# Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity

**Ravi Sinha** and **Rada Mihalcea**
Department of Computer Science and Engineering
University of North Texas
ravisinha@unt.edu, rada@cs.unt.edu

## Abstract

*This paper describes an unsupervised graph-based method for word sense disambiguation, and presents comparative evaluations using several measures of word semantic similarity and several algorithms for graph centrality. The results indicate that the right combination of similarity metrics and graph centrality algorithms can lead to a performance competing with the state-of-the-art in unsupervised word sense disambiguation, as measured on standard data sets.*

## 1 Introduction

Ambiguity is inherent to human language. In particular, word sense ambiguity is prevalent in all natural languages, with a large number of the words in any given language carrying more than one meaning. For instance, the English noun *plant* can mean *green plant* or *factory*; similarly the French word *feuille* can mean *leaf* or *paper*. The correct sense of an ambiguous word can be selected based on the context where it occurs, and correspondingly the problem of *word sense disambiguation* is defined as the task of automatically assigning the most appropriate meaning to a polysemous word within a given context.

In this paper, we describe a graph-based algorithm for unsupervised word sense disambiguation. The algorithm annotates all the words in a text by exploiting similarities identified among word senses, and using centrality algorithms applied on the graphs encoding these sense dependencies. The paper provides a comparative evaluation of several measures of word semantic similarity using a graphical framework. Specifically, we experiment with six knowledge-based measures of similarity and four graph centrality algorithms. The results show that the right combination of similarity measures and graph centrality algorithms can lead to state-of-the-art performance on unsupervised word sense disambiguation.

The paper is organized as follows. We first describe the graph-based method for word sense disambiguation, followed by a description of the similarity measures and graph-centrality algorithms. Next, we present several comparative evaluations carried out on the SENSEVAL data sets, and provide results obtained using each of the similarity measures and centrality algorithms, as well as combinations of these. Finally, we conclude with a discussion of the results.

## 2 Graph-based Centrality for Word Sense Disambiguation

In this section, 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 a generalization of the random-walk sequence data labeling algorithm proposed in our previous work [11]; for the sake of completeness, we reproduce the algorithm here.

Given a sequence of words $W = \{w_1, w_2, ..., w_n\}$, each word $w_i$ with corresponding admissible 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 directed or indirected 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, as illustrated later. Figure 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 a dependency.

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 and identifies the importance of each label (vertex) in the graph. The 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

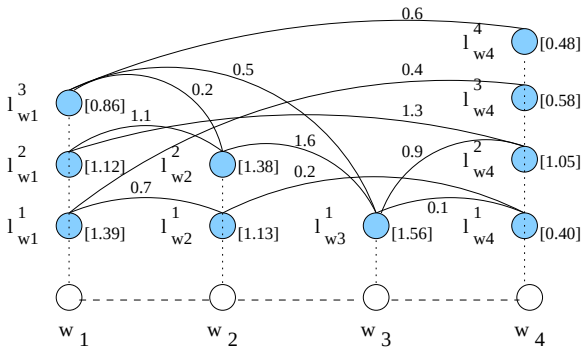


**Figure 1. Sample graph built on the set of possible labels (shaded nodes) for a sequence of four words (unshaded nodes). Label dependencies are indicated as edge weights. Scores computed by the graph-based algorithm are shown in brackets, next to each label.**

word. For instance, for the graph drawn in Figure 1, 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 remarkable property that makes these graph-based algorithms appealing is the fact that they take into account information drawn from the entire graph, capturing relationships among all the words in a sequence, which makes them superior to other approaches that rely only on local information individually derived for each word.

## Word Sense Disambiguation

Given a sequence of words with their corresponding admissible labels, the disambiguation algorithm seeks to identify a graph of label dependencies on which the centrality can be measured, resulting in a set of scores that can be used for label assignment. Algorithm 1 shows the pseudocode for the labeling process. The algorithm consists of three main steps: (1) construction of label dependencies graph; (2) label scoring using graph-based centrality algorithms; (3) label assignment.

First, a weighted graph of label dependencies is built by adding a vertex for each admissible label, and an edge for each pair of labels 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 label dependency is sought. For instance, if $MaxDist$ is set to 3, no edges will be drawn between labels corresponding to words that are more than three words apart, counting all running words. Label 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).

Next, scores are assigned to vertices using a graph-based centrality algorithm. In this paper, we experiment with four centrality algorithms, namely: indegree, Closeness, Betweenness, and PageRank (see Section 4).

Finally, the most likely set of labels is determined by identifying for each word the label that has the highest score. Note that all admissible labels corresponding to the words in the input sequence are assigned with a score, and thus the selection of two or more most likely labels for a word is also possible.

**Algorithm 1** Graph Centrality for Word Sense Disambiguation

**Input:** Sequence $W = \{w_i | i = 1..N\}$
**Input:** Admissible labels $L_{w_i} = \{l^t_{w_i} | t = 1..N_{w_i}\}, i = 1..N$
**Output:** Sequence of labels $L = \{l_{w_i} | i = 1..N\}$, with label $l_{w_i}$ corresponding to word $w_i$ from the input sequence.

**Build graph G of label 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**

**Label assignment**

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

## 3 Measures of Word Semantic Similarity

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. [3] 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 [14]. We provide below a short description for each of these six metrics.

The **Leacock & Chodorow** [7] similarity is determined as:

$$Sim_{lch} = -\log \frac{length}{2 * D} \quad (1)$$

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 [8] 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** [17] 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:

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

The measure introduced by **Resnik** [15] returns the information content (IC) of the LCS of two concepts:

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

where IC is defined as:

$$IC(c) = -\log P(c) \quad (4)$$

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** [9], which builds on Resnik's measure of similarity, and adds a normalization factor consisting of the information content of the two input concepts:

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

Finally, the last similarity metric considered is **Jiang & Conrath** [6]:

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

## 4 Graph-based Centrality Algorithms

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. For weighted graphs, 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:

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

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 [12]. Here, we adopt a different strategy, where the weights on the edges are themselves normalized according to their ranges (see Section 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:

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

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 [5]. 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 [4]. Formally:

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

where $\sigma_{V_b,V_c}$ 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 approxi-

mate well the original definition of betweenness, while being significantly more efficient [1].

Finally, the last graph centrality algorithm we consider is **PageRank**. 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:

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

where $d$ is a parameter that is set between 0 and 1. The typical value for $d$ is 0.85 [2], 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:

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

# 5 Experiments and Results

Several experiments are run using the algorithm described in Section 2. The graph-based word sense disambiguation method is implemented using the WordNet-based similarity measures and the graph centrality algorithms 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 and whenever there is a relationship between these senses based upon the different similarity measures, an edge is drawn between them. The edge weights are normalized so that an 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 the node that has the highest value is assigned as the sense for the word.

## 5.1 Data

The experiments are primarily carried out on the SENSEVAL-2 [13] and SENSEVAL-3 [16] English all-words data sets. Specifically, we use the SENSEVAL-3 as a development data set, and the entire SENSEVAL-2 corpus as a test set. The decision for using SENSEVAL-2 as our final test set is motivated by two main reasons. First, the SENSEVAL-2 all-words data set is significantly larger than the SENSEVAL-3 data set, and thus more appropriate as a test set. Though this might appear counter-intuitive from a supervised training point of view, this is reasonable in our experiments because no training is involved. Our system is unsupervised, and thus we try to determine the optimal value for the parameters on as little data as possible and test on as large a dataset as possible. Second, there is a larger body of previous work on unsupervised word sense disambiguation that was evaluated on the SENSEVAL-2 data set, which can be used as a base of comparison.

## 5.2 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.

Several comparative evaluations were run on a subset of the development data set, namely the first file from SENSEVAL-3; the best results, obtained using a window size of 6, are shown in Table 1. Note that all the measures, 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 and *lch* tends to work best for verbs. The method with the highest coverage is *lesk*, which is the only metric that can address adjectives and adverbs.

### Normalization

Given that different methods are better for different parts of speech, a natural next step would be to combine several methods into a common graph representation. Before

| part-of-speech | lesk | jcn | res | lin | lch | wup |
|---|---|---|---|---|---|---|
| Noun | 83 | **85** | 53 | 49 | 58 | 62 |
| Verb | 53 | 63 | 17 | 21 | **66** | 59 |

**Table 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.**

this step can be performed, we need to address aspects concerned with the normalization of the measures.

We perform 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. In the following, we concentrate our attention on the *lch, jcn* and *lesk* measures; the other measures can be normalized using a similar approach.

Our first attempt at normalization was to use the technique proposed by Budanitsky and Hirst [3], 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 [3]. 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. 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, and thus the normalization was done with respect to this range. Finally, since the *lch* values ranged from 0.34 to 3.33, they were normalized and mapped to the [0,1] scale using this interval. This normalization procedure resulted in a 10% increase in recall on the development data.

## Combination of Similarity Measures

Given a normalization technique, the next step was to implement a combination of the similarity measures, which accounts for the strength of each individual metric. We build a graph where we use the similarity metric *jcn* to draw similarity values between nouns and the similarity metric *lch* to draw similarity values between verbs. All the other edges in the graph, including links between adjectives and adverbs, or links across different parts-of-speech, are drawn using the *lesk* measure. The results obtained on the entire development data set using this combination graph are shown in Table 2.

| | noun | verb | adj | adv | all |
|---|---|---|---|---|---|
| P | 61.10 | 43.31 | 53.02 | 100.00 | 53.43 |
| R | 61.10 | 43.31 | 52.87 | 100.00 | 53.43 |
| F | 61.10 | 43.31 | 52.94 | 100.00 | 53.43 |

**Table 2. Results obtained using a combination of similarity methods.**

To assess the performance of the combined similarity measure, as compared to the individual metrics, three separate evaluations were run on the development data set, where the graph was constructed using the individual metrics *jcn, lch* or *lesk*. Table 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 part-of-speech pairs (e.g., adjective-noun, verb-adverb), while the other metrics fail to do so. However, Table 3 proves that a combination of the three measures can be even better than simply using *lesk*. Moreover, the combination makes the entire system faster, as *jcn* and *lch* both tend to perform faster than *lesk*.

| | jcn | lch | lesk | combined |
|---|---|---|---|---|
| P | 51.57 | 41.47 | 51.87 | 53.43 |
| R | 19.12 | 16.02 | 44.97 | 53.43 |
| F | 27.89 | 23.11 | 48.17 | 53.43 |

**Table 3. Results obtained using individual or combined similarity metrics**

### 5.3 Evaluation of 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 identification of the best graph centrality algorithm. The algorithms are run on graphs obtained from our previous experiments, namely those obtained by combining the three semantic similarity measures *lesk, jcn* and *lch*. Table 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.

Given the diversity of the results obtained with the graph centrality algorithms, as the final step in our experiments,

| | noun | verb | adj | adv | all |
|---|---|---|---|---|---|
| | | | indegree | | |
| P | 61.10 | 43.31 | 53.02 | 100.00 | 53.43 |
| R | 61.10 | 43.31 | 52.87 | 100.00 | 53.43 |
| F | 61.10 | 43.31 | 52.94 | 100.00 | 53.43 |
| | | | PageRank | | |
| P | 60.50 | 41.74 | 53.71 | 100.00 | 52.82 |
| R | 60.23 | 40.85 | 53.71 | 100.00 | 52.30 |
| F | 60.36 | 41.29 | 53.71 | 100.00 | 52.55 |
| | | | closeness | | |
| P | 32.39 | 14.63 | 41.54 | 100.00 | 28.01 |
| R | 32.39 | 14.63 | 41.42 | 100.00 | 28.01 |
| F | 32.39 | 14.63 | 41.47 | 100.00 | 28.01 |
| | | | betweenness | | |
| P | 49.43 | 20.73 | 50.00 | 100.00 | 39.48 |
| R | 49.43 | 20.73 | 50.00 | 100.00 | 39.48 |
| F | 49.43 | 20.73 | 50.00 | 100.00 | 39.48 |

**Table 4. Results obtained using different graph centrality algorithms.**

we implemented a voting scheme among these four measures. Specifically, we obtain the sense predictions from the individual methods, and then apply a voting among these predictions. We also keep track of which metric has predicted which sense. 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.

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.

The results obtained using the voting scheme are reported in Table 5. Note that in this table the precision and the recall are not identical, since there are cases when the algorithms do not agree with one another and thus no prediction can be made.

| | noun | verb | adj | adv | all |
|---|---|---|---|---|---|
| P | 61.22 | 45.18 | 54.79 | 100.00 | 54.86 |
| R | 60.45 | 40.57 | 54.14 | 100.00 | 52.40 |
| F | 60.83 | 42.75 | 54.46 | 100.00 | 53.60 |

**Table 5. Results obtained using voting over several graph centrality algorithms.**

### 5.4 Results on Test Data

The final system, providing the best results on the development data set, integrates three similarity measures (*jcn* for nouns, *lch* for verbs, *lesk* for the other parts of speech) and combines in a voting scheme four graph centrality algorithms (indegree, PageRank, closeness and betweenness).

As a final experiment, the system was evaluated on the test data, consisting of the SENSEVAL-2 data set. The results are shown in Table 6.

| | noun | verb | adj | adv | all |
|---|---|---|---|---|---|
| P | 67.73 | 36.05 | 62.21 | 60.47 | 58.83 |
| R | 65.63 | 32.30 | 61.42 | 60.23 | 56.37 |
| F | 66.24 | 34.07 | 61.81 | 60.35 | 57.57 |

**Table 6. Results on the test set using the final system.**

A comparison of these results with other methods is provided in Table 7, which shows the results obtained using: (1) only the *lesk* measure with an indegree algorithm; (2) a combination of similarity measures with an indegree algorithm; (3) a combination of similarity measures with a PageRank algorithm; (4) the final system, consisting of a combination of similarity measures and a voting over four centrality algorithms.

| | lesk indegree | combined indegree | combined PageRank | combined voting |
|---|---|---|---|---|
| P | 56.86 | 57.72 | 56.58 | **58.83** |
| R | 50.02 | **56.54** | 55.14 | 56.37 |
| F | 53.22 | 57.12 | 55.85 | **57.57** |

**Table 7. Comparison of results on the test set**

## 6 Comparison with Previous Work

Our work is related to the evaluations reported by Navigli and Lapata in [12]. 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 sense, 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 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 on the Senseval-3 data. They obtain a precision and recall of 61.90, 36.10 and 62.80 for nouns, verbs and adjectives respectively, compared to a precision of 61.22, 45.18 and 54.79 and a recall of 60.45, 40.57 and 54.14 for the same parts of speech, as obtained by us.

On Senseval-2, most of the results reported are in the range of 45–53% [11]. In particular, the best performing unsupervised system at Senseval-2 [10] had an overall precision and recall of 45.10%. Hence, our system with its 58.83% precision and 56.37% recall represents a significant improvement.

Our approach builds on a method similar to the one used in [11], which uses, instead of the semantic similarity measures being experimented with here, a measure of similarity based on sense definitions computable on any machine readable dictionary. That approach yielded an overall score of 54.20% on the Senseval-2 dataset. In comparison, the present approach improves significantly over those results, leading to a relative error rate reduction of 8.7%. The Senseval-3 results reported in [11] consisted of a precision and recall of 52.2%, and thus the current system gives a relative error rate reduction of 5.7%.

## 7 Conclusions

In this paper, we described an unsupervised graph-based word sense disambiguation algorithm, which combines several semantic similarity measures and algorithms for graph centrality. To our knowledge, no 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 graph centrality algorithms can significantly outperform methods proposed in the past for this problem. Specifically, our proposed method was found to lead to relative error rate reductions of 5–8% as compared to state-of-the-art methods proposed in previous work.

## Acknowledgments

This work was supported in part by a research grant from the Texas Advanced Research Program (#003594).

## References

[1] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7), 1998.

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

[4] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[5] L. C. Freeman. Centrality in social networks: Conceptual clarification I. *Social Networks*, 1:215–239, 1979.

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

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

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

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

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

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

[12] R. Navigli and M. Lapata. Graph connectivity measures for unsupervised word sense disambiguation. *ICJAI*, pages 1683–1688, 2007.

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

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

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

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

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