Unsupervised Large-Vocabulary Word Sense Disambiguation with Graph-based Algorithms for Sequence Data Labeling

Rada Mihalcea
Department of Computer Science
University of North Texas
rada@cs.unt.edu

Abstract

This paper introduces a graph-based algorithm for sequence data labeling, using random walks on graphs encoding label dependencies. The algorithm is illustrated and tested in the context of an unsupervised word sense disambiguation problem, and shown to significantly outperform the accuracy achieved through individual label assignment, as measured on standard sense-annotated data sets.

1 Introduction

Many natural language processing tasks consist of labeling sequences of words with linguistic annotations, e.g. word sense disambiguation, part-of-speech tagging, named entity recognition, and others. Typical labeling algorithms attempt to formulate the annotation task as a traditional learning problem, where the correct label is *individually* determined for each word in the sequence using a learning process, usually conducted independent of the labels assigned to the other words in the sequence. Such algorithms do not have the ability to encode and thereby exploit dependencies across labels corresponding to the words in the sequence, which potentially limits their performance in applications where such dependencies can influence the selection of the correct set of labels.

In this paper, we introduce a graph-based sequence data labeling algorithm well suited for such natural language annotation tasks. The algorithm simultaneously annotates all the words in a sequence by exploiting relations identified among word labels, using random walks on graphs encoding label dependencies. The random walks are mathematically modeled through iterative graph-based algorithms, which are applied on the label graph associated with the given sequence of words, resulting in a stationary distribution over label probabilities. These probabilities are then used to simultaneously select the most probable set of labels for the words in the input sequence.

The annotation method is illustrated and tested on an unsupervised word sense disambiguation problem, targeting the annotation of all open-class words in unrestricted text using information derived exclusively from dictionary definitions. The graph-based sequence data labeling algorithm significantly outperforms the accuracy achieved through individual data labeling, resulting in an error reduction of 10.7%, as measured on standard sense-annotated data sets. The method is also shown to exceed the performance of other previously proposed unsupervised word sense disambiguation algorithms.

2 Iterative Graphical Algorithms for Sequence Data Labeling

In this section, we introduce the iterative graphical algorithm for sequence data labeling. The algorithm is succinctly illustrated using a sample sequence for a generic annotation problem, with a more extensive illustration and evaluation provided in Section 3.

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 graph-

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 411–418, Vancouver, October 2005.

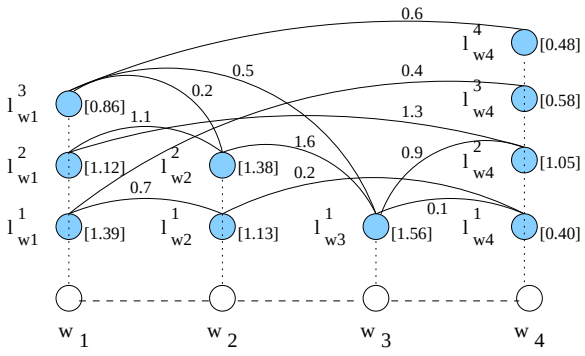

Figure 1: Sample graph built on the set of possible labels (shaded nodes) for a sequence of four words (unshaded nodes). Label dependencies are indicated as edge weights. Scores computed by the graph-based algorithm are shown in brackets, next to each label.

ical 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 recursively determined using an iterative graph-based ranking algorithm, which runs over the graph of labels and identifies the importance of each label (vertex) in the graph. The iterative graphical algorithm is modeling a random walk, leading to a stationary distribution over label probabilities, represented as scores attached to vertices in the graph. These scores are then used to identify the most probable label for each word, resulting in the annotation of all the words in the input sequence. 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 iterative graphical algorithms appealing for sequence data labeling is the fact that they take into account global information recursively drawn from the entire graph, rather than relying on local vertex-specific information. Through the random walk performed on the label graph, these iterative algorithms attempt to collectively exploit the dependencies drawn between *all* labels in the graph, which makes them superior to other approaches that rely only on local information, individually derived for each word in the sequence.

2.1 Graph-based Ranking

The basic idea implemented by an iterative graph-based ranking algorithm 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.

While there are several graph-based ranking algorithms previously proposed in the literature, we focus on only one such algorithm, namely PageRank (Brin and Page, 1998), as it was previously found successful in a number of applications, including Web link analysis, social networks, citation analysis, and more recently in several text processing applications.

Given a graph $G = (V, E)$, let $In(V_a)$ be the set of vertices that point to vertex V_a (predecessors), and let $Out(V_a)$ be the set of vertices that vertex V_a points to (successors). The PageRank score associated with the vertex V_a is then defined using a recursive function that integrates the scores of its predecessors:

$$P(V_a) = (1-d) + d * \sum_{V_b \in In(V_a)} \frac{P(V_b)}{|Out(V_b)|} \quad (1)$$

where d is a parameter that is set between 0 and 1[1].

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. Based on the Ergodic theorem for Markov chains (Grimmett and Stirzaker, 1989), the algorithm is guaranteed to converge if the graph is both aperiodic and irreducible. The first condition is achieved for any graph that is a non-bipartite graph, while the second condition holds for any strongly connected graph – property achieved by PageRank through the random jumps introduced by the $(1-d)$ factor. In matrix notation, the PageRank vector of stationary probabilities is the principal eigenvector for the matrix A_{row}, which is obtained from the adjacency matrix A representing the graph, with all rows normalized to sum to 1: ($P = A^T_{row} P$).

Intuitively, the stationary probability associated with a vertex in the graph represents the probability

[1]The typical value for d is 0.85 (Brin and Page, 1998), and this is the value we are also using in our implementation.

of finding the walker at that vertex during the random walk, and thus it represents the importance of the vertex within the graph. In the context of sequence data labeling, the random walk is performed on the label graph associated with a sequence of words, and thus the resulting stationary distribution of probabilities can be used to decide on the most probable set of labels for the given sequence.

2.2 Ranking on Weighted Graphs

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. The weighted version of the ranking algorithm is particularly useful for sequence data labeling, since the dependencies between pairs of labels are more naturally modeled through weights indicating their strength, rather than binary $0/1$ values. Given a set of weights w_{ab} associated with edges connecting vertices V_a and V_b, the weighted PageRank score is determined as:

$$WP(V_a) = (1-d)+d \sum_{V_b \in In(V_a)} \frac{w_{ba}}{\sum_{V_c \in Out(V_b)} w_{bc}} WP(V_b) \quad (2)$$

2.3 Algorithm for Sequence Data Labeling

Given a sequence of words with their corresponding admissible labels, the algorithm for sequence data labeling seeks to identify a graph of label dependencies on which a random walk can be performed, 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 ranking 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, whose definition depends on the application and type of resources available (see Section 2.4).

Next, scores are assigned to vertices using a graph-based ranking algorithm. Current experiments are based on PageRank, but other ranking algorithms can be used as well.

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-based Sequence Data Labeling

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

repeat
  for all $V_a \in Vertices(G)$ **do**
    $WP(V_a) = (1-d) + d * \sum_{V_b \in In(V_a)} w_{ba} WP(V_b) / \sum_{V_c \in Out(V_b)} w_{bc}$
  end for
until convergence of scores $WP(V_a)$

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

2.4 Label Dependencies

Label dependencies can be defined in various ways, depending on the application at hand and 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, dependencies can be derived by other means, e.g. part-

of-speech probabilities can be approximated from a raw corpus as in (Cutting et al., 1992), word-sense dependencies can be derived as definition-based similarities, etc. Label dependencies are set as weights on the arcs drawn between corresponding labels. Arcs can be directed or undirected for joint probabilities or similarity measures, and are usually directed for conditional probabilities.

2.5 Labeling Example

Consider again the example from Figure 1, consisting of a sequence of four words, and their possible corresponding labels. In the first step of the algorithm, label dependencies are determined, and let us assume that the values for these dependencies are as indicated through the edge weights in Figure 1. Next, vertices in the graph are scored using an iterative ranking algorithm, resulting in a score attached to each label, shown in brackets next to each vertex. Finally, the most probable label for each word is selected. Word w_1 is thus assigned with label $l^1_{w_1}$, since the score of this label (1.39) is the maximum among the scores associated with all its possible labels (1.39, 1.12, 0.86). Similarly, word w_2 is assigned with label $l^2_{w_2}$, w_3 with label $l^1_{w_3}$, and w_4 receives label $l^2_{w_4}$.

2.6 Efficiency Considerations

For a sequence of words $W = \{w_1, w_2, ..., w_n\}$, each word w_i with N_{w_i} admissible labels, the running time of the graph-based sequence data labeling algorithm is proportional with $O(C \sum_{i=1}^{n} \sum_{j=i+1}^{i+MaxDist} (N_{w_i} \times N_{w_j}))$ (the time spent in building the label graph and iterating the algorithm for a constant number of times C). This is order of magnitudes better than the running time of $O(\prod_{i=1}^{n} N_{w_i})$ for algorithms that attempt to select the best sequence of labels by searching through the entire space of possible label combinations, although it can be significantly higher than the running time of $O(\sum_{i=1}^{n} N_{w_i})$ for individual data labeling.

2.7 Other Algorithms for Sequence Data Labeling

It is interesting to contrast our algorithm with previously proposed models for sequence data labeling, e.g. Hidden Markov Models, Maximum Entropy Markov Models, or Conditional Random Fields. Although they differ in the model used (generative, discriminative, or dual), and the type of probabilities involved (joint or conditional), these previous algorithms are all parameterized algorithms that typically require parameter training through maximization of likelihood on training examples. In these models, parameters that maximize sequence probabilities are *learned* from a corpus during a *training phase*, and then applied to the annotation of new unseen data. Instead, in the algorithm proposed in this paper, the likelihood of a sequence of labels is determined during *test phase*, through random walks performed on the label graph built for the data to be annotated. While current evaluations of our algorithm are performed on an unsupervised labeling task, future work will consider the evaluation of the algorithm in the presence of an annotated corpus, which will allow for direct comparison with these previously proposed models for sequence data labeling.

3 Experiments in Word Sense Disambiguation

The algorithm for sequence data labeling is illustrated and tested on an all-words word sense disambiguation problem. Word sense disambiguation is a labeling task consisting of assigning the correct meaning to each open-class word in a sequence (usually a sentence). Most of the efforts for solving this problem were concentrated so far toward targeted supervised learning, where each sense tagged occurrence of a particular word is transformed into a feature vector used in an automatic learning process. The applicability of such supervised algorithms is however limited to those few words for which sense tagged data is available, and their accuracy is strongly connected to the amount of labeled data available at hand. Instead, algorithms that attempt to disambiguate all-words in unrestricted text have received significantly less attention, as the development and success of such algorithms has been hindered by both (a) lack of resources (training data), and (b) efficiency aspects resulting from the large size of the problem.

3.1 Graph-based Sequence Data Labeling for Unsupervised Word Sense Disambiguation

To apply the graph-based sequence data labeling algorithm to the disambiguation of an input text, we need information on labels (word senses) and dependencies (word sense dependencies). Word senses can be easily obtained from any sense inventory, e.g. WordNet or LDOCE. Sense dependencies can be derived in various ways, depending on the type of resources available for the language and/or domain at hand. In this paper, we explore the unsupervised derivation of sense

dependencies using information drawn from machine readable dictionaries, which is general and can be applied to any language or domain for which a sense inventory is available.

Relying exclusively on a machine readable dictionary, a sense dependency can be defined as a measure of similarity between word senses. There are several metrics that can be used for this purpose, see for instance (Budanitsky and Hirst, 2001) for an overview. However, most of them rely on measures of semantic distance computed on semantic networks, and thus they are limited by the availability of explicitly encoded semantic relations (e.g. *is-a*, *part-of*). To maintain the unsupervised aspect of the algorithm, we chose instead to use a measure of similarity based on sense definitions, which can be computed on any dictionary, and can be evaluated across different parts-of-speech.

Given two word senses and their corresponding definitions, the sense similarity is determined as a function of definition overlap, measured as the number of common tokens between the two definitions, after running them through a simple filter that eliminates all stop-words. To avoid promoting long definitions, we also use a normalization factor, and divide the content overlap of the two definitions with the length of each definition. This sense similarity measure is inspired by the definition of the Lesk algorithm (Lesk, 1986).

Starting with a sense inventory and a function for computing sense dependencies, the application of the sequence data labeling algorithm to the unsupervised disambiguation of a new text proceeds as follows. First, for the given text, a label graph is built by adding a vertex for each possible sense for all open-class words in the text. Next, weighted edges are drawn using the definition-based semantic similarity measure, computed for all pairs of senses for words found within a certain distance ($MaxDist$, as defined in Algorithm 1). Once the graph is constructed, the graph-based ranking algorithm is applied, and a score is determined for all word senses in the graph. Finally, for each open-class word in the text, we select the vertex in the label graph which has the highest score, and label the word with the corresponding word sense.

3.2 An Example

Consider the task of assigning senses to the words in the text *The church bells no longer rung on Sundays*[2]. For the purpose of illustration, we assume at most three senses for each word, which are shown in Figure 2. Word senses and definitions are obtained from the WordNet sense inventory (Miller, 1995). All word senses are added as vertices in the label graph, and weighted edges are drawn as dependencies among word senses, derived using the definition-based 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 2. After running the ranking algorithm, scores are identified for each word-sense in the graph, indicated between brackets next to each node. Selecting for each word the sense with the largest score results in the following sense assignment: *The church#2 bells#1*

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

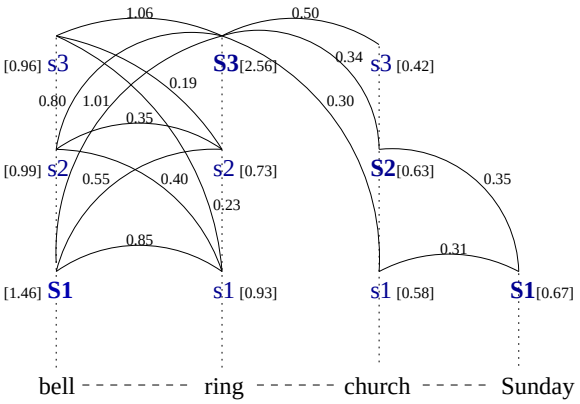

Figure 2: The label graph for assigning senses to words in the sentence *The church bells no longer rung on Sundays.*

[2]Example drawn from the data set provided during the SENSEVAL-2 English all-words task. Manual sense annotations were also made available for this data.

no longer rung#3 on Sundays#1, which is correct according to annotations performed by professional lexicographers.

3.3 Results and Discussion

The algorithm was primarily evaluated on the SENSEVAL-2 English all-words data set, consisting of three documents from Penn Treebank, with 2,456 open-class words (Palmer et al., 2001). Unlike other sense-annotated data sets, e.g. SENSEVAL-3 or SemCor, SENSEVAL-2 is the only testbed for all-words word sense disambiguation that includes a sense map, which allows for additional coarse-grained sense evaluations. Moreover, there is a larger body of previous work that was evaluated on this data set, which can be used as a base of comparison.

The performance of our algorithm is compared with the disambiguation accuracy obtained with a variation of the Lesk algorithm[3] (Lesk, 1986), which selects the meaning of an open-class word by finding the word sense that leads to the highest overlap between the corresponding dictionary definition and the current context. Similar to the definition similarity function used in the graph-based disambiguation algorithm (Section 3.1), the overlap measure used in the Lesk implementation does not take into account stop-words, and it is normalized with the length of each definition to avoid promoting longer definitions.

We are thus comparing the performance of sequence data labeling, which takes into account label dependencies, with individual data labeling, where a label is selected independent of the other labels in the text. Note that both algorithms rely on the same knowledge source, i.e. dictionary definitions, and thus they are directly comparable. Moreover, none of the algorithms take into account the dictionary sense order (e.g. the most frequent sense provided by WordNet), and therefore they are both fully unsupervised.

Table 1 shows precision and recall figures[4] for a context size ($MaxDist$) equal to the length of each sentence, using: (a) sequence data labeling with iterative graph-based algorithms; (b) individual data labeling with a version of the Lesk algorithm; (c) random baseline. Evaluations are run for both fine-grained and coarse-grained sense distinctions, to determine the algorithm performance under different classification granularities.

The accuracy of the graph-based sequence data labeling algorithm exceeds by a large margin the individual data labeling algorithm, resulting in 10.7% error rate reduction for fine-grained sense distinctions, which is statistically significant ($p < 0.0001$, paired t-test). Performance improvements are equally distributed across all parts-of-speech, with comparable improvements obtained for nouns, verbs, and adjectives. A similar error rate reduction of 11.0% is obtained for coarse-grained sense distinctions, which suggests that the performance of the graph-based sequence data labeling algorithm does not depend on classification granularity, and similar improvements over individual data labeling can be obtained regardless of the average number of labels per word.

We also measured the variation of performance with context size, and evaluated the disambiguation accuracy for both algorithms for a window size ranging from two words to an entire sentence. The window size parameter limits the number of surrounding words considered when seeking label dependencies (sequence data labeling), or the words counted in the measure of definition–context overlap (individual data labeling). Figure 3 plots the disambiguation accuracy of the two algorithms as a function of context size. As seen in the figure, both algorithms benefit from larger contexts, with a steady increase in performance observed for increasingly larger window sizes. Although the initial growth observed for the sequence data labeling algorithm is somewhat sharper, the gap between the two curves stabilizes for window sizes larger than five words, which suggests that the improvement in performance achieved with sequence data labeling over individual data labeling does not depend on the size of available context.

The algorithm was also evaluated on two other data sets, SENSEVAL-3 English all-words data (Snyder and Palmer, 2004) and a subset of SemCor (Miller et al., 1993), although only fine-grained sense evaluations could be conducted on these test sets. The disambiguation precision on the SENSEVAL-3 data was measured at 52.2% using sequence data labeling, compared to 48.1% obtained with individual

[3]Given a sequence of words, the original Lesk algorithm attempts to identify the combination of word senses that maximizes the redundancy (overlap) across all corresponding definitions. The algorithm was later improved through a method for simulated annealing (Cowie et al., 1992), which solved the combinatorial explosion of word senses, while still finding an optimal solution. However, recent comparative evaluations of different variants of the Lesk algorithm have shown that the performance of the original algorithm is significantly exceeded by an algorithm variation that relies on the overlap between word senses and current context (Vasilescu et al., 2004). We are thus using this latter Lesk variant in our implementation.

[4]Recall is particularly low for each individual part-of-speech because it is calculated with respect to the entire data set. The overall precision and recall figures coincide, reflecting the 100% coverage of the algorithm.

Part-of speech	Fine-grained sense distinctions						Coarse-grained sense distinctions					
	Random baseline		Individual (Lesk)		Sequence (graph-based)		Random baseline		Individual (Lesk)		Sequence (graph-based)	
	P	R	P	R	P	R	P	R	P	R	P	R
Noun	41.4%	19.4%	50.3%	23.6%	57.5%	27.0%	42.7%	20.0%	51.4%	24.1%	58.8%	27.5%
Verb	20.7%	3.9%	30.5%	5.7%	36.5%	6.9%	22.8%	4.3%	31.9%	6.0%	37.9%	7.1%
Adjective	41.3%	9.3%	49.1%	11.0%	56.7%	12.7%	42.6%	42.6%	49.8%	11.2%	57.6%	12.9%
Adverb	44.6%	5.2%	64.6%	7.6%	70.9%	8.3%	40.7%	4.8%	65.3%	7.7%	71.9%	8.5%
ALL	37.9%	37.9%	48.7%	48.7%	**54.2%**	**54.2%**	38.7%	38.7%	49.8%	49.8%	**55.3%**	**55.3%**

Table 1: Precision and recall for graph-based sequence data labeling, individual data labeling, and random baseline, for fine-grained and coarse-grained sense distinctions.

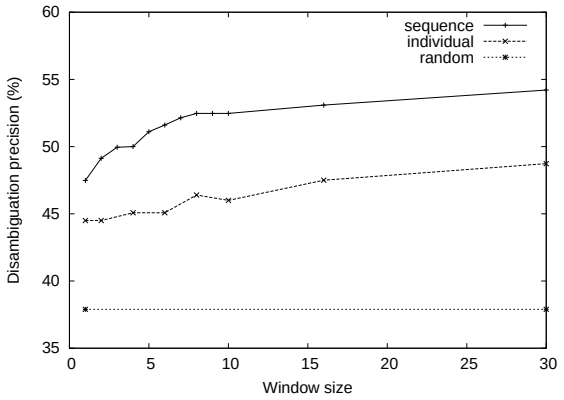

Figure 3: Disambiguation results using sequence data labeling, individual labeling, and random baseline, for various context sizes.

data labeling, and 34.3% achieved through random sense assignment. The average disambiguation figure obtained on all the words in a random subset of 10 SemCor documents, covering different domains, was 56.5% for sequence data labeling, 47.4% for individual labeling, and 35.3% for the random baseline.

Comparison with Related Work

For a given sequence of ambiguous words, the original definition of the Lesk algorithm (Lesk, 1986), and more recent improvements based on simulated annealing (Cowie et al., 1992), seek to identify the combination of senses that maximizes the overlap among their dictionary definitions. Tests performed with this algorithm on the SENSEVAL-2 data set resulted in a disambiguation accuracy of 39.5%. This precision is exceeded by the Lesk algorithm variation used in the experiments reported in this paper, which measures the overlap between sense definitions and the current context, for a precision of 48.7% on the same data set (see Table 1). In the SENSEVAL-2 evaluations, the best performing fully unsupervised algorithm[5] was developed by (Litkowski, 2001), who combines analysis of multiword units and contextual clues based on collocations and content words from dictionary definitions and examples, for an overall precision and recall of 45.1%. More recently, (McCarthy et al., 2004) reports one of the best results on the SENSEVAL-2 data set, using an algorithm that automatically derives the most frequent sense for a word using distributional similarities learned from a large raw corpus, for a disambiguation precision of 53.0% and a recall of 49.0%.

Another related line of work consists of the disambiguation algorithms based on lexical chains (Morris and Hirst, 1991), and the more recent improvements reported in (Galley and McKeown, 2003) – where threads of meaning are identified throughout a text. Lexical chains however only take into account connections between concepts identified in a static way, without considering the importance of the concepts that participate in a relation, which is recursively determined in our algorithm. Moreover, the construction of lexical chains requires structured dictionaries such as WordNet, with explicitly defined semantic relations between word senses, whereas our algorithm can also work with simple unstructured dictionaries that provide only word sense definitions. (Galley and McKeown, 2003) evaluated their algorithm on the nouns from a subset of SEMCOR, reporting 62.09% disambiguation precision. The performance of our algorithm on the same subset of SEMCOR nouns was measured at 64.2%[6]. Finally, another disambiguation method relying on graph algorithms that exploit the

[5] Algorithms that integrate the most frequent sense in WordNet are not considered here, since this represents a supervised knowledge source (WordNet sense frequencies are derived from a sense-annotated corpus).

[6] Note that the results are not directly comparable, since (Galley and McKeown, 2003) used the WordNet sense order to break the ties, whereas we assume that such sense order frequency is not available, and thus we break the ties through random choice.

structure of semantic networks was proposed in (Mihalcea et al., 2004), with a disambiguation accuracy of 50.9% measured on all the words in the SENSEVAL-2 data set.

Although it relies exclusively on dictionary definitions, the graph-based sequence data labeling algorithm proposed in this paper, with its overall performance of 54.2%, exceeds significantly the accuracy of all these previously proposed unsupervised word sense disambiguation methods, proving the benefits of taking into account label dependencies when annotating sequence data. An additional interesting benefit of the algorithm is that it provides a ranking over word senses, and thus the selection of two or more most probable senses for each word is also possible.

4 Conclusions

We proposed a graphical algorithm for sequence data labeling that relies on random walks on graphs encoding label dependencies. Through the label graphs it builds for a given sequence of words, the algorithm exploits relations between word labels, and implements a concept of *recommendation*. A label recommends other related labels, and the strength of the recommendation is recursively computed based on the importance of the labels making the recommendation. In this way, the algorithm simultaneously annotates all the words in an input sequence, by identifying the most probable (most *recommended*) set of labels.

The algorithm was illustrated and tested on an unsupervised word sense disambiguation problem, targeting the annotation of all words in unrestricted texts. Through experiments performed on standard sense-annotated data sets, the graph-based sequence data labeling algorithm was shown to significantly outperform the accuracy achieved through individual data labeling, resulting in a statistically significant error rate reduction of 10.7%. The disambiguation method was also shown to exceed the performance of previously proposed unsupervised word sense disambiguation algorithms. Moreover, comparative results obtained under various experimental settings have shown that the algorithm is robust to changes in classification granularity and context size.

Acknowledgments
This work was partially supported by a National Science Foundation grant IIS-0336793.

References

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

A. Budanitsky and G. Hirst. 2001. 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.

J. Cowie, L. Guthrie, and J. Guthrie. 1992. Lexical disambiguation using simulated annealing. In *Proceedings of the 5th International Conference on Computational Linguistics (COLING 1992)*.

D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing ANLP-92*.

M. Galley and K. McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, Acapulco, Mexico, August.

G. Grimmett and D. Stirzaker. 1989. *Probability and Random Processes*. Oxford University Press.

M.E. Lesk. 1986. 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.

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

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

R. Mihalcea, P. Tarau, and E. Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20st International Conference on Computational Linguistics (COLING 2004)*.

G. Miller, C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, Plainsboro, New Jersey.

G. Miller. 1995. Wordnet: A lexical database. *Communication of the ACM*, 38(11):39–41.

J. Morris and G. Hirst. 1991. Lexical cohesion, the thesaurus, and the structure of text. *Computational Linguistics*, 17(1):21–48.

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

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

F. Vasilescu, P. Langlais, and G. Lapalme. 2004. Evaluating variants of the Lesk approach for disambiguating words. In *Proceedings of the Conference of Language Resources and Evaluations (LREC 2004)*.