need to either significantly improve performance, or find applications for existing systems that perform moderately well. One of the reasons behind this could be that supervised WSD requires manually tagged data to train on, and collecting large amounts of such data for newer domains is not a feasible task. This problem is referred to as the *new knowledge acquisition bottleneck*.

The approach presented in this work is completely unsupervised - thus contributing toward overcoming the knowledge acquisition bottleneck and the scalability problems associated with typical supervised systems. Furthermore, the algorithm combines dictionary-based measures of semantic similarity, overlap of word glosses, and graph-centrality measures, which is an innovative combination of the two fundamentally different approaches for WSD (namely graph-based and dictionary-based) that have mostly only been studied separately so far. Graph-based approaches toward WSD are themselves a relatively new and unexplored territory. Unsupervised WSD in general is catching the attention of many leading researchers today, and when further exploited and extended with graph-based approaches, it promises a significant change in WSD as we know it today.

The results obtained by our system are competitive with the state-of-the-art, which speaks about the potential of these powerful approaches coming together in unison to address WSD.

1.7. Outline of the Remainder of the Thesis

The thesis is organized as follows: In Chapter 2, we take a good look at the history, basic approaches and the state-of-the-art in WSD using traditional approaches as well as graph based approaches. In Chapter 3, the graph-based algorithm for WSD is presented in detail, by going in details over how measures of semantic similarity and graph connectivity are used in it. This is followed by Chapter 4, i.e. the details about our experiments and evaluation. Finally these results are accounted for and a discussion is presented for future work in the final section, Chapter 5.

CHAPTER 2

LITERATURE SURVEY

2.1. Overview

This chapter attempts to take a look at some of the most recent research done in the field of word sense disambiguation (WSD), as well as the current state-of-the-art. WSD is the task of examining the word tokens in a given context and determining which of the several possible senses (according to a given sense inventory) of each word token is being used in that particular context. Given the sheer multitude of senses that a word could have, this task is far from trivial.

WSD is related to the more general task of lexical semantic processing, which comprises tasks focusing on meanings of individual words rather than meanings of a paragraph or a discourse. Even though the approaches to handle lexical semantics are considered to be a little different from those designed to handle compositional semantics, the problems of lexical and compositional ambiguity are closely related, because the ambiguity in the words of a sentence can lead to several possible interpretations of the sentence.

2.2. History of WSD

WSD is one of the oldest problems in computational linguistics. It was in the 1940s that WSD was first formulated as a separate task by Weaver [36]. It was acknowledged from the very beginning that the task is crucial and non-trivial. Zipf [42] then published his *Laws of Meaning*, [1] that accounts for the fact that the more frequent words have more senses compared to the less frequent words.

The inherent difficulty in the task of WSD was well appreciated further in the 1960s, and owing to the unfavorable ALPAC report in 1966, most of MT was abandoned. WSD

[1] different from his Zipf's Law on word frequencies

was then resurrected in the 1970s, within the research in artificial intelligence on complete natural language understanding. Wilks's preference semantics, as mentioned in [38], was one of the first systems to explicitly account for WSD.

The 1980s witnessed a turning point in WSD research, and large scale corpora and other lexical resources became available. Before the 1980s much of WSD research depended on handcrafting of rules. Now it became possible to use knowledge extracted automatically from the resources.

Lesk [17] came up with a simple yet seminal algorithm that used dictionary definitions from the Oxford Advanced Learner's Dictionary (OALD), and this marked the beginning of dictionary-based WSD. Guthrie et. al. [14] used the subject codes found in the Longman's Dictionary of Contemporary English (LDOCE) on top of the Lesk algorithm. Yarowsky [40] combined the information in Roget's thesaurus with co-occurrence data from large corpora in order to learn disambiguation rules for Roget's classes.

The 1990s saw further improvements in the field, which can be categorized in three major groups - WordNet, statistical NLP, and SensEval (later SemEval).

WordNet made it possible for all the researchers to have easy and free access to a standardized inventory using which to compare their work. Its hierarchical structure, synsets, and other such features made it the most used general sense inventory in WSD research.

The mainstream approach in WSD so far has been supervised learning, where systems are trained on manually tagged corpora. Statistical approaches in supervised learning were used by Weiss [37] and several others which was a foresight to the so-called "statistical revolution" in the 1990s. Brown et. al. [8] were the first to use corpus-based WSD in statistical MT.

SensEval (later SemEval) made it possible for researchers to compare different systems with each other because of the fixed set of test words, annotators, sense inventories, and corpora. Before SensEval, the only common ground that WSD researchers had were a lower bound (calculated by either picking a random sense, or taking into account the most frequent senses) and an upper bound (derived from inter-tagger agreement). Now it became

possible to develop different systems and evaluate them on the data sets provided by SensEval, thereby introducing scientific rigor and uniformity. SensEval eventually became the primary forum for all WSD evaluations.

In SensEval-3 (2004), a conclusion was reached that WSD in itself has reached a performance plateau, and no significant leap in the results obtained already is possible. It is since then, that people started thinking about new directions in which WSD research can go. In particular, in recent years there has been considerable growth in the areas of parallel bilingual corpora, and unsupervised corpus-based WSD. This thesis combines unsupervised WSD with dictionary-based methods, and attempts to draw upon the idea that unsupervised WSD is the way to go in future.

2.3. Basic Approaches to WSD

There are primarily two traditional approaches used to solve the problem of WSD:

(i) Dictionary-based/ knowledge-based approaches: Dictionary-based approaches rely on dictionaries, thesauri, other lexical knowledge bases and ontologies. While several methods are used under these approaches - namely selectional restrictions, overlap of definition text, and semantic similarity measures, dictionary-based methods still remain a matter of research.

(ii) Corpus-based approaches: These approaches can further be divided into two categories - supervised and unsupervised.

2.3.1. *Supervised approaches*

Supervised approaches have been the mainstream technique used in WSD for a long time and have produced considerable results. These methods make use of annotated data to train from, or the starting data in a bootstrapping process. The algorithms used under supervised methods include aggregative and discriminative algorithms, using feature selection, parameter optimization and ensemble learning.

While they perform really well, these methods suffer from a problem known as the knowledge-acquisition bottleneck, meaning it is not always feasible to have at one's disposable large corpora annotated manually for every possible domain.

2.3.2. *Unsupervised approaches*

These approaches avoid all external information, working directly with the information available in form of raw unannotated corpora. Potentially, these approaches can overcome the problems faced by supervised approaches as they do not depend on manually annotated corpora. Theoretically, all types of information can be learned by raw corpora, which makes these approaches extremely scalable too. The work done by Schütze [32] follows this direction.

Unsupervised approaches can also induce word senses from training text by clustering word occurrences, and then classifying new occurrences into one of the existing clusters/ sense groupings (Agirre and Edmonds [1]).

Modern systems in WSD research are mostly combinations of the above methods. All systems usually extract features of a target word based on the context it is found in, and compare the features with the other information collected for that word. WSD is thus basically a classification problem, notable for its very *high-dimensional feature space*. The feature space consists of properties such as *part of speech, word form - written or lemma, subject or domain code, semantic class* etc.

2.4. Graph-based Approaches to WSD

One direction that modern research in WSD has taken is graph-based approaches. Work has been done in this area by Barzilay and Elhadad [3], Navigli and Velardi [24], and Mihalcea [22]. Comparative studies performed by Mihalcea [22] indicate that graph-based methods often outperform the similarity-based ones by a significant margin.

One of the earliest works in graph-based WSD, namely that by Barzilay and Elhadad [3], focuses on using lexical chains based on lexical cohesion for producing indicative summaries of text. Navigli and Lapata [26] state that most unsupervised algorithms can be seen as

either similarity-based approaches or graph-based approaches. The authors further note that these graph-based algorithms often have two stages - during the first stage a graph is constructed based on all the possible sequences of senses for the words to be disambiguated. During the next stage the structure of the graph is exploited to determine the most important nodes in the graph, which eventually leads to disambiguation of the polysemous words in the context. As mentioned in the above referenced paper, graph-based approaches attempt to assign senses to words collectively in a global manner, by exploiting dependencies across senses, while in contrast similarity-based approaches disambiguate each word individually without looking at the senses that are assigned to the words immediately before and after.

Also, as noted in Navigli and Lapata [26], graph connectivity measures could be *local* or *global*. If the graph connectivity measure is local, then for each word to be disambiguated, the measure is able to select one sense which is most suitable in the given context. If the connectivity measure happens to be global, then the disambiguation methodology changes a little bit, in that instead of providing the disambiguated senses for each polysemous word, the algorithms score the overall interpretation of the sentence. Thus, if a sentence could have twenty possible interpretations, then a local graph connectivity measure provides us with the best scoring sense for each polysemous word in one iteration, whereas a global graph connectivity measure provides us with twenty overall possible assignments of senses, each assignment with a score. Some of the local centrality measures are *indegree*, *PageRank* (Brin and Page [7]), *closeness* and *betweenness* (Freeman [12]). Some well-known global centrality measures are *compactness* (Botafago et. al. [5]), *graph entropy* and *edge density*.

Apart from providing a comprehensive comparison amongst different measures of graph centralities, Navigli and Lapata [26] present a generic graph-based algorithm which handles one sentence at a time and disambiguates polysemous words based entirely on the structure of the graph of their sense-inventory (WordNet or Extended WordNet (Navigli [25].) For their purposes, they view the WordNet graph as a set of nodes (comprising *synsets*) and edges (comprising the relationship between the synsets, e.g. *synonyms*, *meronyms* etc.) For

a sentence to be disambiguated, they start with a graph $G = (V, E)$ where V consists of each synset in WordNet corresponding to all the words in the sentence. Next, they perform a **depth-first search (DFS)** of the WordNet graph. In order to do so, they start with any vertex u in V and keep searching the WordNet graph until they find another vertex v in it which was already present in V. Every time this happens, they add all the intermediate nodes on this path to G as undirected edges.

Navigli and Lapata use random assignment as lower bound and a first-sense heuristic based on SemCor as an upper bound, and use simulated annealing to speed up the algorithm. They report the performance of their system on SemCor (Miller et. al. [23]) and SensEval-3 English all-words data. For WordNet, the results obtained by them when using some local graph connectivity measures are depicted in Table 2.1. For a more detailed report including the results from the Extended WordNet and some global measures, please refer to Navigli and Lapata [25]. Table 2.2 depicts the results of the best performing measure, KPP (Key Player Problem, a variant of Closeness, Borgatti [4]) on SensEval-3 English all-words data.

	WordNet		
Measure	Prec	Rec	F1
Baseline	23.7	23.7	23.7
Indegree	35.3	24.0	28.6
Betweenness	38.4	15.5	22.1
Closeness variant (KPP)	31.8	31.8	31.8
PageRank	35.3	24.0	28.6

Table 2.1. Performance of local graph connectivity measures on SemCor data

The conclusions drawn from the above presented work by Navigli and Lapata are that (1) local measures perform better than global measures, and (2) KPP performs better than in-degree, betweenness and other centrality measures.

Measure	Part of speech	Prec	Rec	F1
KPP	Nouns	61.9	61.9	61.9
	Adjectives	62.8	62.8	62.8
	Verbs	36.1	36.1	36.1

TABLE 2.2. Performance of KPP on SENSEVAL-3 English all-words data

Our work, though related to the experiments reported above by Navigli and Lapata, is considerably different. In their work, the graphs are built directly from WordNet, and thus include links explicitly encoded in the structure of WordNet, rather than accounting for semantic similarities, as we do. Given a sentence and the list of senses for all the words in the sentence, for each sense they traverse the WordNet graph using a depth-first search strategy, and if a new node is found in the WordNet graph that also exists in the list of the word senses for the current sentence, all the intermediate edges and nodes from WordNet are added to the graph. Since the edges in the WordNet graph are semantic relations and not numerical quantities, the graph built in their method is unweighted.

In contrast, our approach, although formulated in a similar graph-based setting, does not disambiguate on a sentence-by-sentence basis, but rather on the basis of the target word and a number of words before and after the target word. We thus construct separate graphs for each word to be disambiguated. Our approach yields almost identical results for nouns, and considerably better results for verbs, as measured on the SENSEVAL-3 data, which was used in their experiments. They obtain a precision and recall of 61.90, 36.10 and 62.80 for nouns, verbs and adjectives respectively, compared to a precision and recall of of 61.93, 46.24 and 53.63 for the same parts of speech, as obtained by us (see Table 4.7).

Another recent work in graph-based algorithms for WSD (Mihalcea [22]), presents an algorithm for automatic sequence data labeling, tested on the problem of unsupervised WSD targeting all open-class words in unrestricted text. The algorithm uses a global annotation approach, annotating all the words in a sequence simultaneously by exploiting dependencies

across labels and random walks. The work presented in this thesis builds on the algorithm proposed in this paper, therefore please refer to Chapter 3 for a detailed description of the algorithm. The author of the above mentioned paper reports results using the presented graph model and a definition-based similarity measure, obtained on SensEval-2. Her work also compares results with an enhanced version of the **Lesk** algorithm, presented by Cowie et. al. in [10]. Since both algorithms (Mihalcea and Cowie) work with dictionary overlaps, it is reasonable to directly compare results from the two algorithms. The author reports accuracies of 54.2% for fine-grained sense distinctions and 55.3% for coarse-grained sense distinctions, both times achieving an error rate reduction of about 11% over the improved lesk algorithm. The author further reports a fine-grained accuracy of 55.2% on SensEval-3 and 56.5% on a subset of SemCor, both significantly better than accuracies achieved using individual data labeling (lesk) or random assignment.

Our approach builds on a method similar to the one reported above, although instead of a measure of similarity based on sense definitions computable on any machine readable dictionary, we experiment with various semantic similarity measures. While the approach with the sense definition overlap gives an accuracy of 54.2% on SensEval-2 data, the approach presented in this thesis gives an overall score of 58.39%, which represents a significant improvement. The SensEval-3 results reported in [22] consisted of a precision and recall of 52.20%, which are again improved over by our current system that provides an overall score of 55.05%.

Some other notable algorithms which are similar to the approach used by Mihalcea [22] are Hidden Markov Models, especially the well-known **Viterbi** algorithm. However, whereas the Viterbi algorithm attempts to maximize the overall score of a sentence (global connectivity measure), our algorithm, albeit drawing upon a global graph, assigns senses locally, individually to each word.

Finally, our work is also comparable to a method for word sense disambiguation based on lexical chains, proposed by Galley and McKeown in [13]. In that method, lexical chains

were constructed over a text by using the semantic relations from WordNet, which were empirically assigned with a weight (e.g., a synonymy relation identified between word senses in the same sentence was assigned with a weight of 1, whereas a sibling relation found across three sentences had a weight of 0.3). Once the lexical chains were constructed, a sense was selected for each word based on the strength of the connectivity to other words in the chain. The algorithm was evaluated on the disambiguation of all the nouns from 74 documents from SemCor, which led to an overall score of 62.09%. An evaluation of our system on the nouns from the same data set led to a significantly higher disambiguation score of 68.70% (see Table 4.7).

2.5. State-of-the-art in Non-graph-based WSD

The disambiguation of homographs is considered a solved problem, and various systems have repeatedly reported results with accuracies nearing 95%. In 1995, Yarowsky [41] reported an accuracy of 96.5% on a related task, while in 2001 Stevenson and Wilks [35] reported an accuracy of 94.7%.

The disambiguation of polysemy is harder, but it has been improving over the years. Table 2.3 shows the progressive results reported by the systems participating in SensEval over the years. The apparent drop in the best results from SensEval-1 to SensEval-2 may be attributed toward the fact that SensEval-2 was a harder task in itself as it was based on finer-grained senses from WordNet as compared to Hector for SensEval-1. Refer to Agirre and Edmonds [1] for details.

At SensEval-2, a fully unsupervised system (one not using a back-off method to most-frequent sense) developed by Litkowski [19] combined analysis of multiword units and contextual clues based on collocations and content words from dictionary definitions and examples, and had an overall accuracy of 45.1%. McCarthy et. al.[21] report one of the best evaluations on SensEval-2, using automatic derivation of the most frequent sense of a word using distributional similarities learned from a large raw corpus, for a precision of 53.0% and a recall of 49.0%.

In SemEval-2007, one of the tasks (please refer to Agirre et. al. [2]) was related to cross-lingual IR, and was proclaimed to be one of the first attempts toward an application-driven WSD exercise. In this task, participants tried to disambiguate text by assigning WordNet 1.6 synsets to it, and then expanding the text to other languages on which an IR exercise was to be performed. The retrieval results indicated how good the WSD system was. On the SensEval-2 all-words data, the best performing system reached a precision of 58.4% and a recall of 57.7% respectively. On the SensEval-3 all words data, the best performing score was a precision of 59.1% and a recall of 56.6% respectively.

A lot of WSD research so far has focused on content words - namely nouns, verbs, adjectives and adverbs. Recently at SemEval-2007 one of the tasks was to disambiguate prepositions (please refer to Litkowski [20].) It was designed as a lexical sample task, and the best-performing systems reached a recall of 100% and precisions of 69.3% and 75.5% respectively for fine-grained and coarse-grained sense distinctions. This task has generated considerable information for further research on preposition WSD and this work can further be used in several NLP tasks.

A coarse-grained WSD task (refer to Navigli et. al. [27]) was also administered at SemEval2007, and an overall F-1 measure score of 82.50% was achieved. The system used a back-off strategy, depending on the most-frequent-sense heuristic whenever no sense assignment was attempted. In contrast to previous attempts at coarse-grained WSD, this exercise was performed as an all-words task on a coarse-grained version of WordNet. Since recent consensus amongst WSD researchers has been that some sense inventories are so fine-grained that even inter-tagger agreement is not encouraging, this research sheds light on a relatively new direction that WSD research can take.

Last but not least, SemEval-2007 also had a WSD task comprising an English fine-grained all-words task, evaluated on data from the Wall-Street Journal (WSJ) and annotated with senses of WordNet 2.1, and an English coarse-grained lexical sample task, on a selected set of lexemes (refer to Pradhan et. al. [30].) The systems reached an overall F-score of

59.1% for the all-words task and an F-score of 88.7% on the lexical sample task. One of the conclusions drawn from this task were that if the human annotator agreement is high, we can hope to build a WSD system which might benefit any of the other NLP tasks.

In the light of all this work done in both graph-based as well as non-graph based WSD, our system with a SensEval-2 English all-words F-1 score of 58.39%, SensEval-3 English all-words F-1 score of 55.05% and an F-1 score of 63.79% on a subset of the SemCor corpus performs on par with some of the best results reported so far on these data sets, with the added advantage that our approach is completely unsupervised.

Year	Senseval	Task	Language	System	Result
1997	SensEval-1 (Hector)	LS	English	Inter-tagger agreement	80%
				Best	77%
2001	SensEval-2 (WordNet)	LS	English	Inter-tagger agreement	86%
				MFS baseline supervised	48%
				Best supervised	64%
				MFS baseline unsupervised	16%
				Best unsupervised	40%
		AW		MFS baseline supervised	57%
				Best supervised	69%
				Best unsupervised	55%
2004	SensEval-3 (WordNet)	LS	English	Inter-tagger agreement	67%
				MFS baseline supervised	55%
				Best supervised	73%
				Best unsupervised	66%
		AW		MFS baseline supervised	62%
				Best supervised	65%
				Best unsupervised	58%
2007	SemEval-2007	IR	Cross-lingual	SensEval-2 all-words data	58%
				SensEval-3 all-words data	59%
		LS	English prep.	coarse-grained	76%
				fine-grained	69%
		AW	English	coarse-grained	83%
		AW	English	fine-grained	59%
		LS	English	coarse-grained	89%

Table 2.3. Advances in WSD over the years, LS = Lexical Sample, AW = All Words, MFS = Most Frequent Sense, prep. = prepositions

CHAPTER 3

A GRAPH-BASED ALGORITHM FOR WORD SENSE DISAMBIGUATION

In this chapter,[1] we describe the graph representation used to model word sense dependencies in text, and show how graph centrality algorithms can be used to determine the most likely combination of word senses. This is an extension of the random-walk sequence data labeling algorithm proposed by Mihalcea in [22].

3.1. Graph Representation

Given a sequence of words $W = \{w_1, w_2, ..., w_n\}$, each word w_i with corresponding possible labels $L_{w_i} = \{l^1_{w_i}, l^2_{w_i}, ..., l^{N_{w_i}}_{w_i}\}$, we define a label graph G = (V,E) such that there is a vertex $v \in V$ for every possible label $l^j_{w_i}$, $i = 1..n$, $j = 1..N_{w_i}$. Dependencies between pairs of labels are represented as undirected edges $e \in E$, defined over the set of vertex pairs $V \times V$. Such label dependencies can be learned from annotated data, or derived by other means. In our case, these dependencies are learned from WordNet and measures of semantic similarity. Figure 3.1 shows an example of a graphical structure derived over the set of labels for a sequence of four words. Note that the graph does not have to be fully connected, as not all label pairs can be related by any dependency function. The figure 3.1 depicts the situation after the dependencies have been resolved, but before the graph centrality algorithms are run to determine the scores of the vertices.

Given such a label graph associated with a sequence of words, the likelihood of each label can be determined using a graph-based centrality algorithm, which runs over the graph of labels just created, and identifies the importance of each label (vertex) in the graph. The

[1] Parts of this chapter have been previously published in a work by the same author (Sinha and Mihalcea, [33])

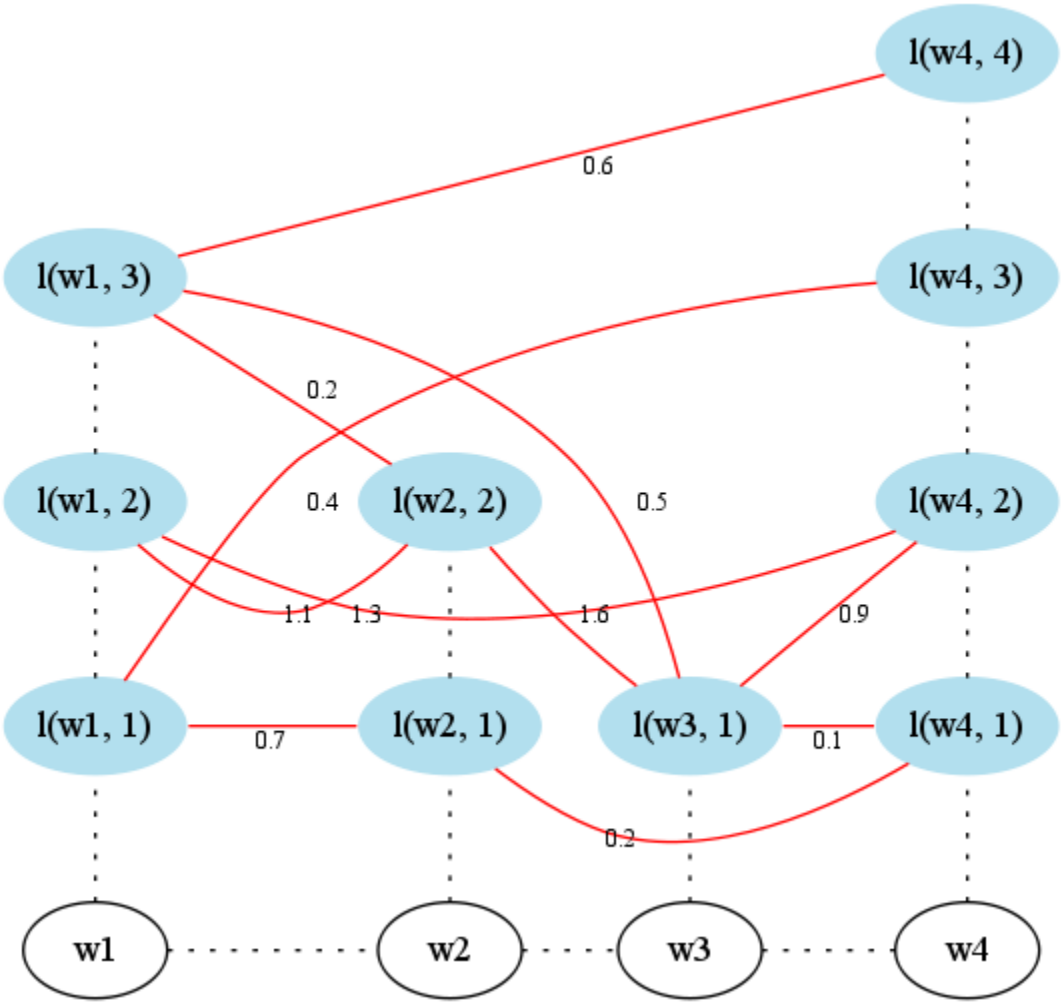

FIGURE 3.1. Sample graph built on the set of possible labels (shaded nodes) for a sequence of four words (unshaded nodes). Label dependencies are indicated as edge weights.

graph-based algorithm results in a set of scores attached to vertices in the graph, which are used to identify the most probable label (sense) for each word.

For instance, for the graph drawn in Figure 3.2, the word w_1 will be assigned with label $l^1_{w_1}$, since the score associated with this label (1.39) is the maximum among the scores assigned to all admissible labels associated with this word.

A property that makes these graph-based algorithms interesting is the fact that they take into account information drawn from the entire graph, capturing relationships among all the

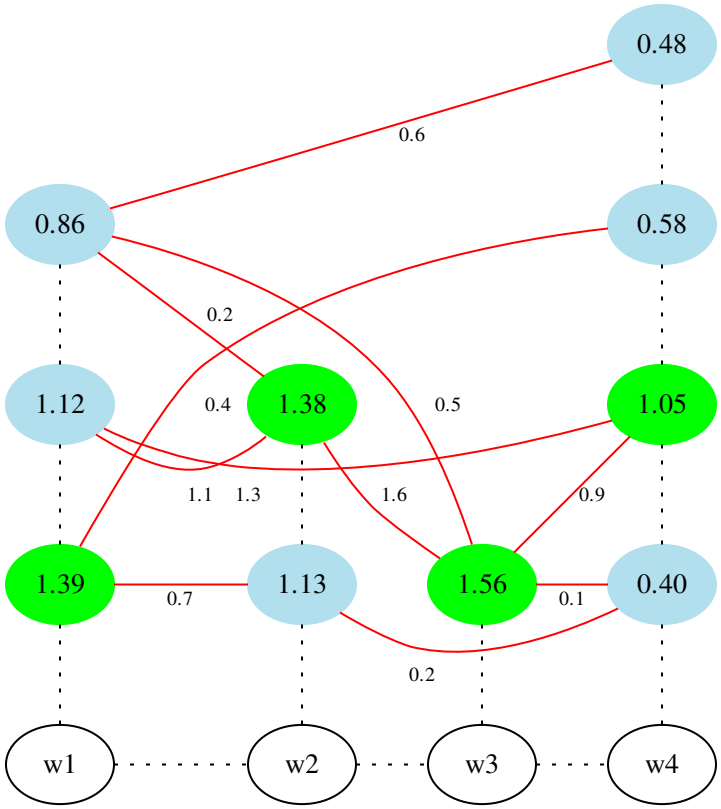

FIGURE 3.2. Graph centrality measures are run on the graph previously created. The nodes in dark gray depict the chosen nodes, i.e. the nodes with the highest centrality values for each word to be labeled. The centrality measure could be chosen out of a large number of options.

words in a sequence, and following this *global* technique they fall back to a *local* measure of graph centrality to assign labels. As will be seen ahead, this combination works very well.

3.2. Word Sense Disambiguation Algorithm

Given a sequence of words with their corresponding admissible labels (senses), the disambiguation algorithm seeks to identify a graph of sense dependencies on which the centrality can be measured, resulting in a set of scores that can be used for sense assignment. Algorithm 1 shows the pseudocode for the labeling process. The algorithm consists of three main

Algorithm 1 Graph Centrality for Word Sense Disambiguation

Input: Sequence $W = \{w_i | i = 1..N\}$

Input: Admissible senses $L_{w_i} = \{l^t_{w_i} | t = 1..N_{w_i}\}$,$i = 1..N$

Output: Sequence of senses $L = \{l_{w_i} | i = 1..N\}$, with sense l_{w_i} corresponding to word w_i from the input sequence.

Build graph G of sense dependencies

for $i = 1$ to N **do**
  for $j = i + 1$ to N **do**
    if $j - i > MaxDist$ **then**
      break
    end if
    for $t = 1$ to N_{w_i} **do**
      for $s = 1$ to N_{w_j} **do**
        $weight \leftarrow Dependency(l^t_{w_i}, l^s_{w_j}, w_i, w_j)$
        if $weight > 0$ **then**
          $AddEdge(G, l^t_{w_i}, l^s_{w_j}, weight)$
        end if
      end for
    end for
  end for
end for

Score vertices in G

for all $V_a \in Vertices(G)$ **do**
  $Score(V_a) \leftarrow Centrality(V_a)$
end for

Sense assignment

for $i = 1$ to N **do**
  $l_{w_i} \leftarrow argmax\{WP(l^t_{w_i}) | t = 1..N_{w_i}\}$
end for

steps: (1) construction of sense dependencies graph; (2) sense scoring using graph-based centrality algorithms; (3) sense assignment.

First, a weighted graph of sense dependencies is built by adding a vertex for each admissible sense, and an edge for each pair of senses for which a dependency is identified. A maximum allowable distance can be set ($MaxDist$), indicating a constraint over the distance between words for which a sense dependency is sought. For instance, if $MaxDist$ is set to 3, no edges will be drawn between senses corresponding to words that are more than three words apart, counting all running words. Alternatively, a window of, say 3 words could be chosen so that in order to disambiguate a word the algorithm looks at three words preceding the word and three words following the word. Sense dependencies are determined through the *Dependency* function, which encodes the relation between word senses. We experiment with six different measures of word semantic similarity as a means to derive the dependency between word senses (see Section 3.6).

Next, scores are assigned to vertices using a graph-based centrality algorithm. In this work, we experiment with four centrality algorithms, namely: indegree, closeness, betweenness, and PageRank (see Section 3.7).

Finally, the most likely set of senses is determined by identifying for each word the sense that has the highest score. Note that all admissible senses corresponding to the words in the input sequence are assigned with a score, and thus the selection of two or more most likely senses for a word is also possible. Right now, our algorithm chooses the first sense which has a centrality score greater than the previous senses. If after that another sense is found to have the same centrality score, the previous sense remains the predicted sense.

3.3. Word Sense Dependencies

Word sense dependencies can be defined in various ways, depending on the knowledge sources that are available. If an annotated corpus is available, dependencies can be defined as label co-occurrence probabilities approximated with frequency counts $P(l^t_{w_i}, l^s_{w_j})$, or as conditional probabilities $P(l^t_{w_i}|l^s_{w_j})$. Optionally, these dependencies can be lexicalized by taking into account the corresponding words in the sequence, e.g. $P(l^t_{w_i}|l^s_{w_j}) \times P(w_i|l^t_{w_i})$.

In the absence of an annotated corpus (as is the case in our experiments), dependencies can be derived based on the information available in dictionaries or semantic networks, by measuring the semantic similarity between word senses. In this work, we experiment with a variety of such similarity measures, which are described in Section 3.6. [2]

Once calculated, the dependencies between word senses are set as weights on the arcs drawn between the corresponding senses. Arcs can be directed or undirected for joint probabilities or similarity measures, and are usually directed for conditional probabilities. In our experiments, the arcs are weighted and undirected.

3.4. Labeling Example

Consider again the example from Figure 3.1, consisting of a sequence of four words, and their possible corresponding senses. In the first step of the algorithm, sense dependencies are determined, and let us assume that the values for these dependencies are as indicated through the edge weights in Figure 3.1. Next, vertices in the graph are scored using a graph centrality algorithm, resulting in a score attached to each sense, shown in place of the name of the vertex in Figure 3.2. Finally, the most probable sense for each word is selected.

Word w_1 is thus assigned with sense $l^1_{w_1}$, since the score of this sense (1.39) is the maximum among the scores associated with all its possible senses (1.39, 1.12, 0.86). Similarly, word w_2 is assigned with sense $l^2_{w_2}$, w_3 with sense $l^1_{w_3}$, and w_4 receives sense $l^2_{w_4}$.

3.5. Efficiency Considerations

For a sequence of words $W = \{w_1, w_2, ..., w_n\}$, each word w_i with N_{w_i} possible senses, the running time of the graph-based sequence data labeling algorithm is proportional to $O(C \sum_{i=1}^{n} \sum_{j=i+1}^{i+MaxDist} (N_{w_i} \times N_{w_j}))$ (the time spent in building the sense graph, and possibly

[2] Please note that, even though SemCor and SensEval corpora have manual tags, our approach does not look at the tags for the development or testing phase. The tags are only used to check the accuracy of the system after the system has used the proposed algorithm and has thereby predicted senses. In fact, we have even extended our work so that the user has the choice of either checking the performance of the system by working on a corpus which is manually tagged, or actually tagging an untagged corpus with senses.

iterating the algorithm for a constant number of times C). This is orders of magnitude better than the running time of $O(\prod_{i=1}^{n} N_{w_i})$ for algorithms that attempt to select the best sequence of senses by searching through the entire space of possible sense combinations, although it can be significantly higher than the running time of $O(\sum_{i=1}^{n} N_{w_i})$ for individual sense labeling, where no dependency graphs are constructed over the possible word senses in the input sequence.

3.6. Measures of Word Semantic Similarity

Having described the mechanics of the algorithm above, we now describe the means used to determine the dependency function amongst the words. There are a number of measures that were developed to quantify the degree to which two words are semantically related using information drawn from semantic networks – see e.g. work by Budanitsky [9] for an overview. We present below several measures found to work well on the WordNet hierarchy. All these measures assume as input a pair of concepts, and return a value indicating their semantic relatedness. The six measures below were selected based on their observed performance in other language processing applications, and for their relatively high computational efficiency.

We conduct our evaluation using the following word similarity metrics: Leacock & Chodorow, Lesk, Wu & Palmer, Resnik, Lin, and Jiang & Conrath. We use the WordNet-based implementation of these metrics, as available in the WordNet::Similarity package (please refer to Patwardhan [29]). We provide below short descriptions for each of these six metrics.

The **Leacock & Chodorow(lch)** [16] similarity is determined as:

$$Sim_{lch} = -\log \frac{length}{2 * D} \tag{1}$$

where $length$ is the length of the shortest path between two concepts using node-counting, and D is the maximum depth of the taxonomy.

The **Lesk** similarity of two concepts is defined as a function of the overlap between the corresponding definitions, as provided by a dictionary. It is based on an algorithm proposed

by Lesk [17] as a solution for word sense disambiguation. The application of the Lesk similarity measure is not limited to semantic networks, and it can be used in conjunction with any dictionary that provides word definitions.

The **Wu and Palmer(wup)** [39] similarity metric measures the depth of two given concepts in the WordNet taxonomy, and the depth of the least common subsumer (LCS), and combines these figures into a similarity score:

$$Sim_{wup} = \frac{2 * depth(LCS)}{depth(concept_1) + depth(concept_2)} \tag{2}$$

The measure introduced by **Resnik(res)** [31] returns the information content (IC) of the LCS of two concepts:

$$Sim_{res} = IC(LCS) \tag{3}$$

where IC is defined as:

$$IC(c) = -\log P(c) \tag{4}$$

and $P(c)$ is the probability of encountering an instance of concept c in a large corpus.

The next measure we use in our experiments is the metric introduced by **Lin** [18], which builds on Resnik's measure of similarity, and adds a normalization factor consisting of the information content of the two input concepts:

$$Sim_{lin} = \frac{2 * IC(LCS)}{IC(concept_1) + IC(concept_2)} \tag{5}$$

Finally, the last similarity metric considered is **Jiang & Conrath(jcn)** [15]:

$$Sim_{jcn} = \frac{1}{IC(concept_1) + IC(concept_2) - 2 * IC(LCS)} \tag{6}$$

3.7. Graph-based Centrality Algorithms

Now that we know how to determine the *Dependency* function in our graph, and therefore to get all the vertices connected with edges having weights as determined by the *Dependency* function (any combination of the semantic similarity metrics described above), we describe a way to assign scores to the individual nodes, using measures of graph connectivity or centrality.

The basic idea implemented by a graph centrality algorithm is that the "importance" of a node in a graph can be determined by taking into account the relation of the node with other nodes in the graph. In our experiments, we use four centrality algorithms: indegree, closeness, betweenness, and PageRank.

The **indegree** of a vertex refers to the number of edges incident on that vertex. For an undirected graph, as used in our experiments, the "indegree" is equivalent to the degree of the vertex; thus, an edge contributes towards the degrees of the vertices at both its ends. Since our graphs are weighted, we calculate the indegree by taking into account the weights on the edges, and adding them together into a score that reflects the centrality of the vertex. Thus, for an undirected weighted graph $G = (V, E)$, the indegree is defined as follows:

$$Indegree(V_a) = \sum_{(V_a, V_b) \in E} w_{ab} \tag{7}$$

where w_{ab} is the weight on the edge between V_a and V_b.

The indegree is usually normalized by dividing the value by the maximum degree in the graph (Navigli [26]). Here, we adopt a different strategy, where the weights on the edges are themselves normalized according to their ranges (see Section 4.3 for details).

The **closeness** of a vertex can be defined in multiple ways. In our experiments, we define the closeness of a vertex as the reciprocal of the sum of the shortest paths between the vertex and all the other vertices in the graph:

$$Closeness(V_a) = \frac{1}{\sum_{V_b \in V} s(V_a, V_b)} \tag{8}$$

where $s(V_a, V_b)$ is used to denote the "shortest path" or "shortest geodesic distance" between the nodes V_a and V_b. Here the nodes represent the words. The shortest geodesic distance can be computed using the Dijkstra's algorithm. The description of closeness can be found in Freeman [12]. In the weighted graphs built in our experiments, we use a weighted version of the closeness measure, which takes into account the weights on the edges while computing the shortest path.

The **betweenness** of a node is defined in terms of how "in-between" a vertex is among the other vertices in the graph (Freeman [11]). Formally:

$$Betweenness(V_a) = \sum_{V_b \in V, V_c \in V} \frac{\sigma_{V_b,V_c}(V_a)}{\sigma_{V_b,V_c}} \tag{9}$$

where σ_{V_b,V_c} represents the total number of shortest geodesic paths between V_b and V_c, while $\sigma_{V_b,V_c}(V_a)$ means the number of such paths that pass through V_a.

Closeness and betweenness are usually regarded as extremely computationally expensive methods owing to the number of shortest paths that need to be calculated. For betweenness, we use a simplified algorithm found to approximate well the original definition of betweenness, while being significantly more efficient (refer to Brandes [6]).

Finally, the last graph centrality algorithm we consider is **PageRank** (Brin and Page [7]). The main idea implemented by PageRank is that of "voting" or "recommendation." When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting a vote determines how important the vote itself is, and this information is also taken into account by the ranking algorithm. Although PageRank was originally defined on directed graphs, it can also be applied on undirected graphs. The PageRank score associated with a vertex V_a is defined using a recursive function:

$$PageRank(V_a) = (1-d) + d * \sum_{(V_a,V_b)\in E} \frac{PageRank(V_b)}{|degree(V_b)|} \tag{10}$$

where d is a parameter that is set between 0 and 1. The typical value for d is 0.85 (Brin and Page [7]), and this is the value we are using in our implementation.

This vertex scoring scheme is based on a random-walk model, where a walker takes random steps on the graph G, with the walk being modeled as a Markov process – that is, the decision on what edge to follow is solely based on the vertex where the walker is currently located. Under certain conditions, this model converges to a stationary distribution of probabilities, associated with vertices in the graph.

In a weighted graph, the decision on what edge to follow during a random walk is also taking into account the weights of outgoing edges, with a higher likelihood of following an edge that has a larger weight. Given a set of weights w_{ab} associated with edges connecting vertices V_a and V_b, the weighted PageRank score is determined as:

$$PageRank(V_a) = (1-d) + d \sum_{(V_a,V_b)\in E} \frac{w_{ba}}{\sum_{(V_c,V_b)\in E} w_{bc}} PageRank(V_b) \tag{11}$$

3.8. An Example

Consider the task of assigning senses to the words in the text *The church bells no longer rung on Sundays.*[3] For the purpose of illustration, we assume at most three senses for each word, which are shown in Figure 3.3. Word senses and definitions are obtained from the WordNet sense inventory. All word senses are added as vertices in the label graph, and weighted edges are drawn as dependencies among word senses, derived using the Lesk similarity measure (no edges are drawn between word senses with a similarity of zero). The resulting label graph is an undirected weighted graph, as shown in Figure 3.3. After running the PageRank graph centrality algorithm, scores are identified for each word-sense in the

[3]Example drawn from the data set provided during the SensEval-2 English all-words task.

graph, indicated in Figure 3.4. Selecting for each word the sense with the largest score results in the following sense assignment: *church#2*, *bell#1*, *ring#3*, *Sunday#1*, which is correct according to annotations performed by professional lexicographers.

The **church bells** no longer **rung** on **Sundays**.

church

1: one of the groups of Christians who have their own beliefs and forms of worship

2: a place for public (especially Christian) worship

3: a service conducted in a church

bell

1: a hollow device made of metal that makes a ringing sound when struck

2: a push button at an outer door that gives a ringing or buzzing signal when pushed

3: the sound of a bell

ring

1: make a ringing sound

2: ring or echo with sound

3: make (bells) ring, often for the purposes of musical edification

Sunday

1: first day of the week; observed as a day of rest and worship by most Christians

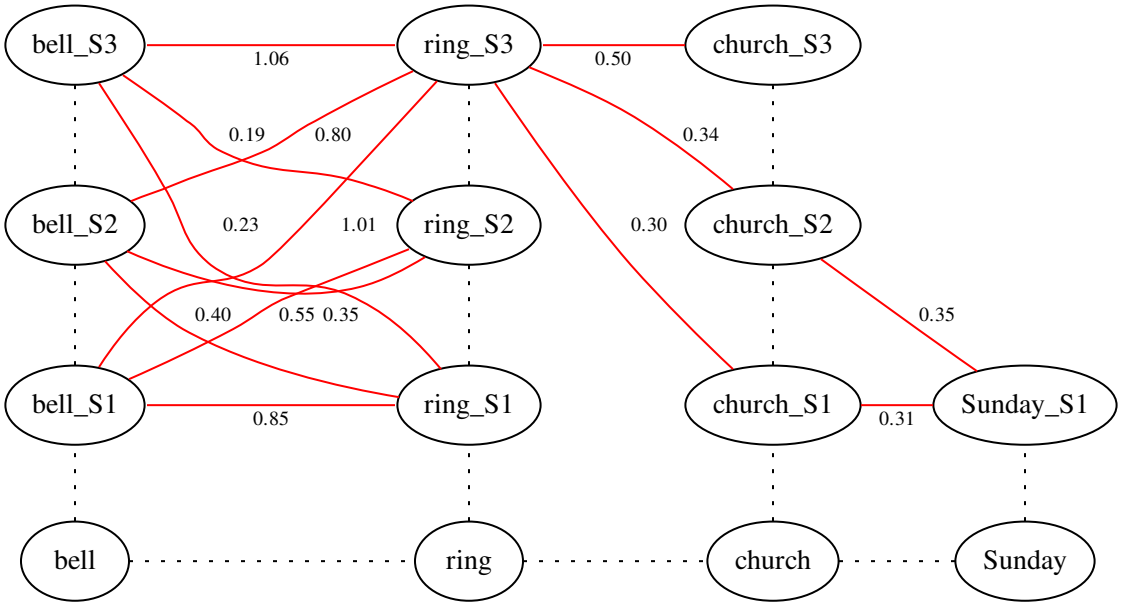

FIGURE 3.3. The graph for assigning senses to the words in *The church bells no longer rung on Sundays.*, before PageRank is run.

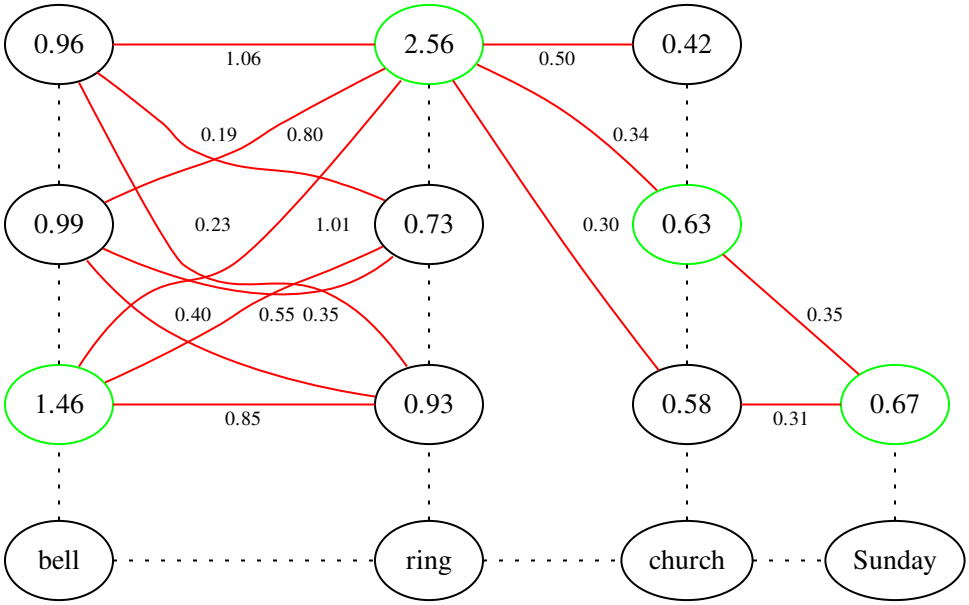

FIGURE 3.4. After PageRank has finished running, the nodes with lighter-colored borders are the senses assigned to the respective words.

CHAPTER 4

EXPERIMENTS AND EVALUATIONS

In this chapter we present the actual experiments and the results obtained thereof. Several experiments were run using the algorithm described in Chapter 3, using a combination of semantic similarity measures and graph centrality algorithms, as described before.

The graph construction works as follows. For each word to be disambiguated, a window is constructed using a few words before and a few words after the word. All the senses of these words are listed such that the senses belonging to one word correspond to a group or a sub-graph. Whenever there is a relationship between senses corresponding to neighboring words (not senses belonging to the same word) based upon the different semantic similarity measures, an edge is drawn between them, with the similarity score being the edge weight. The edge weights are normalized so that a uniform range is used for all the similarity measures.

Each word thus has a window associated with it, including several words before and after that word, which in turn means that each word has a corresponding graph associated with it, and it is *that* word that gets disambiguated after the centrality measures are run on that graph. The values that each node in the graph receives as a result of the centrality algorithm are collected, and out of each group (i.e. the set of senses for one word) the node that has the highest centrality value is assigned as the sense for the word.

4.1. Data

The experiments are primarily carried out on 10 files from the SemCor corpus [23], which were randomly selected while making sure that none of these files were used by Galley and McKeown [13] in their lexical chains experiments. We use this data set of 10 files for development purposes, to determine the optimal settings of window sizes, similarity

measures, and choices for the disambiguation algorithm. Once the settings are determined, the final testing is reported on the SensEval-2 and SensEval-3 English all-words data sets, as well as on the SemCor subset used by Galley and McKeown in [13]. This allows us to compare our results with those of previously reported word sense disambiguation methods that were tested on the same data sets.

4.2. Evaluation of Word Similarity Measures

We started by evaluating the individual disambiguation performance of each similarity measure, using graphs built using one part-of-speech at a time. In these experiments, since the goal is to determine the performance of the similarity measures, and consequently decide on the best combination of measures, we only use one graph-centrality algorithm, namely the indegree algorithm. It is possible that different centrality measures might have different optimal combinations of the similarity measurs, which could be a future direction this work could take.

Several comparative evaluations were run on the development data set; the best results, obtained using a window size of 6, are shown in Table 4.1. Note that all the measures [1] except for **lesk**, work only on nouns and verbs, and thus the results are reported only for these parts-of-speech. As seen in the table, the results indicate that **jcn** tends to work best for nouns as well as for verbs. The method with the highest coverage is **lesk**, which is the only metric that can address adjectives and adverbs. The window size of 6 is empirically determined; we experimented with different windows sizes starting from 2, and went on increasing the size as long as the results kept increasing.

4.3. Normalization

Given that different methods are better for different parts of speech, a natural next step would be to combine several semantic similarity measures into a common graph representation. Before this step can be performed, we need to address aspects concerned with the normalization of the measures.

[1] as taken from the Perl module WordNet::Similarity

part-of-speech	lesk	jcn	res	lin	lch	wup
Noun	2916	**3122**	2076	2552	2492	2446
Verb	1153	**1367**	397	519	1190	1147

TABLE 4.1. Noun and verb true positives returned by the different similarity measures; results obtained on the development data set using a window size of 6.

We performed extensive experiments for normalizing the scores provided by the different similarity measures. As these metrics are fundamentally different, they return values within different ranges. Thus, a vertex in the graph has incoming edges with weights that cannot be directly compared and combined (because, for example, a vertex might have an incident edge which has an edge weight returned by the **lesk** similarity measure, while it has another incident edge which has an edge weight returned by the **jcn** similarity measure). In the following, we concentrate our attention on the **jcn** and **lesk** measures; since these are the ones we use for the combination in our system. The system we developed in the process of this thesis work is versatile and highly parametrized, and graphs could be built using any other combination of similarity measures and those other measures can be normalized using a similar approach if desired.

Our first attempt at normalization was to use the technique proposed by Budanitsky and Hirst [9], and classify the similarity measures as either "connected" or "not connected". In order to achieve this, the values of the different measures were extracted from the graph and plotted individually. Threshold values were then selected in the ranges of the measures; below these thresholds, the similarities are considered 0, i.e. "not connected," and above them, they are considered 1, i.e. "connected." The results obtained using this normalization technique were not satisfactory, perhaps mainly due to the fact that they depend on the value selected for the threshold (as mentioned in [9]). As done in the past, we used the mean values as thresholds, but this technique did not yield favorable results.

Our next attempt was to normalize the results individually according to their ranges. For the **lesk** measure, we observed that the edge weights were in a range from 0 up to an arbitrary large number. However, there were relatively a very few instances where the similarity value returned by this measure was greater than 240. Consequently, values greater than 240 were set to 1, and the rest were mapped onto the interval [0,1]. Similarly, the **jcn** values were found to range from 0.04 to 0.2, with the exception of some "very similar" (same) senses which return a similarity value of the range of millions; and thus the normalization was done with respect to this range. This normalization procedure resulted in a 10% increase in recall on the development data.

4.4. Combination of the Similarity Measures

Given a normalization technique, the next step was to implement a combination of the similarity measures, so as to combine the strengths of each individual metric together into a combined graph representation. We build a graph where we use the similarity metric **jcn** to determine similarity values (and hence the edge weights) between the senses of words tagged as *nouns* as well as those tagged as *verbs*. All the other edges in the graph, including links between *adjectives* and *adverbs*, or links across different parts-of-speech, for example those between nouns and verbs, or verbs and adverbs, are drawn using the **lesk** measure. The results obtained on the entire development data set (namely the 10 randomly selected files from SemCor) using this combination graph are shown in Table 4.2.

To assess the performance of the combined similarity measure, as compared to the individual metrics, two separate evaluations were run on the development data set, where the graph was constructed using the individual metrics **jcn** or **lesk**. Table 4.3 shows the results obtained in each of these experiments. As seen in the table, the combination performs significantly better than the best performing measure, i.e. **lesk**. Note that, when the graphs are built for all the parts of speech and individual similarity measures are used, **lesk** outperforms any other one measure because it returns similarity values between all the permutations of

	noun	verb	adj	adv	all
P	72.47	51.00	68.71	61.92	64.53
R	72.43	50.96	68.71	61.92	64.51
F	72.45	50.98	68.71	61.92	64.52

TABLE 4.2. Results obtained using a combination of similarity methods on the development data.

part-of-speech pairs (e.g., adjective-noun, verb-adverb), while the other metrics fail to do so. However, Table 4.3 proves that a combination of the two measures can be even better than simply using **lesk** or **jcn**. Moreover, the combination makes the entire system faster, as **jcn** tends to perform much faster than **lesk**.

	jcn		lesk		combined	
	n	v	n	v	n	v
P	71.57	50.00	66.85	42.20	**72.47**	**51.00**
R	70.89	48.11	66.21	40.58	**72.43**	**50.96**
F	71.22	49.04	66.53	41.37	**72.45**	**50.98**

TABLE 4.3. Results obtained using individual or combined similarity metrics

4.5. Evaluation of the Graph Centrality Algorithms

All the experiments so far have been carried out using the indegree centrality algorithm. Our next set of experiments is thus concerned with the evaluation of several graph centrality algorithms run on top of the graph built in the previous stage. The algorithms were run on graphs obtained from our previous experiments, namely those obtained by combining the two semantic similarity measures **lesk** and **jcn**. Table 4.4 shows the results obtained with PageRank, closeness, and betweenness; for comparison purposes, we also include the results

obtained using the indegree. Following comparative experiments run on the development data set, we selected a window size of 6 that was found to lead to the best results, and only these results are reported.

	noun	verb	adj	adv	all
indegree					
P	72.47	51.00	68.71	61.92	64.53
R	72.43	50.96	68.71	61.92	64.51
F	72.45	50.98	68.71	61.92	64.52
PageRank					
P	67.68	47.79	68.62	61.85	61.38
R	66.14	47.31	67.70	60.96	60.35
F	66.90	47.55	68.16	61.40	60.86
closeness					
P	31.19	11.94	39.65	57.11	29.64
R	31.17	11.93	39.65	57.11	29.63
F	31.18	11.94	39.65	57.11	29.63
betweenness					
P	55.97	24.37	56.37	62.31	47.52
R	55.94	24.36	56.37	62.31	47.50
F	55.96	24.37	56.37	62.31	47.51

TABLE 4.4. Results obtained using different graph centrality algorithms.

In order to determine if the graph centrality measures make diverse word sense choices, we measured the *Pearson correlation* between pairs of systems based on different graph centrality algorithms. Table 4.5 shows the correlation observed between the senses assigned by

different centrality methods. The table (and the Pearson correlation) indicates that PageRank and Indegree are closely related to each other, which is reasonable because PageRank is an extended version of Indegree.

	Indegree	PageRank	Closeness	Betweenness
Indegree	1.00			
PageRank	0.87	1.00		
Closeness	0.13	0.16	1.00	
Betweenness	0.45	0.45	0.39	1.00

TABLE 4.5. Pearson Correlation between the systems

4.6. Voting Between the Graph Centrality Algorithms

Given the diversity of the results obtained with the graph centrality algorithms, as the final step in our experiments, we implemented a voting scheme among these four measures. Specifically, we obtain the sense predictions from the individual methods - namely indegree, Pagerank, closeness and betweenness, and then apply a voting among these predictions.

We also keep track of which metric has actually predicted a sense out of the four possibilities. If two or more metrics return the same sense, we consider that the voting system has addressed the word, and hence the sense selected by most of the methods is assigned. The voting scheme could be one out of many possibilities. To name a few, we can use two-measures-at-a-time simple agreement; agreement between all four measures; a genetic algorithm running on the four solutions as well as random combination of all four, etc.

As an example for the voting process, consider for instance the word "unambiguous" from the SENSEVAL data set. The indegree, PageRank and betweenness algorithms selected sense #2 (defined in WordNet as *admitting of no doubt or misunderstanding*), while the closeness algorithm chose sense #1 (defined in WordNet as *having or exhibiting a single clearly defined*

meaning). Since most of the methods selected the sense #2, this is also the sense predicted by the our system, which in this case happens to be correct.

We performed several experiments with each of the above possibilities, and the one which gave us the most favorable results is a combination of two measures at a time. The essence of this scheme is that when any two of the algorithms agree, we report that sense to be predicted; otherwise, we do not predict anything at all. The results obtained using the voting scheme are reported in Table 4.6.

	PageRank			Closeness			Betweenness		
	P	R	F	P	R	F	P	R	F
Indegree	65.08	59.35	62.08	97.89	21.91	35.80	76.73	41.36	53.75
PageRank				93.75	21.49	34.98	74.67	39.61	51.76
Closeness							76.94	23.05	35.48

TABLE 4.6. Results obtained using voting over several graph centrality algorithms.

Not surprisingly, this voting scheme leads to high precision and low recall. In particular, combinations involving the closeness measure can lead to a precision figure as high as 97%, but with a low recall of 21%. Combinations involving the betweenness method can give precision figures of 76%, with a recall of 41%. Thus, for the purpose of an overall high-precision system, a combination of systems is desired. On the other hand, for the purpose of an overall high-performance system, with a balance between precision and recall, the individual methods are a better option.

In the following, we run additional evaluations using the best identified individual method (indegree), on several larger data sets.

4.7. Evaluation and Results

The final system, providing the best results on the development data set, integrates two similarity measures (**jcn** for nouns and verbs, and **lesk** for the other parts of speech)

and uses the indegree graph centrality algorithm. We use this system to run larger scale evaluations on three data sets. Specifically, we evaluate the system on the SensEval-2 [28] and SensEval-3 [34] English all-words data, as well as on the set of 74 SemCor files that were used in the experiments reported in [13].[2] The disambiguation results obtained on these three data sets are shown in Table 4.7.

	noun	verb	adj	adv	all
			Senseval-2		
P	69.06	36.64	61.18	59.06	59.01
R	68.06	34.81	60.41	59.06	57.78
F	68.56	35.70	60.79	59.06	58.39
			Senseval-3		
P	61.93	46.24	53.63	100.00	55.05
R	61.93	46.24	53.63	100.00	55.05
F	61.93	46.24	53.63	100.00	55.05
			SemCor		
P	68.70	50.06	68.38	64.18	63.79
R	68.70	50.06	68.38	64.18	63.79
F	68.70	50.06	68.38	64.18	63.79

Table 4.7. Disambiguation results on three test data sets.

4.8. Discussion

During the course of this work we experimented with several different techniques and methodologies. We present below a summary of which ones worked, which ones did not work, and also some efficiency considerations.

[2]Many thanks to Michel Galley for providing us with the list of files used in his experiments.

While building the graphs during the comparison of the different measures of semantic similarity, we noticed that the six measures of semantic similarity all work well for *noun* disambiguation, with **jcn** topping the list and **lesk** coming a close second. These results can be seen in Table 4.1. *Verbs*, on the other hand, showed a more varied range of output, with the results from some measures considerably different from one another. In case of verbs, **jcn** outperformed the other measures, **lesk**, **lch** and **wup** followed closely behind, and then there was a large gap between these results and the results from the other two measures. The execution time of the **lesk** measure was one of the largest in the group, while **jcn** ran much faster.

During the normalization phase, the technique of incorporating in the graph values over the range of 0 and 1 (real numbers) worked much better than a binary classification. During the evaluation of the graph centrality algorithms, the execution speeds of *indegree* and *Pagerank* were comparable and both were reasonably fast. On a data set of the size of SensEval-3, they both typically took 20-30 minutes to finish, which is good considering the potentially huge graphs created by our algorithm (given the enormous number of senses for some content words in WordNet). *Closeness* and *betweenness*, on the other hand, were extremely slow, and took 12-13 hours to finish on the same data set.

During the voting phase in order to combine the results from *indegree*, *Pagerank*, *closeness* and *betweenness*, the combination of two measures at a time performed better than combining three or all four measures together. Further, an experiment with **genetic algorithms** was also performed, with the members of the population being randomly selected from any of the four measures for each word to be disambiguated. To come up with a proper fitness function for the genetic algorithm was the greatest challenge, and several experiments were performed to that end. On top of these the algorithm incorporated random mutation and 2-point crossovers. However, the genetic algorithm did not improve the results over those obtained using indegree.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this work, we described an unsupervised graph-based word sense disambiguation algorithm, which combines several semantic similarity measures and algorithms for graph centrality. To our knowledge, little attempt has been made in the past to address the problem of word sense disambiguation by comparatively evaluating measures of word similarity in a graph theoretical framework.

Through experiments performed on standard sense-annotated data sets, we showed that the right combination of word similarity metrics and the right graph centrality algorithms can significantly outperform methods proposed in the past for this problem. As is stands, the combination of different similarity measures along with the simple centrality algorithm of indegree works very well.

Detailed comparisons of results with the state-of-the-art are presented in Chapter 2. We showed that our system performs as well as some of the best results reported on standard benchmarks and provides significant improvements over several others.

Some of the directions future work can take are - experiment with other measures of semantic similarity or graph centrality (especially global measures which provide an overall score to a sequence of words rather than to an individual word), and experiment with asymmetric similarity and other methods of normalization. Since the different measures of semantic similarity return values over different ranges, and sometimes mean different things, it is challenging to come up with a way to combine them. Another interesting area would be to perform Pagerank, closeness or betweenness on the graphs while comparing the different similarity measures (in order to choose the best combination) rather than indegree. This approach probably will be much slower than indegree, but it might yield very different results.

As mentioned in the thesis, unsupervised WSD is a mine potentially full of great riches, waiting to be explored. If systems are built which learn automatically from existing resources, without asking for manually tagged data sets, WSD research can take a new road and become much more useful and scalable than it currently is. The challenge then, is to make systems more and more accurate, so that they can be used as explicit systems themselves.

BIBLIOGRAPHY

[1] Eneko Agirre and Philip Edmonds, *Text, speech and language technology*, Word Sense Disambiguation: Algorithms and Applications, vol. 33, Springer, 2006.

[2] Eneko Agirre, Bernardo Magnini, Oier Lopez de Lacalle, Arantxa Otegi, German Rigau, and Piek Vossen, *Semeval-2007 task 01: Evaluating wsd on cross-language information retrieval*, Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007) (Prague, Czech Republic), Association for Computational Linguistics, June 2007, pp. 1–6.

[3] Regina Barzilay and Michael Elhadad, *Using lexical chains for text summarization*, In proceedings of the ACL workshop on intelligent scalable text summarization (1997), 10–17.

[4] Stephen P. Borgatti, *Identifying sets of key players in a network*, 2003, pp. 127–131.

[5] Rodrigo A. Botafago, Ehud Rivlin, and Ben Shneiderman, *Structural analysis of hypertexts: Identifying hierarchies and useful metrics*, vol. 10, 1992.

[6] Ulrik Brandes, *A faster algorithm for betweenness centrality*, Journal of Mathematical Sociology 25 (2001), no. 2, 163–177.

[7] S. Brin and L. Page, *The anatomy of a large-scale hypertextual Web search engine*, Computer Networks and ISDN Systems 30 (1998), no. 1–7.

[8] Peter F. Brown, Stephen Della Pietra, Vicent J Della Pietra, and Robert L. Mercer, *Word-sense disambiguation using statistical methods*, 1991, pp. 264–270.

[9] A. Budanitsky and G. Hirst, *Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures*, Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources (Pittsburgh), 2001.

[10] J. Cowie, L. Guthrie, and J. Guthrie, *Lexical disambiguation using simulated annealing*, 1992.

[11] Linton C. Freeman, *A set of measures of centrality based on betweenness*, Sociometry 40 (1977), no. 1, 35–41.

[12] ———, *Centrality in social networks: Conceptual clarification i*, Social Networks 1 (1979), 215–239.

[13] Michel Galley and Kathleen McKeown, *Improving word sense disambiguation in lexical chaining*, Proceedings of the 18th International Joint Conference on Artifical Intelligence IJCAI 2003 (Acapulco, Mexico), 2003.

[14] Guthrie L. Wilkes Y. Guthrie, J. A. and Aidinejadm H., *Subject-dependent co-occurrence and word sense disambiguation*, 1991, pp. 146–152.

[15] J. Jiang and D. Conrath, *Semantic similarity based on corpus statistics and lexical taxonomy*, Proceedings of the International Conference on Research in Computational Linguistics (Taiwan), 1997.

[16] C. Leacock and M. Chodorow, *Combining local context and WordNet sense similarity for word sense identification*, WordNet, An Electronic Lexical Database, The MIT Press, 1998.

[17] M.E. Lesk, *Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone*, Proceedings of the SIGDOC Conference 1986 (Toronto), June 1986.

[18] D. Lin, *An information-theoretic definition of similarity*, Proceedings of the 15th International Conference on Machine Learning (Madison, WI), 1998.

[19] K. Litkowski, *Use of machine readable dictionaries in word sense disambiguation for Senseval-2*, Proceedings of ACL/SIGLEX Senseval-2 (Toulouse, France), 2001.

[20] Kenneth C. Litkowski and Orin Hargraves, *Semeval-2007 task 06: Word-sense disambiguation of prepositions*, Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007) (Prague, Czech Republic), Association for Computational Linguistics, June 2007, pp. 24–29.

[21] D. McCarthy, R. Koeling, J. Weeds, and J. Carroll, *Using automatically acquired predominant senses for word sense disambiguation*, Proceedings of ACL/SIGLEX Senseval-3 (Barcelona, Spain), July 2004.

[22] R. Mihalcea, *Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling*, Proceedings of the Human Language Technology / Empirical Methods in Natural Language Processing conference (Vancouver), 2005.

[23] George Miller, Claudia Leacock, Tengi Randee, and Ross Bunker, *A semantic concordance*, 1993, pp. 303–308.

[24] R. Navigli and P. Velardi, *Structural semantic interconnections: a knowledge-based approach to word sense disambiguation*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27 (2005).

[25] Roberto Navigli, *Semi-automatic extension of large-scale linguistic knowledge bases*, 2005, pp. 548–553.

[26] Roberto Navigli and Mirella Lapata, *Graph connectivity measures for unsupervised word sense disambiguation*, ICJAI (2007), 1683–1688.

[27] Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves, *Semeval-2007 task 07: Coarse-grained english all-words task*, Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007) (Prague, Czech Republic), Association for Computational Linguistics, June 2007, pp. 30–35.

[28] M. Palmer, C. Fellbaum, S. Cotton, L. Delfs, and H.T. Dang, *English tasks: all-words and verb lexical sample*, Proceedings of ACL/SIGLEX Senseval-2 (Toulouse, France), 2001.

[29] S. Patwardhan, S. Banerjee, and T. Pedersen, *Using measures of semantic relatedness for word sense disambiguation*, Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics (Mexico City), February 2003.

[30] Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer, *Semeval-2007 task-17: English lexical sample, srl and all words*, Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007) (Prague, Czech Republic), Association for Computational Linguistics, June 2007, pp. 87–92.

[31] P. Resnik, *Using information content to evaluate semantic similarity*, Proceedings of the 14th International Joint Conference on Artificial Intelligence (Montreal, Canada), 1995.

[32] H. Schuetze, *Automatic word sense discrimination*, 1998.

[33] Ravi Sinha and Rada Mihalcea, *Unsupervised graph-based word sense disambiguation using measures of word semantic similarity*, 2007.

[34] B. Snyder and M. Palmer, *The English all-words task*, Proceedings of ACL/SIGLEX Senseval-3 (Barcelona, Spain), July 2004.

[35] Mark Stevenson and Yorick Wilks, *The interaction of knowledge sources in word sense disambiguation*, vol. 27, pp. 321–349, 2001.

[36] Warren Weaver, *Machine translation of languages*, John Wiley and Sons, 1949.

[37] Stephen Weiss, *Learning to disambiguate*, vol. 9, 1973.

[38] Yorick Wilks, *Formal semantics of natural language*, Cambridge University Press, Cambridge, UK, 1975.

[39] Z. Wu and M. Palmer, *Verb semantics and lexical selection*, Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (Las Cruces, New Mexico), 1994.

[40] D. Yarowsky, *Word sense disambiguation using statistical models of roget's categories trained on large corpora*, 1992, pp. 454 – 460.

[41] ———, *Unsupervised word sense disambiguation rivaling supervised methods*, 1995, pp. 189–196.

[42] George K. Zipf, *Human behavior and the principle of least effort: An introduction to human ecology*, Addison-Wesley, Cambridge, MA, 1949.