Lexical Chaining and Word-Sense-Disambiguation

Lexical Chaining and Word-Sense-Disambiguation

Rani Nelken and Stuart M. Shieber

TR-06-07

Computer Science Group
Harvard University
Cambridge, Massachusetts

LEXICAL CHAINING AND WORD-SENSE-DISAMBIGUATION

RANI NELKEN AND STUART M. SHIEBER
SCHOOL OF ENGINEERING AND APPLIED SCIENCES
HARVARD UNIVERSITY
CAMBRIDGE, MA 02138

ABSTRACT. Lexical chains algorithms attempt to find sequences of words in a document that are closely related semantically. Such chains have been argued to provide a good indication of the topics covered by the document without requiring a deeper analysis of the text, and have been proposed for many NLP tasks. Different underlying lexical semantic relations based on WordNet have been used for this task. Since links in WordNet connect synsets rather than words, open word-sense disambiguation becomes a necessary part of any chaining algorithm, even if the intended application is not disambiguation. Previous chaining algorithms have combined the tasks of disambiguation and chaining by choosing those word senses that maximize chain connectivity, a strategy which yields poor disambiguation accuracy in practice.

We present a novel probabilistic algorithm for finding lexical chains. Our algorithm explicitly balances the requirements of maximizing chain connectivity with the choice of probable word-senses. The algorithm achieves better disambiguation results than all previous ones, but under its optimal settings shifts this balance totally in favor of probable senses, essentially ignoring the chains. This model points to an inherent conflict between chaining and word-sense-disambiguation. By establishing an upper bound on the disambiguation potential of lexical chains, we show that chaining is theoretically highly unlikely to achieve accurate disambiguation.

Moreover, by defining a novel intrinsic evaluation criterion for lexical chains, we show that poor disambiguation accuracy also implies poor chain accuracy. Our results have crucial implications for chaining algorithms. At the very least, they show that disentangling disambiguation from chaining significantly improves chaining accuracy. The hardness of all-words disambiguation, however, implies that finding accurate lexical chains is harder than suggested by the literature.

1. INTRODUCTION

Morris and Hirst (1991) introduced the notion of *lexical chains*, which are sequences of semantically related words. Identifying such chains has been claimed to be a very useful technique for many applications. The intuition is that lexical chains may be able to provide an indication of text meaning without the need for deep syntactic or semantic processing of a text. For instance, consider the following toy two-sentence document:

(1) Tony Blair's favorite drink is tea. He prefers Oolong.

In this sentence, the words “drink”, “tea”, and “Oolong” might be taken to form a lexical chain, as they are all closely related semantically. Morris and Hirst suggested that identifying such chains would be useful for determining text structure. Subsequent work has suggested using lexical chains for many NLP tasks, including word sense disambiguation (WSD) (Okumura and Honda, 1994; Mihalcea and Moldovan, 2001), summarization (Barzilay and Elhadad, 1999; Silber and McCoy, 2002), context-sensitive spelling correction (Hirst and St-Onge, 1998), automatic hypertext link generation (Green, 1999), topic detection and tracking (Stokes, Carthy, and Smeaton, 2004), and more. Following most of the canonical work on lexical chains, we will focus here solely on nouns, though Novischi and Moldovan (2002; 2006) applied lexical chains to verbs for question answering.

Several different semantic relations based on WordNet (Miller, 1995; Fellbaum, 1998) have been used as the basis of lexical chaining, as we review in Section 2. An invariant feature of using WordNet-based relations is that they are defined not on words, but on *synsets*—WordNet’s term for a set of synonymous words, used to represent a concept. Consequently, any chaining algorithm is forced to make a choice between different word senses. For instance, the word “drink” is connected to the chain extracted from Sentence 1 through its “single serving of a beverage” sense (as opposed, e.g., to its “large body of water” sense). Such a choice is required for each of the words in the document, which amounts to performing *all-words* disambiguation. Of course, the other words in the chain might provide strong clues for such WSD, and indeed, chaining algorithms have used this fact as their exclusive source of disambiguation, as we review in Section 3.

In this article we examine the relation between chaining and WSD more closely. Previous algorithms choose word-senses by maximizing chain connectivity. In Section 4, we present a new probabilistic algorithm that balances this requirement of chain connectivity with a choice of probable senses. In Section 5 we perform an evaluation of lexical chaining algorithms with respect to their WSD accuracy. We show that chaining algorithms perform poorly at this task. We show that this lack of WSD accuracy is due to an inherent theoretic limitation of chaining algorithms, by providing an upper bound on their disambiguation potential in Section 6. While not acknowledged by previous research on chaining, the fact that chaining algorithms are poor disambiguators is not surprising. Indeed, all-words WSD has proven to be difficult (Edmonds and Cotton, 2001; Snyder and Palmer, 2004) (in contrast with the simpler “lexical sample” disambiguation of a predefined small set of target words). Our main contribution in this article is exploring the implications of the poor WSD accuracy of chaining algorithms. We address the question of the extent to which WSD accuracy affects chaining accuracy, by introducing a novel intrinsic evaluation criterion for the accuracy of lexical chains in Section 7. We then use that criterion in Section 8 to measure the impact of WSD accuracy on chaining accuracy, showing that in fact, a degradation in WSD accuracy implies a significant degradation in chaining accuracy. Finally, in Section 9 we discuss the implications of this fact both for improving chaining accuracy, and for the applications of chaining algorithms to NLP tasks.

2. SEMANTIC RELATIONS

A key component of lexical chaining is the choice of the underlying semantic relation. Morris and Hirst (1991) used a relation based on Roget's thesaurus to (manually) find chains. Most subsequent work used WordNet (Miller, 1995; Fellbaum, 1998) rather than Roget's, and several different relations have been proposed. See (Budanitsky and Hirst, 2001; Patwardhan, Banerjee, and Pedersen, 2003; Michelizzi, 2005; Budanitsky and Hirst, 2006) for comparative evaluations of many of these relations. It is not our purpose here to define an optimal relation. Instead, we will focus on the particular relations used in previous work, *hso* (Hirst and St-Onge, 1998), *wn* (Barzilay and Elhadad, 1999; Silber and McCoy, 2002; Galley and McKeown, 2003), and *jcn* (Jiang and Conrath, 1998).

Hirst and St-Onge (1998) defined a very inclusive relation, *hso*, which includes repetition or partial overlap between compound nouns, synonymy, single hops along any of WordNet's relations, and multiple hop links along certain "allowable paths". Barzilay and Elhadad (1999) used a more restrictive version, which we call *wn*. The *wn* relation links two word-senses if they are in one of the following relations: they are the same, they are synonyms, one is a hypernym of the other, one is a multi-hypernym of another, one is a sibling of the other in the hypernym hierarchy, or one is a meronym of the other. Much of the subsequent work on lexical chains further restricted *wn*, as shown in Table 1. We will focus here on Barzilay's most inclusive version.

TABLE 1. Variants of *wn* used for chaining

algorithm	repetition, synonym, sibling	hypernym/ hyponym	meronym/ holonym
Barzilay & Elhadad	$\surd$	multiple link	single link
Silber & McCoy	$\surd$	multiple link	$\times$
Galley & McKeown	$\surd$	single link	$\times$

The *hso* and *wn* relations are variations on a similar theme of defining structural relations on WordNet. The *jcn* relation combines WordNet structure with word frequency counts, an idea first proposed by Resnik (1995). Each word instance in a corpus such as the Brown corpus, is counted towards any synset it belongs to as well as any node above it in the hypernym hierarchy. Turning these counts into probabilities, $p(s)$, the relation is defined as:

$$jcn(s_1,s_2) = -\log p(s_1) - \log p(s_2) + 2\log p(lcs(s_1,s_2)) \quad ,$$

where $lcs(s_1,s_2)$ is the lowest common subsumer of s_1 and s_2 in the hypernym hierarchy. While *jcn* has not been used for lexical chaining per se, it is one of the more promising relations as measured with respect to both applications and correlation with human judgments of word similarity (Budanitsky and Hirst, 2006). Following

Budanitsky and Hirst, we binarize the *jcn* relation by choosing a threshold optimizing the correlation with a set of human word-pair similarity judgments. See Appendix A for details.

3. How chaining works

Since WordNet relations are defined on synsets, all lexical chaining algorithms must undertake a form of WSD, even if that is not their stated goal. Hirst and St. Onge's (1998) incremental chaining algorithm works roughly as follows. Linearly traversing the document, each new word occurrence is compared with the currently constructed chains. If one of the word's senses is related to one of the words in these chains, choose that sense of the word and add the word to the chain. If the word can be added to more than one chain, choose between them based on different weightings of the underlying relation (e.g., prefer a synonym to a hypernym), or the length of the chain (preferring longer chains to shorter ones). If the word is unrelated to any existing chain, start a new chain with that word.

Subsequent work superseded this greedy incremental approach with a global one. Barzilay and Elhadad constructed chains similarly to Hirst and St. Onge, but delayed the disambiguation decision until after the chains have been grown. They first construct putative chains entertaining different sense choices, and once the number of possible interpretations exceeds a certain threshold, choose those senses that yield the longest chains, pruning all the other senses.

Silber and McCoy suggested a more efficient algorithm. For each possible synset covered by a word of the document, they construct a "meta-chain", consisting of all the possible word-senses that are related to it, scored by the sum of the relation weights. They then linearly traverse the noun instances of the document, choosing for each one the sense that yields the maximal meta-chain score.

Galley & McKeown follow a similar strategy of first constructing the connectivity graph of all the possible word-senses, and for each word, choosing the word sense that maximizes the sum of edge weights to other nodes in the graph. Since their graph nodes are word types rather than tokens, they enforce the "one sense per discourse" constraint. All other nodes and edges are pruned, and the resulting connected nodes form the chains.

Viewed abstractly, these algorithms attempt to assign word senses to the words in a way that maximizes chain connectivity. Thus, a word-sense that is connected to some chain is preferred over a disconnected one. Differences between the different algorithms reduce to differences in search strategy over this space.

For purposes of generality, we abstract away from some of the details of chaining algorithms. First, we ignore the differential weightings of the basic relations underlying the semantic similarity relations, viewing these instead as binary relations. Second, some chaining algorithms restrict chaining to a predefined contextual window ranging from a fixed number of paragraphs to the entire document. There is an obvious precision/recall tradeoff in choosing the size of this contextual window. Since we are mainly concerned here with upper bounds, we choose the most inclusive settings possible for these relations. At least for some applications,

e.g., context-sensitive spelling (Hirst and Budanitsky, 2005), this window has been shown to yield the best results.

WordNet word-senses are numbered in decreasing order of frequency. We write *w#i* to denote the *i*'th most frequent sense of word *w*. We omit this sense number if it is the first sense. Several of the chaining algorithms use this information for tie-breaking between equally attractive senses. In case of a tie, Silber chooses higher-numbered senses (since they are more specific), while Galley chooses lower-numbered ones. Galley also assigns the first sense to any word left disconnected.

For a word w and synset s, we write $w \in s$ to indicate that w is one of the synonyms in s.

4. ALGORITHM

We now present a new generative algorithm for finding all the lexical chains (comprising only nouns) in a document. Our algorithm generalizes previous work and allows the integration of the lexical semantic relation with the word-sense distribution data.

4.1. **Motivation.**

For simplicity, we first begin with a model for creating all the nouns belonging to a single lexical chain, and then generalize to multiple chains. Conceptually, behind every lexical chain of words, there is a chain of related synsets, which we call an *abstract chain*. Consider the following simple process for composing a chain, consisting of three steps.

(1) An abstract chain of synsets is chosen, by traversing the graph induced by the underlying semantic relation, e.g., *wn*.
(2) Each synset in the abstract chain is manifested as some noun belonging to the synset, creating the concrete chain of nouns.
(3) The rest of the nouns in the document, those not belonging to the chain, which we call the *background nouns*, are generated, and interspersed with those of the chain.

Steps (1) and (2) can be directly implemented as a Hidden Markov Model (HMM), illustrated in Figure 1. Hidden states correspond to synsets s, where each state s may emit any noun such that $w \in s$. Transitions are allowed between synsets related by *wn*.

To complete the process, the HMM must also generate the background nouns, as in Step (3). A self-looping start state, which can emit any noun, generates all the background nouns before the first noun of the chain. Since chains are non-contiguous, we must also generate background nouns between adjacent chain nouns. We do so by allowing each state s_i to also emit any background noun, $w \notin s_i$. This also handles the generation of background nouns following the final noun of the chain.

For example, consider the hypothetical generation of Sentence (1). According to our model, the author first decides on the abstract lexical chain, which we assume consists of the three synsets,[1] $s_1 = \{$*beverage, drink, drinkable, potable*$\}$, $s_2 = \{$*tea*$\}$

[1]For illustration, we denote synsets here literally as sets of synonyms.

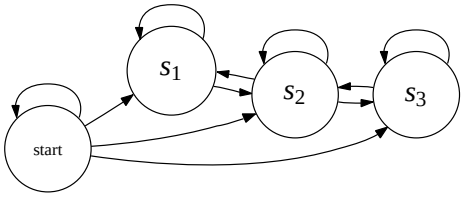

FIGURE 1. Schematic depiction of an HMM for generating the nouns belonging to a lexical chain. The states $s_1, s_2,$ and s_3 correspond to different synsets, such that $wn(s_1, s_2)$ and $wn(s_2, s_3)$ hold, but $wn(s_1, s_3)$ does not. For simplicity, emissions are not shown.

and $s_3 = \{oolong\}$. Then, each of these synsets is manifested as a word. There are multiple choices for s_1, of which the author chooses "drink", but only one possible choice for each of s_2 and s_3. The proper name "Tony Blair", is generated as background. Figure 2 shows a possible path through the HMM, implementing this process.

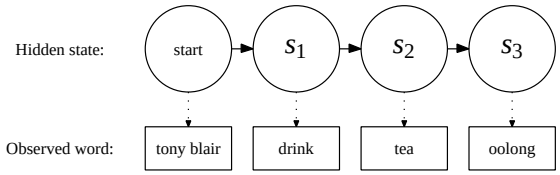

FIGURE 2. Example run through the HMM. Circles depict hidden states, while rectangles depict observed words. Full arrows are used for transitions and dashed arrows for emissions. Starting at the start state, the HMM emits the background proper noun "tony blair", and transitions to s_1, in which it chooses to emit "drink". It then continues to s_2, where it emits "tea", a move made possible since the two synsets are related by *wn* (by hyperonymy). Finally, it moves to s_3, emitting "oolong".

There can be alternative ways in which the HMM might generate the same observed sequence. For instance, "drink" has five different senses in WordNet, given in Table 2, each of which would be represented as a separate HMM state. Thus, a choice of a path through the HMM imposes a particular disambiguation of the words. For instance, the path in Figure 2, according to which "drink" was generated by $s_1 = \{$*beverage, drink, drinkable, potable*$\}$ imposes the choice of drink#3 as the sense of "drink".

Different paths through the HMM will be assigned different probabilities by virtue of the HMM's emission and transition probabilities. We use Viterbi decoding to find optimal paths.

TABLE 2. Senses of "drink" in WordNet, sorted by decreasing frequency

Sense	Corresponding synset	Gloss
drink#1	{drink}	a single beverage serving
drink#2	{drink, drinking, boozing, drunkenness, crapulence}	the act of drinking to excess
drink#3	{beverage, drink, drinkable, potable}	a liquid suitable for drinking
drink#4	{drink}	a large deep body of water
drink#5	{swallow, drink, deglutition}	the act of swallowing

4.2. **Emission probabilities.** Previous algorithms operate as though all word senses are equally likely, with the exception of tie-breaking as described above. We can simulate this approach by using uniform emission probabilities. Instead, we propose to differentially weight different words. Consider first the background nouns. We use a background probability, $p(w)$, estimated over a section of Wall Street Journal text of over 3 million words with add-1 smoothing[2] for these, and in particular we use $p(w)$ for the emission probabilities of the start state.

For a state that corresponds to a synset s, we need to combine two cases: the probability of emitting a chain word, $w \in s$, and the probability of emitting a background word. Consider first the former; we compute the probability of s emitting a chain word, w, using Bayes' rule,

$$p(w \mid s) = \begin{cases} \frac{p(s|w)p(w)}{p(s)} & \text{if } w \in s \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where:

- We use WordNet's sense counts for computing $p(s \mid w)$ with add-1 smoothing, normalizing so that for each s, $\sum_{w \in s} p(s \mid w) = 1$.
- We compute the probability of a synset, $p(s)$, by: $p(s) = \sum_{w \in s} p(s \mid w)p(w)$. Thus, a synset is more likely if the accumulated probability of encountering the words included in it is higher.

In addition to the chain words, each state may also generate background nouns with probability $p(w)$. We can therefore think of the model as a mixture model of $p(w \mid s)$ and $p(w)$ with mixture parameter α, where $0 \leq \alpha \leq 1$, yielding the following final emission probabilities:

$$\tilde{p}(w \mid s) = \begin{cases} \alpha \cdot p(w) + (1-\alpha) \cdot \frac{p(w|s)}{\sum_{w';w' \in s} p(w'|s)} & \text{if } w \in s \\ \alpha \cdot p(w) & \text{otherwise} \end{cases} \tag{2}$$

[2]We chose add-1 smoothing for its simplicity despite its well-known deficiencies. As we will see, these deficiencies seem to be dwarfed by the more fundamental problems illustrated by the upper bound described in Section 6.

The parameter α offers an explicit way of balancing between the two requirements of chaining—maximizing chain connectivity on one hand and choosing probable word-sense on the other hand. For a low value of α, the probability of emitting a background noun becomes low, and the HMM elongates the chain by including more words in it, even at the cost of choosing less probable senses. Conversely, when α is high, emitting background nouns becomes less costly. Hence, the HMM is not forced to make the chain very long, indirectly allowing only more probable senses to be chosen. We learned an optimal value for α by optimizing the HMM's WSD accuracy on ten randomly chosen held-out SemCor documents. Interestingly, best results were achieved when setting α to be 0.9999. We discuss the significance of this choice in Section 5.

4.3. **Transition probabilities.** Previous works used a variety of weighting schemes for scoring chains, depending on the type of relations between the chain nodes (for instance, synonyms are preferred over hypernyms), and their distance in sentences and paragraphs. The actual weights were determined entirely heuristically, and differed between algorithms. Although we could simulate such scoring schemes using the transition probabilities, one would like to have a more principled way of determining these probabilities. Ideally, we would learn such transition probabilities from a corpus manually tagged with lexical chains. Unfortunately, no such resource exists. Even documents tagged with wide-scale disambiguation information as in SemCor (Miller et al., 1993) are rare and insufficiently large for estimating such transition probabilities. We make the major simplification of relying instead on the unigram probabilities, $p(s)$. Thus, the probability of transitioning from state s_1 to a state s_2, related by *wn*, is just $p(s_2)$, normalized over all such s_2's.

$$p(s_2 \mid s_1) = \begin{cases} \frac{p(s_2)}{\sum_{s:wn(s_1,s)} p(s)} & \text{if } wn(s_1,s_2) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

For the start state, we set the following transition probabilities, based on a parameter β:

- The probability of staying in the start state, while generating more background nouns is $p(start \mid start) = \beta$;
- the probability of moving to a first synset is $p(s \mid start) = (1-\beta)p(s)$.

Metaphorically, β signifies the "gravitational pull" of the start state. For a high value of β, the HMM will tend to remain in the start state for a long prefix of the document, assigning words to the background. For lower values of β, the HMM starts the chain earlier. We found an optimal value of $\beta = 0.6$ on the same held-out SemCor set.

For simplicity, our model ignores sentence and paragraph distances, which were included in previous work. We could easily add these by introducing HMM states for emitting sentence and paragraph separator tokens.

4.4. **Finding chains using iterative Viterbi decoding.** The HMM described above provides a generative model for producing chains. For reconstructing chains from a document, we use Viterbi decoding, which finds the most probable path—sequence of synsets—through the HMM that could have generated the document's nouns. To find additional chains, we mask (delete) the nouns spanned by the chain, and rerun Viterbi decoding. We repeat this process until all the nouns in the document have been assigned to some chain, including trivial chains of only one word.

In principle, a single large HMM, with a state for each of WordNet's noun synsets, is applicable for all texts. For efficiency, though, we prefer to construct a separate HMM per document. For a document D, we restrict the states to all possible synsets, S, that include some word in D, under any of its senses. We restrict the possibly emitted words to all the possible manifestations of the synsets in S, even words not in D, which are important for the probability computations. For instance, for the word "drink", we include the synsets corresponding to all five possible senses listed in Table 2, and allow the emission of all the words included in these synsets.

We have implemented the HMM using the AT&T FSM library (Mohri, Pereira, and Riley, 1997), in particular taking advantage of the library's Viterbi decoding utilities for finding chains.

As illustration, Table 3 provides the top ten lexical chains identified by our system for br-e22 (using the sub-optimal parameterization $\alpha = 0.9, \beta = 0.6$). Since chain words are generated by states corresponding to synsets, decoding not only finds chains but also disambiguates the chain words. We report each word sense just once per chain, in order of first appearance in the text.

TABLE 3. Top lexical chains identified by our algorithm on br-e22 using the *wn* relation

(1)	music#1, harmony#2, art#1, dance#1, melody#1, ballet#2, tune#1, stravinsky#2, popularism#1, serialism#1, chorus#3.
(2)	musician#2, composer#1, conductor#1, classicist#1, stravinsky#1.
(3)	world#1, man#4, masses#1, people#1, public#1.
(4)	issue#1, idea#1, center#5, thought#1, goal#1, purpose#1, intention#1, design#5, idea#4.
(5)	paris#1, washington#1.
(6)	era#1, century#1, years#2, epoch#1, stage#1, time-period#1, time#2, age#5.
(7)	pleasure#1, enjoyment#1, enthusiasm#1, tenderness#4, hope#2, mood#1.
(8)	ussr#1, motherland#1, native-land#1.
(9)	quality#1, clarity#2, difference#1, fecundity#3, fruitfulness#1.
(10)	talent#1, flair#1, endowment#1.

There are two interesting trends to note about the chains produced by our system. First, the system tends to prefer higher-frequency senses (i.e., lower WordNet sense indices). Second, unlike Galley and McKeown (2003), we do not explicitly encode the "one-sense-per-discourse" constraint. This property often emerges in the chains

identified by the system, though it is not absolute; for instance, note the two senses of "stravinsky" (the composer, and the music written by him) in Table 3. Since these senses are not linked in WordNet, the two chains are disjoint.

5. CHAINING AND WSD ACCURACY

WSD is inherently tied to chaining. Not only was WSD one of the earliest suggested applications of chaining, Galley and McKeown (2003) have suggested using WSD to evaluate chaining algorithms. Galley and McKeown ran a comparative evaluation on a subset of Semcor (Miller et al., 1993). Since Semcor is the source of the word-sense frequencies used by some of the evaluated algorithms, testing on Semcor might be biased. We therefore reran the systems on the test data of the "English all words" task of the Senseval-3 workshop (Snyder and Palmer, 2004) consisting of three manually disambiguated texts with a total of 918 words.

We received source code for Barzilay's, Silber's, and Galley's systems. To control for different preprocessing, we bypassed any part-of-speech tagging, noun chunking, and paragraph segmentation, using the tagging provided with the evaluation documents instead. Some of the systems only reported word chains, and not the word senses they impose; we augmented them to report the senses as well.[3] Conversely, the implementation of Galley & McKeown's system that we received from the authors only computes the WSD, not the chains themselves.[4]

Results for both datasets are given in Table 4.[5] While Galley & McKeown's algorithm does show clear improvement in WSD accuracy over the previous algorithms, it is decidedly worse than the first-sense heuristic, which simply chooses the most frequent sense. This fact is particularly striking since their algorithm takes word-sense frequency into account as described in Section 3.

TABLE 4. WSD evaluation of chaining algorithms using *wn*

Algorithm	Semcor	Senseval-3
Barzilay & Elhadad	56.6%	53.6%
Silber & McCoy	54.5%	52.9%
Galley & McKeown	63.0%	61.2%
First sense	**76.4%**	**70.4%**

To illustrate the WSD inaccuracy, consider the following chain computed by the Barzilay & Elhadad algorithm (as reimplemented by Silber & McCoy) for one of

[3]Based on such technical considerations we chose to use Silber & McCoy's re-implementation of Barzilay & Elhadad's system rather than the original. We use WordNet 1.7.1 for all experiments involving *wn*.

[4]Barzilay & Elhadad's original system, but not subsequent systems, eliminated certain frequent words from consideration as part of chains (similar to stop-words). We manually applied this heuristic to ignore the words "something", "somebody", "anything", and "anyone".

[5]The result for Galley & McKeown is slightly higher than published due to post-publication improvement (M. Galley, p.c.).

the Senseval-3 documents, CL23, surrounding the concept "man": guy, man, landlord, wife, customer, bourboun#3 (of the royal Bourboun family), end#8, back#4 (end and back as football positions), job#10 (Book of Job), slip#5 (as in "wee slip of a lad"). Since this is a "strong chain"—i.e., one containing many words, the algorithm attempts to elongate it by forcing many words into the chain. It does so by sometimes choosing outrageously improbable senses for the words.

For the *hso* and *jcn* relations we report Michelizzi's results (2005). While Michelizzi's work is not strictly speaking a chaining algorithm, he uses these relations for WSD in a way that is very reminiscent of chaining. Picking a semantic relation, he traverses the document words left to right, and for each word w_i finds the score of the relation between w_i and all the other words in a contextual window, choosing the word-sense that maximizes the sum of these scores. Thus, his results are very pertinent to the discussion. His reported F1-measure on all parts of speech of a 5-document subset of Semcor, and the Senseval-3 dataset are given in Table 5. The results are not only well below the baseline, they are in fact worse than random.[6]

TABLE 5. Michelizzi's WSD results using *hso* and *jcn*

Algorithm	Semcor-5 subset	Senseval-3
hso	25.0%	21.2%
jcn	37.3%	31.5%
Random	41.4%	43.4%
First sense	**76.4%**	**69.3%**

Our HMM algorithm achieves a dramatic improvement in WSD accuracy over all previous algorithms, and is the only one that nearly reaches the baseline, as shown in Table 6. The improvement is achieved by explicitly balancing the requirements of elongating chains and choosing the most probable senses, a balance that is managed explicitly through the use of the parameter α. We achieve the best results when α is set very close to 1, tipping the balance entirely in favor of the probable senses. Decreasing α even slightly immediately degrades WSD accuracy. That such a small change in this parameter has a large effect on accuracy is due to the fact that it is amplified through many iterations of Viterbi decoding. Even at this setting, our model is still subtly different than simply choosing the first sense. Due to our use of Bayes' rule in Equation (1), a synset with fewer word synonyms is preferred over a synset with more. Note that for the *hso* and *jcn* relations, the results are not directly comparable with those of Michelizzi's, since he reports F1 on all parts of speech.

While our algorithm achieves better results than all previous ones, it only reaches the baseline's performance when we set α to be almost 1, under which setting the HMM essentially ignores chains in favor of choosing the most probable senses.

[6]Michelizzi reports better results with the Lesk relation, which is based on word overlap of the synset glosses (Lesk, 1986), but these results too are well below the baseline.

TABLE 6. Top WSD results for HMM algorithm on Senseval-3 documents

Underlying Relation	$\alpha = 0.999$	$\alpha = 0.9999$
wn	67.1%	70.2%
jcn	67.9%	70.1%
hso	61.4%	69.0%

These results point to a basic conflict between chaining and WSD. Previous algorithms relied on the single constraint of maximizing chain length. Our algorithm balances this constraint with the choice of probable senses, but turns out to work best only when the balance is set totally in favor of the probable senses. In Section 6 we show that this conflict is not just apparent in existing algorithms, but is in fact inherent to chaining, by computing an upper bound on the WSD potential of chains.

6. UPPER BOUND ON THE WSD POTENTIAL OF CHAINS

Fixing a semantic relation, let us look more closely at the graph induced by the semantic relation on the correct disambiguation of a document. In this graph, the nodes are the word-senses (types, not tokens), *w#i*, where there is an edge between different word-senses if they are in the relation we are examining. For example, Figure 3 shows this graph for Senseval document CL23 using the *jcn* relation, with distances proportional to the *jcn* value.[7] The graph is rendered using Pajek (Batagelj and Andrej, 2002) using the Fruchterman and Reingold method (1991), with some manual jitter to improve readability.

Strikingly, many nodes in this graph are disconnected, corresponding to word-senses that are not connected to any different correct word-senses in the document. These disconnected word-senses are extremely problematic for chaining algorithms. To see why, consider an instance of a word *w*, the true sense of which, *w#i*, is disconnected in the graph. Whichever chain *w* will be added to, it may never be through sense *i*. Consequently, we can safely say that *w#i* will not be correctly disambiguated solely by virtue of chaining.[8]

Once we also add word-sense frequency into the picture, this fact does not imply that any disconnected words are erroneously disambiguated. We can, however, identify two sub-classes of the disconnected word-senses as extremely likely errors even when incorporating sense frequencies:

- If *w#i* is disconnected and $i > 1$, then *w#i* is not part of a chain, and is also not the most frequent sense. Hence a chaining algorithm has no reason to choose it as the correct sense.

[7] We use WordNet 2.0 for all experiments involving *jcn*.

[8] Multiple occurrences of *w#i* might be chained together in a chain consisting solely of these word-senses, but in that case, there would be no reason to choose sense *i*. Another possibility is for *w#i* to be connected to some incorrect sense of another word $w'\#i'$. Since such cases lead to another disambiguation error, we discount them as noise.

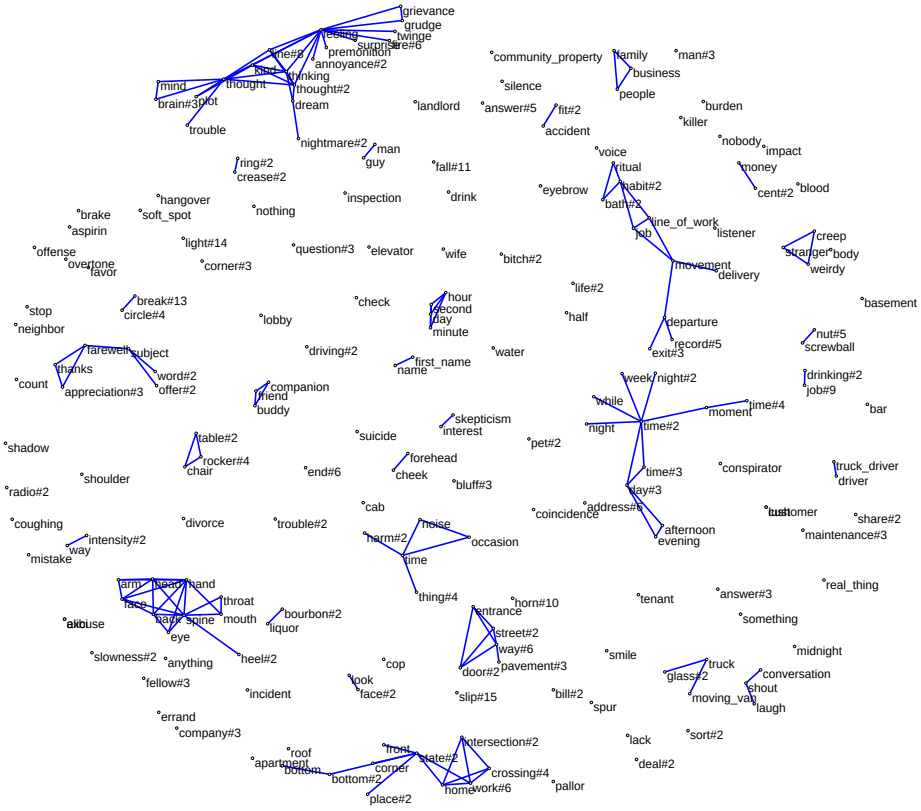

FIGURE 3. Connectivity graph of Senseval document CL23. Nodes are word-sense tokens annotated with the sense number unless it is the first sense. Edges correspond to the *jcn* relation with distances proportional to the value of the relation.

- If *w#i* is disconnected and some other sense of the word, *w# j*, *is* connected, a chaining algorithm would prefer *w# j* over *w#i*.

We call the union of these two classes the *unreachable* senses. We can quantify these for the Senseval-3 documents as shown in Table 7. For the *hso* relation there are very few unreachable senses (at least when allowing the entire document as potential context). For the *wn* and *jcn* relations, however, at least 23.6% of the word-senses are unreachable, and thus we can safely count them as likely errors. Of course, there is no guarantee that a chaining algorithm would be able to correctly disambiguate the reachable senses. In fact, to beat the baseline of 70.4%, a chaining algorithm must get at least 92.1% $(70.4\%/(1-23.6\%))$ accuracy on the reachable senses, which is unlikely to be achieved by chaining alone. Thus, chaining algorithms using either of these two relations are unlikely to beat the first-sense baseline.

TABLE 7. Distribution of disconnected senses on Senseval-3 data. Connectedness is computed separately for each of the 3 documents. Numbers and percentages are given cumulatively over the 918 nouns in the documents.

Relation	Disconnected		Unreachable	
	number	percentage	number	percentage
hso	51	5.6%	48	5.2%
wn	365	39.8%	230	25.1%
jcn	297	32.4%	217	23.6%

The import of these results is that not only do chaining algorithms yield poor WSD results in practice, they are unlikely to beat the baseline even in theory, at least for the *wn* and *jcn* relation. For the *hso* relation, this upper bound is uninformative, but note that since this relation is so inclusive (especially when considering the entire document as context window), it is also likely to connect many incorrect senses, which could explain its weaker performance in practice.

To illustrate this point, consider the following lexical chain extracted from CL23:[9] {moving_van, brakes, horn#10, truck, driver, cab, driver, truck_driver, driver, truck}. Under both the *wn* and the *jcn* relations, the word-sense horn#10 (automobile horn) is disconnected from all the other correct word-senses in CL23, and in particular it is disconnected from the words in this chain (unfortunately, horn is not listed as a meronym of truck, just of car). Thus, no chaining algorithm based (solely) on either *wn* or *jcn* can correctly disambiguate this word.

Under the much more inclusive *hso* relation, horn#10 is related to "brake", and "truck", but also to quite a few other words (e.g., way#6, elevator, table#2), and other senses of the word horn are related to even more words (e.g., pet, stop, hand, blood). It is of course the task of the chaining algorithm to filter the correct connections from these possible ones. Thus, while our upper bound says nothing about the *hso* relation, the challenge for a chaining algorithm based on *hso* is clear. It is entirely conceivable that such an algorithm would be able to find the correct senses, although there is no evidence that it can.

What is the significance of the poor WSD accuracy for the accuracy of chaining itself? Is it possible to achieve accurate chaining despite poor WSD? For many NLP tasks such as machine translation, the contribution of WSD is inconclusive (Carpuat and Wu, 2005). It would be tempting to dismiss the significance of the poor WSD by simply resolving not to use chains for WSD. The problem, however, runs much deeper. Errors in WSD immediately propagate to errors in chaining. For example, if we are unable to connect horn#10 to the words truck, driver, etc., then we can never find the exact chain above. Moreover, since other senses of horn are connected to other words, we are likely to construct another

[9] We thank one of the anonymous reviewers for graciously annotating the document for lexical chains.

chain based on a bogus sense. To quantify the effect of these errors in WSD on the accuracy of chaining, we need an intrinsic evaluation measure.

7. INTRINSIC EVALUATION OF LEXICAL CHAINS

Evaluation of chaining algorithms has traditionally been task-based. Barzilay and Elhadad (1999) used human judges to evaluate the quality of the summaries produced by their algorithm. Silber and McCoy (2002) separately chained texts and their human-generated abstracts, and measured the overlap between the highly-scoring chains in the main text with those of the abstract. Being extrinsic, such evaluation provides only indirect evidence of the accuracy of the produced chains. Moreover, summary evaluation raises its own set of issues (Mani, 2001).

Ideally, a direct intrinsic evaluation metric would consist of a set of documents manually marked for their correct chains, and a way of comparing two chainings on the same document. Unfortunately, manual chaining annotation efforts to date have been extremely limited (Hollingsworth and Teufel, 2005; Morris and Hirst, 2005). Moreover, chaining evaluation is complicated by the lack of a crisp definition of what constitutes a chain.

We propose to compensate for the lack of sufficient manual annotation by providing an automated approximation. Consider the connected components (CC) of the graph defined in Section 6 on the correct word-sense types. These connected components, in turn, partition the document into sets of word-sense tokens that are related by the underlying relation. These sets are plausible approximations of lexical chains, since they define collections of word tokens that are semantically related. For example, "hour", "minute", "second", and "day" form a connected component in the graph, and the corresponding approximate lexical chain would be: { hour, minute(2), second, day }, denoting the number of multiple occurrences in parentheses.

Some additional chains derived this way from the document are: { friend, companion, buddy(2) }, { mouth, arm, spine, throat, hand, back, face, heel#2, eye(10), head } , { fire#6, surprise, premonition, nightmare#2, kind, annoyance#2, thinking, line#8, thought#2(2), dream(3), trouble, mind, grievance(3), plot, grudge(2), feeling, twinge, brain#3, thought }, { driver(3), truck_driver }.

Using the same relation used by an algorithm for its evaluation might introduce bias with respect to the "true" chains, as the underlying relation might itself be suspect. Such bias, however, is really only in favor of the algorithms, as the CCs only allow chains that might potentially be found by the algorithm.

Note that for this approximation to be meaningful, the graph must be sufficiently sparse. Thus, while this approximation works well for *wn* and *jcn*, it is less meaningful for *hso*, which is so densely connected.

In addition, we need a way to compare chains computed by any algorithm with the CCs. Hollingsworth and Teufel (2005) make a first step in this direction by suggesting a metric for comparing different chains based on (partial) term overlap. Fortunately, we can complete this measure to give a fuller picture by applying standard measures of clustering evaluation to chaining evaluation (Zhao and Karypis,

2002). The only caveat with respect to treating chains as clusters is that clusters do not take into account the linear ordering of the words in the chain or in the document. This fact, though, does not seem to diminish from the validity of the evaluation.

Let the gold chaining approximation as computed by the connected components consist of gold chains $G_1,\ldots,G_q$, of lengths $m_1,\ldots,m_q$, and the chains computed by an algorithm be $C_1,\ldots,C_k$, of lengths $n_1,\ldots,n_k$, respectively. Let n_{ij} be the number of word tokens that were assigned to chain C_i and belong to gold chain G_j, and let n be the total number of word-sense types.

The precision of chain C_i with respect to gold chain G_j is: $p_{ij} = \frac{n_{ij}}{n_i}$. The *purity* of a chain is defined as $Purity(C_i) = \max_j(p_{ij})$, intuitively capturing C_i's precision with respect to the gold chain it fits best. The purity of the entire chaining is the weighted average of the individual chain purities: $Purity = \sum_i \frac{n_i}{n} Purity(C_i)$. A second useful measure is *entropy*. The entropy of C_i is $H(C_i) = -\sum_{j=1}^{q} p_{ij} \log p_{ij}$.

The entropy of the entire set of chains is defined as the weighted average of the individual chain entropies: $entropy = \sum_{i=1}^{k} \frac{n_i}{n} H(C_i)$.

We conducted an intrinsic evaluation of the chaining algorithms using these evaluation criteria. Evaluation is done on Document CL23 with respect to the *wn* relation. For each algorithm, we have a choice of either using these lexical chains, or ignoring these and super-imposing the CC construction on top of the word-senses emitted by the algorithm. To consistently accommodate both the baseline and Galley's algorithm, which do not actually output lexical chains, we chose the latter option. Results are shown in Table 8. The ranking of the algorithms according to these criteria is consistent with their ranking according to WSD accuracy. In particular, while the HMM algorithm does better than all previous ones, it performs similarly to the baseline.

TABLE 8. Intrinsic evaluation of chaining algorithms using the CC construction with respect to the *wn* relation over CL23.

Algorithm	Purity	Entropy
Barzilay & Elhadad	0.576	0.214
Silber & McCoy	0.524	0.319
Galley & McKeown	0.617	0.243
HMM	**0.786**	0.070
First sense	0.780	**0.059**

8. POOR WSD ACCURACY IMPLIES POOR CHAINING ACCURACY

Armed with an intrinsic evaluation method for lexical chains, we can now quantify the effect of disambiguation accuracy on chaining accuracy. We performed the following experiment. For Document CL23, we first computed the connected components on the correct disambiguation and used these as our approximation of gold chains. We proceeded to progressively perturb some of the word senses of the

document, each time replacing a correct sense with an incorrect one. For each such perturbation, we recomputed the connected components, and compared the result with the original with respect to purity and entropy. The result is shown in Figure 4 using the *jcn* relation. Results for the *wn* relation are similar. When all the word-senses are the correct ones, purity is 100% and entropy is 0. As more and more word-senses are perturbed, purity degrades linearly. Entropy likewise increases for a while, but note that just before 80% of the word-senses are incorrect, it starts decreasing. This is just an artifact of the entropy computation; when so many of the senses are incorrect, p_{ij} is often 0, in which case, by convention $p_{ij} \log p_{ij}$ is also 0 (as it would be if p_{ij} had been 1).

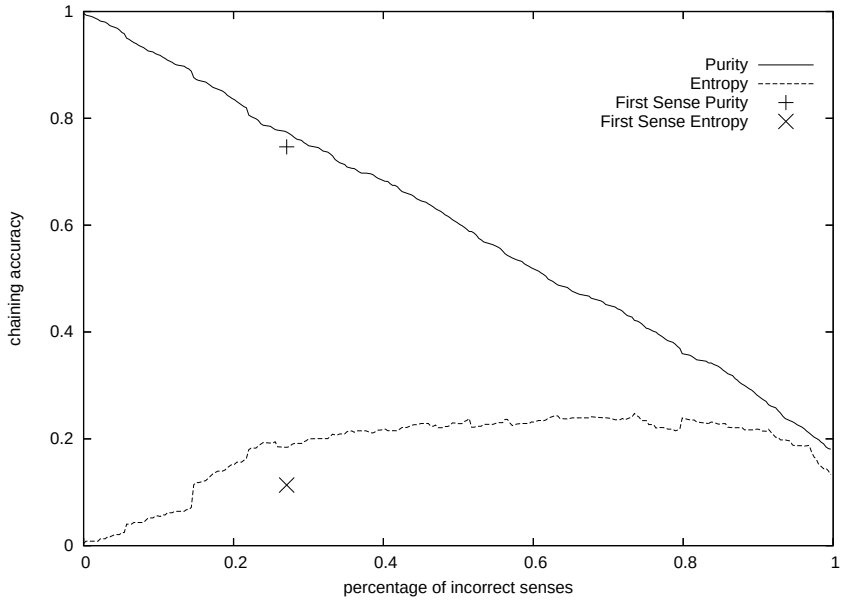

FIGURE 4. Effect of WSD accuracy on chaining purity and entropy

Figure 4 verifies the intuition that a degradation in WSD accuracy leads directly to a severe degradation in chaining accuracy, validating our claim that WSD accuracy is essential for chaining accuracy. As a baseline, we applied the CC construction on the first-senses, and show the resulting purity and entropy scores.[10]

9. CONCLUSION

The relation between lexical chaining and WSD is a subtle one. We have shown that the strategy adopted by chaining algorithms for WSD leads to poor WSD accuracy in practice, and is in fact theoretically unlikely to succeed at this task. The obvious conclusion would simply be not to use chaining algorithms for WSD. While running contra Okomura and Honda, this conclusion by itself is hardly surprising

[10]While the purity value for the baseline is close to the randomized curve, the entropy value is lower. This is because the baseline assigns a uniform sense choice for all occurrences of the same word.

or interesting. Our main point in this article, however, goes beyond that simple conclusion to examine the implications of this fact for chaining research. We have shown that the choice of incorrect word-senses leads to incorrect assignment of words to chains. We show this both for an example in Section 5 and quantitively in Section 8. Thus, WSD accuracy turns out to be crucial for any chaining application, even if WSD is not their stated goal.

We can summarize our contribution with the slogan that chaining is "WSD-complete", in the sense that on one hand, accurate chaining implies accurate disambiguation. On the other hand, given accurate disambiguation, chaining can be achieved simply and efficiently, by connecting the senses. Thus, a positive take-home message of this article is a simple way to improve both the accuracy and the efficiency of chaining—first apply the best possible WSD algorithm, and then connect the resulting senses, for instance using the CC construction. This approach improves not only the WSD accuracy, but consequently also the accuracy of the chains. Additionally, it obviates much of the discussion in the literature on the optimal search strategy for chains.

Unfortunately, the fact that robust open WSD has proved to be so difficult implies that the resulting chains will still be far from accurate. Thus, if current state-of-the-art disambiguation algorithms achieve WSD accuracy just above the baseline, we can expect the chaining purity and entropy to be slightly better than the baseline point in Figure 4.

The ultimate question that remains open is the effect of this inaccuracy on applications, such as the application of lexical chains to summarization. As we have shown, WSD inaccuracy propagates to chaining inaccuracy as measured by decreased purity and increased entropy. If the chains themselves are noisy, how accurate are the summaries based on them? This is an empirical question, which we feel has not been adequately addressed by previous evaluations. Recently, more robust summarization evaluation techniques have been developed (Lin, 2004), opening the way to a principled analysis of this dependency.

A final point relates to the underlying semantic relation. Our results are general in that due to the structure of WordNet, any relation connecting WordNet synsets must apply some form of disambiguation, and is thus prone to the hardness of open disambiguation. For specificity, we have given results for three different relations. Obviously, this analysis does not exhaust either the range of relations discussed in the literature or the range of possible relations. All we have shown is that for restrictive enough relations defined on WordNet senses, WSD is a serious cause for concern when constructing lexical chains. Our methodology can similarly be applied to additional relations. For more inclusive relations, such as the *hso* relation, we run the risk of running into the converse problem of having too many potential connections. Although our formal criterion does not apply to this type of relation, this risk might explain why this relation does not perform as well as some of the more inclusive ones in evaluations.

Nothing in our analysis precludes the possibility that new lexical semantic relations might yield better chaining accuracy. Morris and Hirst (2004) have called for an approach that goes beyond WordNet-based relations to "non-classical relations",

such as the relation between “dog” and “bark”. Such relations may turn out to be much more informative. Moreover, they may be defined directly on words rather than word-senses, and thus might avoid the unfortunate dependence on WSD. One recent intriguing direction is the mining of lexical semantic relations from search engine page counts (Cilibrasi and Vitanyi, 2007; Bollegala, Matsuo, and Ishizuka, 2007), though as Kilgarriff (2007) warns, such raw counts should be taken with a grain of salt.

10. Acknowledgments

We wish to thank Regina Barzilay, Greg Silber, and Michel Galley for providing the source code for each of their respective lexical chaining systems along with much helpful information regarding their work. We also thank Oliver Steele for providing an open source Python interface to WordNet and Ted Pedersen and his team for providing an open source Perl interface to some of the semantic relations based on WordNet. We thank Barbara Grosz for her comments on a previous version of this article. We also thank the members of the UPENN Clunch forum for their comments. Finally, we are extremely appreciative of the comments we received from anonymous referees of this paper. This work was supported in part by grant IIS-0329089 from the National Science Foundation.

Appendix A. Choice of threshold for *jcn*

We follow Budanitsky and Hirst (2006) in choosing a threshold for *jcn* based on correlations with human word-pair similarity judgments (Rubenstein and Goodenough, 1965). This process requires extending the *jcn* relation from senses to words, $jcn(w_1, w_2)$, by choosing the pair of senses with maximal *jcn*. Note that this form of disambiguation of a disembodied pair of words is very different from the kind of WSD of words within a document context. This approach yields a high level of correlation, which they optimize to determine a threshold. They report a correlation of -0.781. Using our implementation on a newer version of WordNet (2.0 vs. 1.5) yields a correlation of -0.8535. To replicate Budanitsky and Hirst's choice of a threshold, we define

$$jcn'(w_1, w_2) = \begin{cases} 1 & \text{if } jcn(w_1, w_2) > th \\ 0 & otherwise \end{cases} .$$

Choosing any value between 5.5 and 7.5 for th yields an optimal correlation of -0.8544, slightly better than the original correlation value without a threshold. We chose the most inclusive of these, $th = 7.5$.

References

Barzilay, Regina and Michael Elhadad. 1999. Using Lexical Chains for Text Summarization. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*. The MIT Press, pages 111–121.

Batagelj, Vladimir and Mrvar Andrej. 2002. *Pajek — Analysis and Visualization of Large Networks*, volume 2265. Springer-Verlag, January.

Bollegala, Danushka, Yutaka Matsuo, and Mitsuru Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *Proceedings of the Sixteenth International World Wide Web Conference(WWW2007)*, pages 757–766, Banff, Alberta, CANADA, May.

Budanitsky, A. and G. Hirst. 2001. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics*, pages 29–34, Pittsburgh, June.

Budanitsky, Alexander and Graeme Hirst. 2006. Evaluating WordNet-based measures of semantic distance. *Computational Linguistics*, 32(1):13–47, March.

Carpuat, Marine and Dekai Wu. 2005. Word sense disambiguation vs. statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 387–394, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Cilibrasi, Rudi L. and Paul M.B. Vitanyi. 2007. The Google similarity distance. *IEEE Trans. Knowledge and Data Engineering*, 19(3):370–383.

Edmonds, Philip and Scott Cotton. 2001. Senseval-2: Overview. In *Proceedings of the Second International workshop on Evaluating Word Sense Disambiguation Systems*, New Brunswick, NJ. Association for Computational Linguistics.

Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Fruchterman, Thomas M. J. and Edward M. Reingold. 1991. Graph drawing by force-directed placement. *Software—Practice and Experience*, 21(11):1129–1164.

Galley, Michel and Kathleen McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proceedings of 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*.

Green, Stephen J. 1999. Building hypertext links by computing semantic similarity. *IEEE Transactions on Knowledge and Data Engineering*, 11(5):713–730.

Hirst, Graeme and Alexander Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87–111, March.

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

Hollingsworth, W. and S. Teufel. 2005. Human annotation of lexical chains: Coverage and agreement measures. In *Proceedings of "ELECTRA": Methodologies and Evaluation of Lexical Cohesion Techniques in Real-world Applications, SIGIR 2005*, pages 26–32, Salvador, Brazil, August.

Jiang, J.J. and D.W. Conrath. 1998. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics, ROCLING X*, Taiwan.

Kilgarriff, Adam. 2007. Googleology is bad science. *Computational Linguistics*, 33(1):147–151.

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

Lin, Chin-Yew. 2004. Rouge: A package for automatic evaluation of summaries. In Marie-Francine Moens and Stan Szpakowicz, editors, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Mani, Inderjeet. 2001. *Automatic Summarization*. John Benjamins Publishing Company, Amsterdam/Philadelphia.

Michelizzi, Jason. 2005. Semantic relatedness applied to all words sense disambiguation. Master's thesis, Department of Computer Science, University of Minnesota, Duluth, Minnesota, July.

Mihalcea, Rada and Dan I. Moldovan. 2001. A highly accurate bootstrapping algorithm for word sense disambiguation. *International Journal on Artificial Intelligence Tools*, 10(1-2):5–21.

Miller, G. A. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Miller, G. A., C. Leacock, T. Randee, and R. Bunker. 1993. A semantic concordance. In *Proceedings of the 3rd DARPA Workshop on Human Language Technology*, pages 303–308.

Mohri, Mehryar, Fernando C. N. Pereira, and Michael Riley. 1997. A rational design for a weighted fi nite-state transducer library. In *Workshop on Implementing Automata*, pages 144–158.

Moldovan, D. and A. Novischi. 2002. Lexical chains for question answering. In *Proceedings of COLING 2002*, pages 674–680.

Morris, J. and G. Hirst. 2005. The subjectivity of lexical cohesion in text. In James C. Chanahan, Yan Qu, and Janyce Wiebe, editors, *Computing attitude and affect in text*. Springer, Dodrecht, The Netherlands.

Morris, Jane and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

Morris, Jane and Graeme Hirst. 2004. Non-classical lexical semantic relations. In *Proceeding of Workshop on Computational Lexical Semantics, Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Boston, May.

Novischi, Adrian and Dan Moldovan. 2006. Question answering with lexical chains propagating verb arguments. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 897–904, Sydney, Australia, July. Association for Computational Linguistics.

Okumura, M. and T. Honda. 1994. Word sense disambiguation and text segmentation based on lexical cohesion. In *Proceedings of the Fifteenth Conference on Computational Linguistics (COLING-94)*, volume 2, pages 755–761.

Patwardhan, S., S. Banerjee, and T. Pedersen. 2003. Using measures of semantic

relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, Mexico City, February.

Resnik, Philip. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, Montreal, August.

Rubenstein, H. and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

Silber, H. Gregory and Kathleen F. McCoy. 2002. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4):487–496.

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

Stokes, Nicola, Joe Carthy, and Alan F. Smeaton. 2004. SeLeCT: A lexical cohesion based news story segmentation system. *Journal of AI Communications*, 17(1):3–12, March.

Zhao, Ying and George Karypis. 2002. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 11th ACM Conference on Information and Knowledge Management (CIKM)*, pages 515–524.