Improving Word Sense Disambiguation in Lexical Chaining

Michel Galley and **Kathleen McKeown**
Columbia University
Department of Computer Science
New York, NY 10027, USA
{galley,kathy}@cs.columbia.edu

Abstract

Previous algorithms to compute lexical chains suffer either from a lack of accuracy in word sense disambiguation (WSD) or from computational inefficiency. In this paper, we present a new linear-time algorithm for lexical chaining that adopts the assumption of one sense per discourse. Our results show an improvement over previous algorithms when evaluated on a WSD task.

1 Introduction

Passages from spoken or written text have a quality of unity that arises in part from the surface properties of the text; syntactic and lexical devices can be used to create a sense of connectedness between sentences, a phenomenon known as *textual cohesion* [Halliday and Hasan, 1976]. Of all cohesion devices, lexical cohesion is probably the most amenable to automatic identification [Hoey, 1991]. Lexical cohesion arises when words are related semantically, for example in reiteration relations between a term and a synonym or superordinate.

Lexical chaining is the process of connecting semantically related words, creating a set of chains that represent different threads of cohesion through the text. This intermediate representation of text has been used in many natural language processing applications, including automatic summarization [Barzilay and Elhadad, 1997; Silber and McCoy, 2003], information retrieval [Al-Halimi and Kazman, 1998], intelligent spell checking [Hirst and St-Onge, 1998], topic segmentation [Kan *et al.*, 1998], and hypertext construction [Green, 1998].

A first computational model of lexical chains was introduced by Hirst and St-Onge [1998]. This linear-time algorithm, however, suffers from inaccurate WSD, since their greedy strategy immediately disambiguates a word as it is first encountered. Later research [Barzilay and Elhadad, 1997] significantly alleviated this problem at the cost of a worse running time (quadratic); computational inefficiency is due to their processing of many possible combinations of word senses in the text in order to decide which assignment is the most likely. More recently, Silber and McCoy [2003] presented an efficient linear-time algorithm to compute lexical chains, which models Barzilay's approach, but nonetheless has inaccuracies in WSD.

In this paper, we further investigate the automatic identification of lexical chains for subsequent use as an intermediate representation of text. In the next section, we propose a new algorithm that runs in linear time and adopts the assumption of one sense per discourse [Gale *et al.*, 1992]. We suggest that separating WSD from the actual chaining of words can increase the quality of chains. In the last section, we present an evaluation of the lexical chaining algorithm proposed in this paper, and compare it against [Barzilay and Elhadad, 1997; Silber and McCoy, 2003] for the task of WSD. This evaluation shows that our algorithm performs significantly better than the other two.

2 Lexical Chaining with a Word Sense Disambiguation Methodology

Our algorithm uses WordNet [Miller, 1990] as a knowledge source to build chains of candidate words (nouns) that are related semantically. We assign a weight to each semantic relation; in our work semantic relations are restricted to hypernyms/hyponyms (e.g. between *cat* and *feline*) and siblings (hyponyms of hypernyms, e.g. *dog* and *wolf*). Distance factors for each type of semantic relation prevent the linkage of words that are too far apart in the text; these factors are summarized in Table 1.

The algorithm can be decomposed into three steps: building a representation of all possible interpretations of the text, disambiguating all words, and finally building the lexical chains. First, similarly to [Silber and McCoy, 2003], we process the whole text and collect all semantic relations between candidate words under *any* of their respective senses. No disambiguation is done at this point; the only purpose is to build a representation used in the next stages of the algorithm. Note that this is the only stage where the text is read; all later stages work on this implicit representation of possible interpretations, called a *disambiguation graph* (Figure 1). In this kind of graph, nodes represent word instances and weighted edges represent semantic relations. Since WordNet doesn't relate words but senses, each node is divided into as many senses as the noun has, and each edge connects exactly two noun senses.

This representation can be built in linear time by first building an array indexed by senses of WordNet and processing a text sequentially, inserting a copy of each candidate word into

Semantic relation	1 sent.	3 sent.	1 par.	other
synonym	1	1	0.5	0.5
hypernym/hyponym	1	0.5	0.3	0.3
sibling	1	0.3	0.2	0

Table 1: Weights of relations depending on the kind of semantic relationship and distance (in number of sentences or paragraphs) between two words.

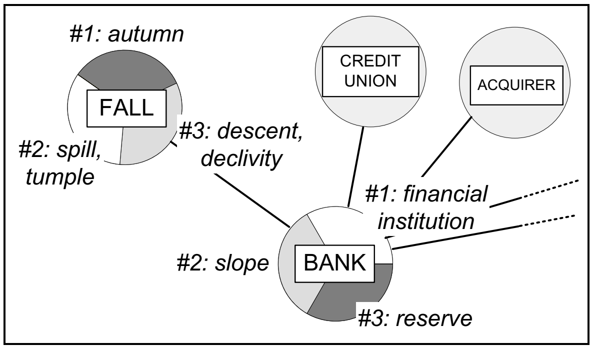

Figure 1: A *disambiguation graph*, an implicit representation of word-sense combinations (in this example, all weights are equal to 1.)

all entries that are valid senses of this word (for example, in Figure 2, the instances *car* and *auto* have been inserted under the same sense in the array). Then, we check whether the noun instance that was just inserted is semantically related to other nouns already present in the array. We do so by looking at hypernyms, hyponyms, and siblings, and if any of these related senses have non-empty entries in the array, we create the appropriate links in the disambiguation graph. For example, in Figure 2, the algorithm finds an hyponymy relation between the noun *taxi* (under its unique sense in the array) and another entry in the array containing *car* and *auto*, so semantic links are added to the disambiguation graph between these two words and *taxi* (shown here attached to the array). Processing all nouns this way, we can create all semantic links in the disambiguation graph in time $O(n)$ (where n is the number of candidate words.)

In the second step, we use the disambiguation graph to perform WSD, enforcing the constraint of one sense per

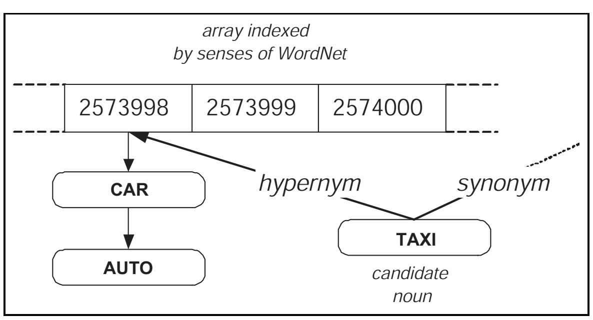

Figure 2: First pass of the algorithm: using an array, we can build the disambiguation graph in linear time.

discourse. We perform the disambiguation of every *word*, instead of disambiguating *word occurrences* as in e.g. [Hirst and St-Onge, 1998; Silber and McCoy, 2003]. We process all occurrences (nodes) of one word at a time, and sum the weight of all edges leaving these nodes under their different senses. The one sense of the word that is assigned the highest score (sum of weights) is considered the most probable sense. For example in Figure 1, the sense of *bank* that has the highest score is *financial institution*. That sense is assigned to all occurrences of the word; in other words, we *impose* the constraint of one sense per discourse. In case of a tie between two or more senses, we select the one sense that comes first in WordNet, since WordNet orders the senses of a word by decreasing order of frequency.

The final step is to build the actual lexical chains by processing the entire disambiguation graph. At this point, we have already assigned a sense to each word, so the last step is to remove from the disambiguation graph all semantic links that connect words taken under their (assumed) wrong senses. Once all such edges have been removed, we are left with the semantic links corresponding to a unique interpretation of the text, and the edges that remain in the graph are the actual lexical chains of our algorithm.[1]

The separation of WSD and lexical chaining into two different sub-tasks is important. All semantic relations, whether correct or incorrect, can be investigated in WSD without necessarily creating incorrect semantic relations in the chaining process. Words are disambiguated by summing weights of semantic relations, but mistakenly counting edges relating words under wrong senses (as in Figure 2 between *fall* and *bank*) doesn't necessarily have the undesirable effect of linking the two words in the same chain. Our assumption is that summing edge weights generally helps in selecting the right senses, e.g. *bank* is disambiguated as a financial institution, and *fall* and *bank* are thus prevented from appearing in the same chain.

3 Evaluation

The evaluation of lexical chains is generally difficult. Even if they can be effectively used in many practical applications like automatic summarization, topic segmentation, and others, lexical chains are seldom desirable outputs in a real-world application, and it is unclear how to assess their quality independently of the underlying application in which they are used. For example, in summarization, it is hard to determine whether a good or bad performance comes from the efficiency of the lexical chaining algorithm or from the appropriateness of using lexical chains in that kind of application. In this section, we evaluate lexical chaining algorithms on the basis of WSD. This arguably is independent of any targeted

[1] Our algorithm has some similarities with Silber and McCoy's algorithm, but it is actually quite different. First, they process each noun instance separately; thus, nothing prevents a noun from having different senses in the same discourse. Second, they process the entire text twice instead of once. In the second pass of their algorithm, they perform WSD and the actual chaining at the same time, whereas we postpone the chaining process until each word has been fully disambiguated.

Algorithm	Accuracy
Barzilay and Elhadad	56.56%
Silber and McCoy	54.48%
Galley and McKeown	62.09%

Table 2: Accuracy of noun disambiguation on semcor.

application, since *any* lexical chaining algorithm has to deal with the problem of WSD. We do not attempt to further evaluate other aspects of chains.

We tested three lexical chaining algorithms on the semantic concordance corpus (semcor), a corpus that was extracted from the Brown Corpus and semantically tagged with WordNet senses. We limited our evaluation to a set of 74 documents of that corpus; this represents about 35,000 nouns. WSD was evaluated on nouns, since all three algorithms that were tested (Barzilay and Elhadad; Silber and McCoy, and ours) build lexical chains with nouns only. We used the original implementation of Barzilay and Elhadad's algorithm, but had to implement Silber and McCoy's algorithm since we didn't have access to their source code. We tested the accuracy of WSD on the set of 35,000 nouns and obtained the results presented in Table 2;[2] accuracy by polysemy is displayed in Figure 3. We can see that our algorithm outperforms Barzilay and Elhadad's, and a one-sided t-test[3] of the null hypothesis of equal means shows significance at the .001 level ($p = 4.52 \cdot 10^{-4}$). Barzilay and Elhadad in turn outperform Silber and McCoy, but this result is not significant at the basic .05 level ($p = 0.0537$).

4 Conclusions

In this paper, we presented an efficient linear-time algorithm to build lexical chains, showing that one sense per discourse can improve performance. We explained how the separation of WSD from the construction of the chains enables a simplification of the task and improves running time. The evaluation of our algorithm against two known lexical chaining algorithms shows that our algorithm is more accurate when it chooses the senses of nouns to include in lexical chains. The implementation of our algorithm is freely available for educational or research purposes at http://www.cs.columbia.edu/~galley/research.html.

Acknowledgments

We thank Regina Barzilay, the three anonymous reviewers, and the Columbia natural language group members for helpful advice and comments.

[2]In Barzilay and Elhadad's algorithm, a word can sometimes belong to two different chains. In order to map each word to one single sense, we applied the *strong chain* sense disambiguation strategy described in [Barzilay, 1997] (i.e. picking the word sense used in the strongest chain).

[3]The samples in the t-test are the WSD accuracies on each individual documents.

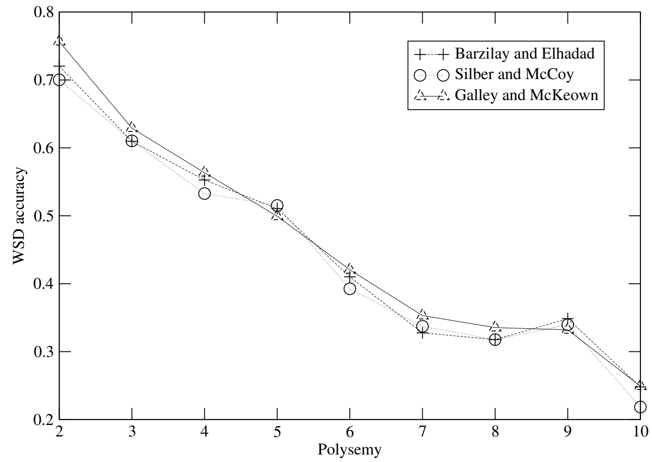

Figure 3: Accuracy by polysemy of the three algorithms.

References

[Al-Halimi and Kazman, 1998] R. Al-Halimi and R. Kazman. Temporal indexing of video through lexical chaining. In *WordNet: An electronic lexical database*. MIT Press, 1998.

[Barzilay and Elhadad, 1997] R. Barzilay and M. Elhadad. Using lexical chains for text summarization. In *Proc. of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL*, 1997.

[Barzilay, 1997] R. Barzilay. Lexical chains for summarization. Master's thesis, Ben-Gurion University, 1997.

[Gale *et al.*, 1992] W. Gale, K. Church, and D. Yarowsky. One sense per discourse. In *Proc. of the DARPA Speech and Natural Language Workshop*, 1992.

[Green, 1998] S. Green. Automated link generation: Can we do better than term repetition? In *Proc. of the 7th International World-Wide Web Conference*, 1998.

[Halliday and Hasan, 1976] M. Halliday and R. Hasan. *Cohesion in English*. Longman, London, 1976.

[Hirst and St-Onge, 1998] G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An electronic lexical database*. MIT Press, 1998.

[Hoey, 1991] M. Hoey. *Patterns of lexis in text*. Oxford University Press, 1991.

[Kan *et al.*, 1998] M.-Y. Kan, J. Klavans, and K. McKeown. Linear segmentation and segment significance. In *Proc. of the 6th Workshop on Very Large Corpora (WVLC-98)*, 1998.

[Miller, 1990] G. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[Silber and McCoy, 2003] G. Silber and K. McCoy. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 29(1), 2003.