Lexical Chains using Distributional Measures of Concept Distance

Meghana Marathe

A research paper submitted in conformity with
the requirements for the degree of Master of Science

Department of Computer Science
University of Toronto

May 2009

॥ श्री ॥

Lexical Chains using Distributional Measures of Concept Distance

Meghana Marathe

Department of Computer Science, University of Toronto

Toronto, ON, M5S3G4

Abstract

In practice, lexical chains are typically built using term reiteration or resource-based measures of semantic distance. The former approach misses out on a significant portion of the inherent semantic information in a text, while the latter suffers from the limitations of the linguistic resource it depends upon.

In this paper, chains are constructed using the framework of distributional measures of concept distance, which combines the advantages of resource-based and distributional measures of semantic distance. These chains were evaluated on the task of text segmentation and in a study that asked linguistically-trained judges to rate them qualitatively. While performing as well as or better than state-of-the-art methods in the former task, they were rated significantly lower for coherence than chains built using Lin's WordNet-based measure.

Contents

1 Introduction

Lexical chains are sequences of semantically related words in a text (Halliday and Hasan, 1976). A word is added to an existing chain only if it is related to one or more of the words already in the chain by a *cohesive* relation.

In practice, the cohesion between two words is approximated either by term reiteration (Hearst, 1994, 1997) or by the *semantic distance* between them. Methods that restrict lexical cohesion to reiteration consider two terms to be related only if they are instances of the same word. Hence, these methods miss out on a significant portion of the semantic information inherent to a text. Semantic distance is typically computed using one of the following two classes of methods:

Resource-based measures These measures rely on specific features of a linguistic resource to calculate semantic distance. For instance, Morris and Hirst (1991) used thesaural relations whereas Barzilay and Elhadad (1997) used relations between WordNet (Fellbaum, 1998) synonym-sets to estimate semantic distance. While these methods capture a much larger amount of semantic information than term reiteration, their dependence on a specific resource is often problematic. For example, these methods may not be able to operate across parts of speech or in languages where the resource does not exist; or consider non-classical relations.

Measures of distributional similarity These are at the opposite end of the spectrum from resource-based methods, relying solely on co-occurrence information drawn from large corpora to calculate semantic distance. Although these measures are not affected by the limitations of a specific linguistic resource, they run into word sense ambiguity problems because they consider the surface forms of words and not their meanings. Furthermore, their correlation with human judgements is observed to

be fairly low (Weeds, 2003).

This motivates the need for a method that incorporates the advantages of both resource-based and distributional measures of semantic distance. Mohammad and Hirst (2006) proposed *distributional measures of concept distance* that combine distributional co-occurrence information with semantic information from a lexicographic resource, such as a thesaurus. These measures were shown to outperform traditional distributional measures on the tasks of correcting real-word spelling errors, and ranking word pairs in order of semantic distance (Mohammad and Hirst, 2006).

In this work, we build lexical chains using Mohammad and Hirst's framework of distributional measures of concept distance. To test the effectiveness of this method, we evaluate the chains via two experiments: applying them to the task of *text segmentation*; and asking linguistically trained judges to rate them qualitatively.

Text segmentation is the task of dividing a text document into cohesive units or segments by topic (Hollingsworth, 2008). In particular, we focus upon *linear* segmentation, in which segments are not further subdivided; as opposed to *hierarchical* segmentation, where each unit may in turn be divided into sub-units.

Given that most texts are by default subdivided into paragraphs, the reader might wonder why segmentation is required at all. However, it has been observed that real-world text is often subdivided into paragraphs more to achieve a visual layout that aids reading than to indicate a change in the topic(s) under discussion (Stark, 1988). An obvious example is the layout of the columns in many newspapers (Longacre, 1979; Hearst, 1997). Text segmentation finds an application in many important tasks, such as these:

- *Text summarization*: Text segmentation is often the first step in extractive text summarization (Barzilay and Elhadad, 1997). *Extractive* summarization is the

task of constructing a summary by choosing sentences from the text itself.

- *Information retrieval*: Salton et al. (1993) find that comparing a query against sections and then paragraphs is more successful than comparing only against entire documents. Additionally, users may find it more helpful if the relevant paragraph(s) are displayed in the results of their query (Hearst, 1997).

Morris and Hirst (1991) were the first to suggest using lexical chains for text segmentation, which has since become a standard application of lexical chains. Since lexical chains consist of semantically related words, each chain corresponds to a theme or topic (or a set thereof) in the text (Morris and Hirst, 1991). As a result, lexical chains provide three useful cues, namely:

- A significant number of chains beginning at a point in text probably indicates the emergence of some new topic(s).

- A significant number of chains ending at a point in text probably means that certain topics are not discussed henceforth in the text.

- Points where the number of chains beginning or ending is not significant probably represent a continuation in the discussion of some topic(s).

Our hypothesis is that these cues help detect positions at which there are changes or shifts in topic, representing segment boundaries.

Organization The rest of this paper is organized as follows: Section 2 surveys the previous work in lexical chaining, text segmentation and measures of semantic distance. Section 3 describes the algorithms used for building chains and segmenting text. Section 4 details the data, methodology and results of applying lexical chains to text

segmentation. Section 5 presents the study which asked judges to qualitatively evaluate lexical chains. Section 6 summarizes the results from the two preceding sections; enumerates the contributions and limitations of this work; and suggests directions for future work.

2 Background

This section provides a review of previous work in lexical chaining and text segmentation, and provides the motivation for the proposed method.

2.1 Lexical Chains

This section discusses the origins and typical implementations of lexical chains; the concept of chain strength; and commonly used evaluation techniques.

2.1.1 The Foundation

Halliday and Hasan (1976) laid the foundation for lexical chains, when they suggested relating words of a text back to the first word to which they are cohesively "tied". They also specified the following five types of lexical cohesion, based on the dependency relationship between the words:

1. Reiteration with identity of reference

   e.g., Dumbledore clicked the *deluminator*. It was a curious device, his *deluminator*.

2. Reiteration without identity of reference

   e.g., Harry liked his old *wand* better. In his opinion, there existed no finer *wand*.

3. Reiteration by means of superordinate

   e.g., Hagrid showed them the baby *dragon*. He had an inexplicable fondness for dangerous *beasts*.

4. Systematic semantic relation (systematically classifiable)

   e.g., Gryffindors sported *red* for the special occasion. Slytherins, *green*.

5. Nonsystematic semantic relation (not systematically classifiable)

   e.g., Hermione was the brightest *witch* of her age. She always mastered new *spells* before anyone else in her year.

However, they did not consider exploiting the transitivity of these relationships, nor did they discuss computational methods for finding lexical chains.

2.1.2 A Computational Method

Morris and Hirst (1991) were the first to suggest computational means of building lexical chains. They used *Roget's International Thesaurus, 4th Edition* (1977) to find lexical relationships between words.

Roget's has a hierarchical structure organized around 1042 categories. At the top of the hierarchy are eight major classes: *Abstract Relations*, *Space*, *Physics*, *Matter*, *Sensation*, *Intellect*, *Volition*, and *Affections*. Each class is divided into subclasses (e.g., *Organic Matter* under *Matter*), which are further divided into sub-subclasses (e.g., *Vitality* under *Organic Matter*). Sub-subclasses are in turn divided into categories (e.g., *Life* under *Vitality*).

Each category contains a series of paragraphs that group closely related words. A paragraph contains words of only one syntactic category. Within each paragraph there are finer groups that may have pointers to related categories or paragraphs.

Finally, *Roget's* provides an index to facilitate retrieval of words related to a given word. Each index entry corresponds to one word, containing a list of the categories that it belongs to. Each category is represented by a *label* – a representative word – and a *number*.

Morris and Hirst proposed five kinds of lexical relationships, capitalizing on the structure of *Roget's*:

1. Two words have a common category in their index entries.

2. One word has a category in its index entry that contains a pointer to a category of the other word.

3. A word is either a label in the other word's index entry or is contained within a category of the other word.

4. Two words are in the same group (hence semantically related).

5. The two words have categories in their index entries that both point to a common category.

They allowed up to one transitive link while building lexical chains. Thus, if word a is related to word b, word b is related to word c, and word c is related to word d, then word a is related to word c but not word d. They reasoned that two or more transitive links render the relationship non-intuitive.

Their analysis of five texts showed that there should be no more than two or three sentences between a word in a chain and the previous word in the chain to which it can be linked. At larger distances – ranging from four to nineteen sentences – the word signaled a *return* to an existing chain. This phenomenon, called a *chain return*, occurs when a chain is revived several sentences after it has clearly stopped. Morris and Hirst

observed that chain returns consisting of a single word were always a repetition of one of the words in that chain; whereas returns consisting of multiple words did not necessarily use repetition.

Morris and Hirst concluded that lexical chains computed by their algorithm correspond closely to the intentional structure of that text produced from the structural analysis method of Grosz and Sidner (1986). Intentional structure is based on the idea that every discourse has an overall purpose; and that every discourse segment has a purpose, specifying how it contributes to the overall purpose.

Unfortunately, no online copy of the thesaurus was available to Morris and Hirst (see Hirst and St-Onge, 1998), so the algorithm was worked out by hand, preventing extensive tests.

2.1.3 Using WordNet

There have since been several attempts (Hirst and St-Onge, 1998; Stokes et al., 2004; Yang and Powers, 2006) at constructing lexical chains using WordNet (Fellbaum, 1998), a large lexical database for English.

WordNet groups nouns, verbs, adjectives and adverbs into sets of synonyms (called "synsets"), each expressing a distinct concept. Synsets are interlinked through semantic and lexical relations. The structure of WordNet being quite different from that of *Roget's*, researchers proposed various WordNet-centred notions of semantic relatedness instead of those defined by Morris and Hirst.

Hirst and St-Onge (1998), for instance, classifed WordNet synset relations into upward, downward, and horizontal directions. These directions were used to define three kinds of relations – medium-strong, strong and extra-strong. For a given pair of words, the connections between some synset of one word and some synset of the other and the directions of these connections determine how related the words are.

Stokes et al. (2004) proposed the use of lexical chaining as a means of segmenting news stories. They experimented with synonymy, specialization, and part-whole relationships from WordNet; and statistical word association as indicators of lexical cohesion for building chains. Even so, they concluded that optimal performance was achieved when only noun repetition patterns were examined during boundary detection. This seems counterintuitive, and we believe that reliance on WordNet may be the cause.

Yang and Powers (2006) employed WordNet together with the Edinburgh Association Thesaurus (EAT)[1] to build "improved" lexical chains called lexical hubs, for word sense disambiguation (WSD). The EAT consists of an associative network of words, constructed by asking subjects to state the first word they thought of in response to a stimulus word (Kiss et al., 1973). Since WordNet usually restricts itself to paradigmatic relations between words (Fellbaum, 1998), the EAT was used to add associative information. This significantly improved results on the WSD task. However it limits the method's scope to resource-rich languages, requiring not only WordNet but also an associative thesaurus.

These methods suffer from WordNet's fine-grainedness, which has been a typical and frequent criticism of WordNet in the literature. Moreover, it is mainly the noun hierarchy of WordNet that has been extensively developed. Hence these methods cannot exploit the information contained in other parts of speech, such as verbs and adjectives.

2.1.4 Strength of a Chain

Lexical chaining algorithms often produce a much larger number of chains than desired for a particular task (Hollingsworth, 2008). *Chain strength* is used to select the "best" or most relevant chains out of a given set of chains. Morris and Hirst (1991) first

[1] http://www.eat.rl.ac.uk

proposed the concept of chain strength, naming three factors that contribute to it:

1. *Reiteration*: the greater the number of repetitions, the stronger the chain.

2. *Density*: the denser the chain, the stronger it is.

3. *Length* or *size*: the longer/bigger the chain, the stronger it is.

Reiteration is computed by counting the number of word-tokens of each word-type present in the chain. Chain density is the ratio of the number of words in a chain to the number of content words in the text (Hollingsworth, 2008). The length or size of a chain is the number of word-types it contains. Morris and Hirst advocate using a combination of these three factors to compute chain strength.

In practice, chain strength has often been calculated as a weighted sum of the number of occurrences of each word-type in a chain (Barzilay and Elhadad, 1997; Hirst and St-Onge, 1998; Hollingsworth, 2008). The value of a weighting coefficient depends on the kind of lexical relation used to add that term to the chain. It should be noted that this implicitly assumes that the same relation is used to add every occurrence of a word-type to a specific chain.

2.1.5 Evaluation Techniques

Lexical chains are typically evaluated in two main ways:

Quantitative or application-oriented evaluation The chaining method is used as part of some larger application, such as word sense disambiguation (Hirst and St-Onge, 1998) or text segmentation (Stokes et al., 2004). The advantage is that such an evaluation is objective, easily reproducible, and can usually be conducted with a large test set. The main problem is that it is difficult to isolate the influence of the chaining component while interpreting results.

Qualitative or application-independent evaluation Human judges are asked to create (Hollingsworth and Teufel, 2005) or evaluate lexical chains for a given document. The former helps generate gold-standard lexical chains; the latter tells us how "good" or "intuitive" the chains are. This approach suffers from problems arising out of low inter-rater agreement and smaller-sized test sets. Indeed, Hollingsworth and Teufel (2005) showed that it is very difficult for human judges to agree on a gold standard for lexical chains in a given document. However, we will argue that inter-rater agreement is at acceptable levels if the judges are asked only to rate chains (as opposed to creating them).

The general trend so far has been to perform application-oriented evaluation.

2.2 Text Segmentation

This section gives an overview of two well-known methods for segmentation: TextTiling (Hearst, 1993, 1994, 1997) and C99 (Choi, 2000); the Okumura and Honda scoring system for determining segment boundaries using lexical chains; and issues that arise in the evaluation of text segmentation algorithms.

2.2.1 TextTiling

TextTiling (Hearst, 1993, 1994, 1997) is an algorithm for partitioning expository texts into coherent multi-paragraph discourse units that reflect the underlying subtopic structure.

Instead of identifying individual subtopics, TextTiling focuses on detecting *subtopic shifts*. It assumes that a significant change in the vocabulary being employed is indicative of a shift from one subtopic to another. The algorithm proceeds in three phases, described in the following paragraphs.

TextTiling begins by tokenizing the input: The body of the text is extracted from within markup or auxiliary information (if any), tokens that are stop words[2] are stripped off, and the rest are reduced to their roots using a morphological analysis function based on that of Karttunen et al. (1987). The resulting set of tokens is subdivided into pseudo-sentences of a predefined size w, called *token-sequences*.

TextTiling next assigns a lexical score to each token-sequence gap. This score can be determined in one of three ways:

- *Block comparison*: Adjacent blocks of token-sequences are compared for lexical similarity. The *block size*, k, is meant to approximate the average paragraph length. The lexical score for a token-sequence gap is equal to the normalized inner product of two vectors, one for each of the adjacent blocks. Each vector consists of the frequencies of occurrence of all the word-types that appear in the corresponding block. For gap i, the score would be the inner product of vectors representing the block of token-sequences $(i - k)$ to i, and the block of token-sequences $(i + 1)$ to $(i + k + 1)$.

- *Vocabulary introduction*: The score assigned to a token-sequence gap is the sum of the number of new or *not-yet-seen* words in the two token-sequences adjacent to this gap, divided by the total number of tokens in the two token-sequences.

- *Lexical chains*: A modified version of the Morris and Hirst algorithm is used to build chains: term repetition is the sole indicator of lexical cohesion; multiple chains are allowed to span an intention; and chains at all levels of intention are analyzed simultaneously. The score at a token-sequence gap is the number of active chains that span this gap.

[2] A *stop word* is typically a closed-class and/or high-frequency word that is considered to have very low information content.

The final phase – boundary identification – assigns a depth score to each token-sequence gap. For each gap, this score is determined by how sharp a change occurs in the lexical scores assigned to gaps that are encountered as we move away from this gap in both directions. The larger the depth score, the more likely TextTiling is to place a boundary at that location.

Hearst (1997) reports that seven judges were asked to mark paragraph boundaries at which the topic changed for 12 magazine articles, yielding an average kappa score of 0.647. This set of boundaries was used as the gold standard against which three versions of TextTiling (one per scoring method) were evaluated. The block-comparison version was found to perform slightly better than the others.

TextTiling is widely considered a foundational work in paragraph-level text segmentation. Its biggest advantage is that it does not depend on any lexical resource or inference mechanisms. This also means that it can be applied to a variety of natural languages (see Nomoto and Nitta, 1994; Hasnah, 1996). Unfortunately, the algorithm requires setting several interdependent parameters, viz.: the number of words in a token-sequence, w; the number of token-sequences in a block, k; the method, width (s), and number of rounds (n) of smoothing for the lexical scores; and the method to determine the number of boundaries to be placed. The following setting was used by Hearst (1997): $w = 20$, $k = 10$, $n = 1$, $s = 2$.

2.2.2 Okumura and Honda

Okumura and Honda (1994) used a Morris and Hirst style lexical chainer to determine segment boundaries using the following two cues for segmentation:

- When a lexical chain ends, there is a tendency for a segment to end.
- When a new chain begins, it might indicate that a new segment has begun.

Each sentence-gap in the text is assigned a *boundary strength* score equal to the sum of the number of lexical chains that begin and end at this gap. The gaps are then sorted in order of decreasing *boundary strength*, and the first m are chosen as segment boundaries. The number of boundaries, m, is either a fixed value given as input to the algorithm or determined automatically as the number of boundaries with *boundary strength* greater than half the maximum *boundary strength*.

The authors reported preliminary but encouraging results on five Japanese texts. However they did not present any comparison of the performance of their algorithm with that of a baseline or of another algorithm such as TextTiling (Hearst, 1993, 1994, 1997).

2.2.3 C99

C99 (Choi, 2000) is a domain-independent algorithm for linear text segmentation based on two main arguments:

- Since a typical segment has less than 100 tokens, even one additional occurrence of a common word causes disproportionate increase in inter-sentential similarity metrics. Hence such metrics should only be used as relative estimates of similarity, e.g., sentence a is more similar to sentence b than sentence c.

- Since language usage varies over a document, e.g., the introduction is often less cohesive than a section about a specific topic, similarity values from different regions of the document should not be compared directly.

C99 accepts a list of tokenized sentences as input. Thus preprocessing involves passing the text through a tokenizer and a sentence boundary detection algorithm. C99 uses a regular expression-based pattern matcher and a list of stop words to strip

off punctuation and uninformative words. The remaining tokens are reduced to their word stems.

C99 then builds a dictionary of word-stem frequencies for each sentence (in the form of a vector). These vectors are used to compute the cosine similarity $sim(x, y)$ between every pair of sentences x and y, generating a similarity matrix.

Next, each value in the similarity matrix is replaced by its rank in the local region to generate a rank matrix. The rank of a cell is defined as the number of neighbouring elements (in an 11×11 mask centred on that cell) with a lower *sim* value, normalized by the number of elements examined.

A text segment k is defined by two sentences, i and j, represented as a square region along the diagonal of the rank matrix. Segments are identified using divisive clustering based on Reynar's maximization algorithm (Reynar, 1998; Helfman, 1996; Church, 1993; Church and Helfman, 1993): Let s_k be the sum of rank values inside segment k and a_k the area of the square region. Let $B = \{b_1, ..., b_m\}$ denote a list of m coherent text segments. The inside density, D, of B is defined as:

$$D = \frac{\sum_{k=1}^{m} s_k}{\sum_{k=1}^{m} a_k}$$

Starting with the entire document as one segment in B, the algorithm successively splits one segment of B at a time. The point that maximizes D is chosen as the split point, representing a potential boundary. An usually large reduction in the value of D (compared to its value before splitting) suggests that the optimal number of boundaries has been chosen.

C99 outperformed TextTiling (Hearst, 1994), DotPlot (Reynar, 1998) and Segmenter (Kan et al., 1998) on an artificial test corpus of 700 samples (Choi, 2000). Each sample was a concatenation of ten randomly-selected segments of varying lengths

(3 to 11 sentences) from the Brown corpus. However, the use of such concatenated documents as test data has been criticized (see Hearst, 1997), because it often penalizes methods for assigning finer-grained boundaries (e.g., if a strong subtopic change occurring within an artificial 'segment' is marked as a boundary).

2.2.4 Evaluation Metrics

Precision and recall have often been used to evaluate segmentation algorithms (Hearst, 1994; Passonneau and Litman, 1993), but they are considered overly strict: a hypothesized boundary close to a reference boundary is penalized the same as one that is distant from it (Reynar, 1998).

To address this, Passonneau and Litman (1996) proposed allowing fuzzy boundaries when annotators agreed about the existence of a boundary but disagreed about its exact location. Reynar (1994) suggested counting as correct boundaries that appeared within a window of an actual boundary. However, both approaches can be overly generous at times, because they treat inexact and exact matches the same (Reynar, 1998).

Beeferman et al. (1997) proposed the P_k metric, which rewards exact matches more than near misses. They used the probability of two randomly chosen sentences from the text being similarly located in the reference and the hypothesized segmentation to measure performance. Thus, credit is given for cases in which if the two sentences are in the same segment in the reference set, they are also in the same segment in the hypothesized set.

Pevzner and Hearst (2002) demonstrated that the P_k metric penalizes false negatives more heavily than false positives, over-penalizes near-misses, and is significantly affected by variation in the sizes of segments. They proposed a new metric, *WindowDiff*, a modification of P_k. *WindowDiff* moves a fixed-sized window across text and penalizes the algorithm whenever the number of hypothesized boundaries within the window

does not match the number of reference boundaries.

Pevzner and Hearst defined both weighted and unweighted versions of this metric. Weighted *WindowDiff* is defined as follows:

$$WindowDiff(ref, hyp) = \frac{1}{N-k} \sum_{i=1}^{N-k} \left| b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k}) \right| \qquad (1)$$

Here *ref* is the reference segmentation; *hyp* is the proposed segmentation; $b(p, q)$ is the number of boundaries between positions p and q in the text; k is half the average segment length; and N is the total number of sentences in the text. The i in equation 1 is incremented at each sentence-boundary.

On the other hand, unweighted *WindowDiff* assigns a penalty of one whenever the absolute difference between the number of boundaries in the reference and hypothesized segmentations (i.e. the value being summed over in equation 1) exceeds zero.

Both versions of *WindowDiff* assign a score in the range $(0, 1)$ to a hypothesized segmentation. A score of 0 indicates an exact match with the reference segmentation, and a score of 1 indicates that none of the proposed boundaries lie within k sentences of a reference boundary.

Lamprier et al. (2007) argued that *WindowDiff* significantly favours methods that create fewer boundaries, and that *WindowDiff* results from different kinds of corpora are difficult to compare. They proposed a new metric, *NWin*, which normalizes the *WindowDiff* score by the mathematical expectation function of *WindowDiff*.

P_k and *WindowDiff* both are considered standard criteria for evaluating text segmentation methods.

2.3 Measures of Semantic Distance

This section gives a brief overview of the three major classes of methods used to compute semantic distance. For a more complete discussion, please refer to Mohammad and Hirst (2005), and Budanitsky and Hirst (2006).

2.3.1 Resource-based

Resource-based measures are computed using dictionaries, thesauri or wordnets. In a dictionary the semantic distance between two words may, for instance, be defined as the number of common words in the definitions of the two words (Lesk, 1986). In a wordnet it could be defined by the amount of information shared by the nodes corresponding to the two words (Lin, 1998b). In a thesaurus, semantic distance can be defined in terms of the length of the path between the two words through the category structure or index (as in section 2.1.2).

Most of these methods correlate well with human judgements (see Budanitsky and Hirst, 2006), but they have several shortcomings due to their dependence on a specific resource, such as the inability to operate across parts of speech (e.g., the semantic distance between a verb and a noun); or the lack of consideration for non-classical relations (e.g., semantic role relation). This dependence also means that they cannot be applied to languages in which those resources do not exist.

2.3.2 Distributional

Distributional measures treat two words as semantically related if they tend to co-occur with similar contexts. These methods build a distributional profile (DP) per word, consisting of the number of occurrences of that word in various contexts. For example, if the target word is *deluminator* and the corpus contains a sentence such

as *'It was a curious device, his deluminator.'*, the method adds 1 to the count of occurrences of *deluminator* in the context of *curious* and of *device*.

There are several different measures of distributional similarity, such as those proposed by Hindle (1990); Pereira and Tishby (1993); Lin (1998a). They typically differ from each other in their notion of context (e.g., a window of n tokens *vs.* a syntactic argument relationship) and the technique used to incorporate co-occurrence information (e.g., conditional probability *vs.* pointwise mutual information).

These measures can be applied across parts of speech and they can also detect non-classical relationships provided these are reflected in the corpus. However, their correlation with human judgements is observed to be fairly low (Weeds, 2003), and they require extremely large corpora in order to gather sufficient data. In addition, the methods run into problems with word sense ambiguity because they consider only the surface forms of words and not their meanings.

2.3.3 Hybrid

Hybrid methods aim to combine the advantages of resource-based and distributional methods by using both distributional information and a linguistic resource. Three such methods are discussed here, with particular emphasis on the framework proposed by Mohammad and Hirst (2006).

JC Jiang and Conrath (1997) proposed an information-theoretic measure that uses corpus statistics as a corrective factor to the information gained from an IS-A hierarchy such as WordNet (Fellbaum, 1998). The information content (IC) of a node in the hierarchy is a logarithmic function of the probability of occurrence of the corresponding concept or word-sense, estimated using a corpus. Given a pair of words, the semantic distance is estimated by computing how dissimilar, in terms of IC, each

of the words is from their *lowest super-ordinate* (lso). The lso is defined as the lowest node in the hierarchy that subsumes the nodes representing both words. The success of this measure relies largely on the existence of a sufficiently detailed hierarchy for the language under consideration.

SPD Tsang and Stevenson (2004, 2006, to appear) proposed the application of graph-theoretic approaches to a hierarchical ontology (such as WordNet) to determine the semantic distance between a pair of texts. If all the words in a text are mapped to their corresponding concepts in the ontology, the text can be represented as a collection of concepts weighted by word-frequencies, called a *semantic profile*. Semantic distance between two texts is determined by computing the *minimum cost flow* from one semantic profile to the other, representing the effort required to transform one semantic profile to match the other graphically.

DMCD Mohammad and Hirst (2006) proposed the framework of *distributional measures of concept distance*, which combines distributional co-occurrence information with the semantic information from a lexicographic resource. They used the categories from the *Macquarie Thesaurus* (Bernard, 1986) as a set of coarse-grained word senses or concepts, and built a word-category co-occurrence matrix (WCCM) using the sense-annotated *British National Corpus (BNC)*. Cell m_{ij} in the WCCM contained the number of times word i co-occurred (in a window of ± 5 words in the corpus) with any of the words listed under category j in the thesaurus. Distributional profiles of concepts (DPCs) could be derived by applying a suitable statistic, such as odds ratio or pointwise mutual information (PMI), to the WCCM.

Distributional measures of concept distance (DMCDs) are defined as any distributional measures in which DPCs of the categories of the target words are used as

the context, in place of DPs of the words themselves. A DMCD is thus completely defined by choosing the window size (usually ±5 words), the measure of distributional similarity, and the statistic used to measure the strength of association.

DMCDs were evaluated in comparison with distributional and WordNet-based measures on two tasks: ranking word pairs in order of semantic distance with human norms; and correcting real-world spelling errors. DMCDs outperformed distributional measures on both tasks. They did not perform as well as the best WordNet-based measures in ranking word pairs, but in the spelling correction task, DMCDs beat all WordNet-based measures except that of Jiang and Conrath (1997).

3 Method

We aim to investigate whether the use of Mohammad and Hirst's framework of *distributional measures of concept distance* produces "better" lexical chains than those created using traditional resource-based measures; and whether the use of cues provided by these lexical chains results in better performance on the task of text segmentation compared to state-of-the-art segmentation methods.

In this section we describe the general algorithm used for building lexical chains, the two variants of this algorithm that were implemented, and the procedure used for segmenting text using chains generated by these variants.

3.1 A General Algorithm for Lexical Chains

Our lexical chaining algorithm is adapted from the one proposed by Morris and Hirst (1991). It is described as follows:

The algorithm requires the setting of three parameters:

- Indicator of lexical cohesion, I, e.g., a measure of semantic distance.
- Threshold for adding a word to a chain, $threshold_{add}$.
- Threshold for merging two chains, $threshold_{merge}$.

The range of acceptable values for the two thresholds depends upon the range of scores assigned by the method I.

The algorithm expects two inputs:

- Text, in the form of a list of sentences from which punctuation and stop words have been eliminated.
- A method $sim\_ww(x, y)$ that computes the lexical cohesion score between the words x and y according to indicator I.

For each word in the text, the algorithm computes the similarity score between that word and each existing chain (see equation 2). If there are no existing chains, or if the maximum score obtained is lesser than $threshold_{add}$, a new chain containing that word is created.

$$sim\_wc(token, chain) = \underset{word \in chain}{average} \left(sim\_ww(token, word)\right) \tag{2}$$

If there is only one existing chain that obtains the maximum score, the word is added to that chain. If, however, more than one chain obtains the maximum score, these chains become candidates for merging. Similarity scores are computed between each pair of candidate chains (see equation 3). If this score exceeds $threshold_{merge}$, the two chains are merged; else the pair is removed from the candidate pairs. This eventually leads to one surviving candidate, to which the word is added. If no chains are merged, the word is added to the first merge candidate.

$$sim\_cc(chain1, chain2) = \underset{w1 \in chain1, w2 \in chain2}{average} (sim\_ww(w1, w2)) \quad (3)$$

Once all the words in the text have been processed, the algorithm halts, producing a list of lexical chains. Please refer to algorithm 1 for the pseudocode.

Algorithm 1 Building lexical chains

```
list_of_chains = empty
for each word in text do
  max_score = max_{c ∈ list_of_chains} (sim_wc(word, c))
  max_chain = argmax_{c ∈ list_of_chains} (sim_wc(word, c))
  if list_of_chains = empty OR max_score < threshold_add then
    Create new chain c containing word.
    Add c to list_of_chains.
  else if more than one max_chain then
    Merge chains if needed, adding the word to the resultant chain.
  else
    Add word to the chain max_chain.
  end if
end for
return list_of_chains
```

Interpretation of Parameter Values Assuming that the indicator I assigns cohesion scores in the range $(0, 1)$ (where 0 is assigned to semantically distant pairs of words), increasing $threshold_{add}$ beyond 0.8 yields highly conservative chains built mainly using term reiteration, whereas decreasing it below 0.5 yields low-coherence chains where the relationship between words is often not clear. Similarly, a high value of $threshold_{merge}$ leads to very infrequent merging; whereas a low value leads to merging of chains that are not very related to each other.

Chain Strength As noted in section 2.1.4, chain strength calculations commonly make the assumption that the same relation is used to add every occurrence of a word-

type to a specific chain. However, our algorithm uses scores provided by the indicator of lexical cohesion, adding a word to a chain if the average of its cohesion score with each existing word in the chain exceeds a predefined threshold. Thus different occurrences of the same word-type may be added to a chain with different scores. Hence, we eliminate weighting from the calculation of chain strength, effectively reducing it to the length or size of the chain.

3.2 Variants

In our experiments, we use two variants of the general algorithm. Both use $threshold_{add} = 0.8$ and $threshold_{merge} = 0.5$ but differ in their choice of the indicator I:

LexChains-Lin Lin's WordNet-based measure (Lin, 1998b), implemented in the WordNet::Similarity package (Pedersen et al., 2004), is used as the indicator of lexical cohesion. This measure estimates the semantic distance between two words using the amount of information shared by the nodes in WordNet corresponding to these words.

LexChains-Saif Mohammad and Hirst's framework of *distributional measures of concept distance* (Mohammad and Hirst, 2006; Mohammad, 2008) is used as the indicator of lexical cohesion. In particular, we used Lin's measure of distributional similarity (Lin, 1998a) with point-wise mutual information (PMI) as the measure of the strength of association. The Lin-PMI measure was chosen because it consistently performed as well as, if not better than, other distributional measures of concept distance (Mohammad and Hirst, 2006; Mohammad, 2008). Please refer to section 2.3.3 for a more detailed description of Mohammad and Hirst's framework.

3.3 Predicting Boundaries for Text Segmentation

One part of the evaluation consists of applying lexical chains to text segmentation. To choose segment boundaries, we use the scoring system described by Okumura and Honda (1994) coupled with a different way of determining the number of boundaries to predict.

Procedure 2 *predict_boundaries*$(text, \alpha)$

{Identifies segment boundaries based upon lexical chains generated for a document.}

$score = empty$
$segment\_boundaries = empty$
for each gap i in $text$ **do**
  $score_i$ = number of chains beginning at i + number of chains ending at i
end for
$threshold_{seg} = \underset{gap\ i \in text}{average}(score_i) + \alpha$
for each gap i in $text$ **do**
  if $score_i \geq threshold_{seg}$ **then**
    Add i to $segment\_boundaries$.
  end if
end for
return $segment\_boundaries$

After chaining, every gap (between a pair of consecutive sentences) in the text is assigned a score equal to the number of chains beginning and ending at that gap. Boundaries are predicted at gaps whose score exceeds $threshold_{seg}$, computed as a function of the mean gap-score (see procedure 2). The parameter α can either be an absolute value (chosen by tuning it on a development set) or a function of the gap-scores (e.g., variance).

4 Evaluation 1: Text Segmentation

This section describes the data and methodology used and the results obtained in the application-oriented evaluation of the lexical chaining method presented in section 3.

4.1 Data Preparation

Creating gold-standard text-segmentation data based on human judgements is very difficult, because intercoder agreement is fairly low (Hearst, 1997; Passonneau and Litman, 1993). To avoid this problem we used a corpus of research papers, with section- and subsection-boundaries acting as reference segments. Since research papers are written with a view of presenting information in a coherent and structured manner, we believe that the reference segments are a close approximation of gold-standard segments.

The ACL Anthology [3], sponsored by the Association for Computational Linguistics, is the NLP community's research repository. The ACL Anthology Reference Corpus (Bird et al., 2008) is an ongoing effort to provide a standardized reference corpus based on the ACL Anthology. It consists of:

- the source PDF files for articles in the Anthology, as of February 2007;
- raw text for all these articles, extracted automatically from the PDFs using non-OCR based text extraction; and
- metadata for the articles, in the form of BibTeX records.

When we say the text is '*raw*', we mean that there is no mark-up (to delineate headings or sentences) and that extraction-errors (e.g., '...' transcribed as ',Ä¢' in document A00-1006) have not been corrected. We used 20 raw-text documents from

[3] Available at http://www.aclweb.org/anthology/

the ACL ARC corpus, manually marking segment boundaries at the end of each section or subsection larger than 2–3 sentences.

A simple heuristic-based sentence boundary detection algorithm was used to convert the text into a list of sentences, from which punctuation and stop words were then stripped. This list was given as input to the text segmentation method.

4.2 Methodology

Recall that in section 1, we described the hypothesis that the number of lexical chains beginning and ending at a given position in the text helps determine whether it is a potential boundary. In order to test this hypothesis we applied two variants of the lexical chaining method (see section 3.2) to the task of text segmentation and compared their performance with that of JTextTile (Choi, 1999), an improved version of TextTiling (Hearst, 1993, 1994, 1997); and C99 (Choi, 2000), both with default parameter settings. The variants LexChains-Saif and LexChains-Lin both use $\alpha = 3$. Recall that the threshold used to predict segment boundaries depends upon α.

A segment-boundary is defined by the number of the sentence it occurs after. A '*strictly-correct*' boundary is one that occurs at the same sentence-gap as a boundary in the reference segmentation. A '*nearly-correct*' boundary is one that is either strictly correct or occurs one gap before or after a boundary in the reference segmentation.

We evaluate the segmentation proposed by each method using three sets of measures:

Strict precision, strict recall, strict F-score Strict precision is the number of strictly-correct proposed segments divided by the total number of segments in the hypothesized segmentation. Strict recall is the number of strictly-correct proposed segments in the hypothesized segmentation divided by the number of segments in the

Method	Strict			Relaxed			*WindowDiff*	
	Precision	Recall	F-score	Precision	Recall	F-score	Weighted	Unweighted
JTextTile	13.2%	16.4%	14.2%	18.0%	21.9%	19.2%	0.625	0.56
C99	13.0%	14.6%	13.4%	20.4%	23.6%	21.3%	0.595	0.537
LexC-Lin	15.0%	**22.9**%	17.5%	24.7%	**35.8**%	28.3%	0.729	0.515
LexC-Saif	**18.5**%	18.9%	**18.0**%	**29.8**%	31.0%	**29.4**%	**0.577**	**0.463**

Table 1: Precision, recall, f-score, and *WindowDiff* values for JTextTile, C99, LexChains-Lin and LexChains-Saif, averaged over 20 documents.

gold-standard segmentation. Strict F-score is the harmonic mean of strict precision and strict recall. For all three measures, the higher the value, the better.

Relaxed precision, relaxed recall, relaxed F-score These measures are defined the same as their strict counterparts, except for nearly-correct boundaries.

Weighted and unweighted *WindowDiff* This metric (Pevzner and Hearst, 2002) assigns a score in the range $(0, 1)$ to a hypothesized segmentation, where a score of 0 indicates an exact match with the reference segmentation, and a score of 1 indicates that none of the proposed boundaries lie within k sentences of a reference boundary, k being half the average segment length. Please refer to section 2.2.4 for a more detailed discussion.

4.3 Results

The precision, recall, F-score, and *WindowDiff* values for the four methods are reported in Table 1. The best score in each column is rendered in boldface. From the table, it is clear that the two lexical chaining methods, especially LexChains-Saif, outperform the other methods in all metrics.

The difference in the strict and relaxed scores of LexChains-Saif and LexChains-

Lin is statistically insignificant[4]. The strict and relaxed scores for LexChains-Saif differ from those of C99 with a confidence interval of 90% and 98% respectively. Similarly, strict precision, and all relaxed scores for LexChains-Saif differ from those of JTextTile, with a confidence interval of 90% and 99% respectively.

While C99 performs nearly as well as LexChains-Saif on weighted *WindowDiff*, on unweighted *WindowDiff* LexChains-Saif outperforms C99 with a confidence interval of 90%, and JTextTile with an interval of 99%.

5 Evaluation 2: Human Judgement

This section describes a user study (and a preceding pilot study) in which judges were asked to evaluate lexical chains created by the two variants of the algorithm given in section 3.

5.1 Data Preparation

Both studies used documents from the ACL ARC corpus (Bird et al., 2008), preprocessed in the manner described in section 4.1, except that no gold-standard boundaries were marked. The two variants of the lexical chaining algorithm, LexChains-Lin and LexChains-Saif (see section 3.2), were then run on each document.

From the output set of chains, the seven strongest chains were chosen for each study. The strength of a chain was defined as the number of tokens it contained (see sections 2.1.4 and 3.1 for a complete discussion).

Thus each document from the corpus had two sets of lexical chains: one generated by LexChains-Lin, and the other by LexChains-Saif. However, the judges were only

[4]We used the independent Student's t-test and the Wilcoxon signed-rank test to check whether two sets of samples (scores) arise from statistically different populations.

presented with one set (and its corresponding document) at a time, and were asked to rate it by itself. They were informed neither of the method used to generate those chains, nor of the fact that more than one method was used.

In both studies, the judges were given the following resources per document:

- Original: The research paper from the ACL Anthology corresponding to this document.
- In-context chains: The seven strongest lexical chains displayed in-context, i.e., highlighted within the document. Each chain was assigned a separate colour.
- Only-chains: A listing of all the chains generated for this document ordered by the position at which they began in the text. This was provided to enable easy identification of coherence problems (if any) within a chain. Again, the seven strongest chains were highlighted using a separate colour for each.

The judges were also free to refer to dictionaries or thesauri if needed.

5.2 Evaluation Criteria

In both studies, judges were asked to rate the set of the seven strongest lexical chains given for a document according to two criteria:

- Coherence: We believe that a chain is coherent if there exist clear relationships between the words it contains. The judges were encouraged to consider both classical relations such as synonymy, meronymy, hypernymy; and non-classical ones such as semantic role relations (e.g., *choir* and *sing*, or *cut* and *knife*), and high co-occurrence (e.g., *baseball* and *diamond*).
- Coverage: We define a lexical chain to have good coverage if it includes all the words from the document that ought to be in it, given the existing words in the

Document	Criterion	LexChains-Saif				LexChains-Lin			
		J_A	J_B	J_C	Average	J_A	J_B	J_C	Average
Doc_1	Coherence	3.5	3	3	**3.17**	1	4	4	3
	Coverage	2.5	3	3	**2.83**	3	2	3	2.67
Doc_2	Coherence	2.5	4	3	**3.17**	3.5	3	3	**3.17**
	Coverage	2	3	4	**3**	1.5	3	3	2.5
Doc_3	Coherence	2.5	2	3	2.5	3	4	4	**3.67**
	Coverage	1.5	2	4	2.5	1	3	4	**2.67**
Average	Coherence	2.83	3	3	2.95	2.5	3.67	3.67	**3.28**
	Coverage	2	2.67	3.67	**2.78**	1.83	2.67	3.67	2.61

Table 2: Coherence and coverage scores assigned by three judges (J_A, J_B, J_C) to three documents (Doc_1, Doc_2, Doc_3) in the pilot study.

chain. For example, if a chain is of the form {'*noun*', '*verb*', '*noun*', ...}, but another occurrence of the word '*verb*' is omitted, coverage suffers.

While it is hard to measure the quality of a chain, we believe that these two criteria can together provide a good approximation for the same. Thus a chain that shows high coherence and high coverage should most likely be "good" or "intuitive".

5.3 Pilot Study

In the pilot study, three documents from the ACL ARC corpus were chained using LexChains-Lin and LexChains-Saif, to generate six test documents. Three graduate students (from the Computational Linguistics research group of the Department of Computer Science, University of Toronto) were asked to rate the lexical chains of each document for coherence and coverage on a scale of 0 to 4. A score of 0 meant poor coherence/coverage, whereas that of 4 meant excellent coherence/coverage. The judges could also propose another criterion for rating lexical chains, but none of them did so.

The results of the study are presented in Table 2. The best score, averaged over judges, for each document is rendered in boldface. Except for the coherence score as-

signed to lexical chains generated by LexChains-Lin for Doc_1, the kappa coefficient (κ) of inter-rater agreement falls within the range (0.6–0.9), allowing us to draw "tentative" to "replicable" conclusions (Carletta, 1996).

We see that chains generated by LexChains-Saif for Doc_1 and Doc_2 were rated, on average, to have higher coverage than those by LexChains-Lin. On the other hand, chains created by LexChains-Lin for Doc_3 were rated more coherent than those by LexChains-Saif. The judges' preferred method of chaining seems to vary according to the document under consideration.

Averaging scores over documents shows that judge J_A seems to favour chains generated by LexChains-Saif for both coverage and coherence; whereas judges J_B and J_C seem to prefer those generated by LexChains-Lin for coherence. Averaging scores over both judges and documents shows that chains generated by LexChains-Lin are rated considerably more coherent, whereas those generated by LexChains-Saif exhibit slightly higher coverage.

The main feedback received from the judges was that asking them to give a single value (e.g., coherence score) to represent the coherence of lexical chains for an entire document was too vague or coarse-grained a task.

5.4 Main Study

In this study, six more documents from the ACL ARC corpus were chained using LexChains-Lin and LexChains-Saif, to produce twelve test documents. To make the task less vague, the judges (three other graduate students, also from the Computational Linguistics research group) were asked to rate the test documents on two levels:

- Word-level: Given the seven strongest chains, the judges were asked to circle words that should not belong to a particular chain (indicating a problem in

Document	LexChains-Saif				LexChains-Lin			
	J_D	J_E	J_F	Average	J_D	J_E	J_F	Average
Doc_4	53 (16)	80 (47)	81 (42)	71 (34)	0 (0)	30 (10)	9 (8)	13 (6)
Doc_5	46 (29)	14 (12)	48 (37)	36 (26)	3 (3)	9 (6)	8 (4)	7 (4)
Doc_6	19 (11)	40 (25)	59 (32)	39 (23)	8 (5)	31 (13)	31 (13)	23 (10)
Doc_7	15 (9)	50 (26)	39 (25)	35 (20)	1 (1)	0 (0)	18 (4)	6 (2)
Doc_8	9 (9)	150 (67)	125 (56)	95 (44)	23 (13)	43 (19)	76 (28)	47 (20)
Doc_9	42 (30)	14 (10)	28 (27)	28 (22)	1 (1)	1 (1)	12 (8)	5 (3)
Average	31 (17)	58 (31)	63 (37)	51 (28)	6(4)	19 (8)	26 (11)	17 (8)

Table 3: Number of word-tokens circled to indicate a lack of coherence by three judges (J_D, J_E, J_F) in six documents (Doc_4, Doc_5, Doc_6, Doc_7, Doc_8, Doc_9) as part of the main study. The numbers in parentheses stand for the number of distinct word-types within the circled tokens.

coherence). They were also asked to circle words that ought to be in one of the strongest chains but weren't (indicating a problem in coverage). To distinguish between the two kinds of annotation, the judges were provided with markers of different colours.

- Overall: As in the pilot study, the judges were asked to rate the lexical chains for coherence and coverage on a scale of 0 to 7. A score of 0 meant very poor coherence/coverage, whereas that of 7 meant excellent coherence/coverage. We increased the range of valid scores (from (0–4) in the pilot) so that the judges could assign finer-grained ratings.

The results of the main study are presented in Tables (3 – 9).

Table 3 shows the number of words (tokens) circled by judges J_D, J_E, and J_F per test document to indicate coherence-based problems. The numbers in parentheses give the number of word-types corresponding to the circled words. We see that all three judges marked significantly fewer (with a confidence interval[5] of 98%) words in

[5]We used the independent Student's t-test and the Wilcoxon signed-rank test to check whether two sets of samples arise from statistically different populations.

Doc.	LexChains-Saif				LexChains-Lin			
	$J_D \cap J_E$	$J_E \cap J_F$	$J_F \cap J_D$	All	$J_D \cap J_E$	$J_E \cap J_F$	$J_F \cap J_D$	All
Doc_4	25 (13)	62 (34)	21 (12)	21 (12)	0 (0)	4 (4)	0 (0)	0 (0)
Doc_5	11 (9)	14 (12)	24 (21)	11 (9)	2 (2)	6 (3)	1 (1)	0 (0)
Doc_6	19 (11)	31 (20)	19 (11)	19 (11)	7 (4)	27 (10)	6 (4)	5 (3)
Doc_7	5 (5)	15 (10)	10 (7)	5 (3)	0 (0)	0 (0)	0 (0)	0 (0)
Doc_8	8 (8)	102 (45)	9 (9)	8 (8)	7 (7)	41 (18)	21 (12)	6 (6)
Doc_9	5 (5)	6 (6)	17 (16)	4 (4)	0 (0)	0 (0)	1 (1)	0 (0)
Average	12 (8)	38 (21)	17 (13)	11 (8)	3(2)	13 (6)	5 (3)	2 (2)

Table 4: Overlap amongst the three judges (J_D, J_E, J_F) in the number of words circled to indicate a lack of coherence in six documents (Doc_4, Doc_5, Doc_6, Doc_7, Doc_8, Doc_9) as part of the main study. The numbers in parentheses stand for the number of distinct word-types within the overlapping words.

documents chained by LexChains-Lin compared to those chained by LexChains-Saif, except for judge J_D in document Doc_8.

It might be argued that since each judge can have a different threshold for acceptable coherence, simply averaging the number of words circled does not provide sufficient information. Table 4 shows both pairwise and overall overlap in the three judges' coherence annotations in terms of word-tokens and word-types. Even if we consider only those words that all three judges agree upon, considerably fewer words were marked in documents chained by LexChains-Lin compared to those chained by LexChains-Saif. In addition, annotations by judges J_E and J_F show the highest overlap on average, meaning that they agree the most with each other on coherence-related problems.

Since each judge may have a different threshold for acceptable coherence, we also compute a *voted agreement score* per test document, defined as follows:

$$score_{agreement} = \prod_{w \in \ circled \ wordtypes} \frac{(number \ of \ votes \ for \ w)}{3} \tag{4}$$

Document	LexChains-Saif	LexChains-Lin
Doc_4	$2.56e^{-14}$	$\mathbf{3.35e^{-06}}$
Doc_5	$2.18e^{-13}$	**0.02926**
Doc_6	$2.01e^{-10}$	**0.00032**
Doc_7	$1.69e^{-16}$	**0.00411**
Doc_8	$1.10e^{-22}$	$\mathbf{5.57e^{-06}}$
Doc_9	$2.70e^{-15}$	**0.00010**
Average	$3.36e^{-11}$	**0.00564**

Table 5: $Score_{agreement}$ values computed for coherence annotations of six documents (Doc_4, Doc_5, Doc_6, Doc_7, Doc_8, Doc_9) by three judges (J_D, J_E, J_F) in the main study. The higher the score, the fewer the number of words circled as coherence problems, and the higher the agreement amongst judges.

Thus, every word-type is assigned a score equal to the number of judges that circled at least one instance of that word-type, i.e. the number of votes it received, divided by the total number of judges. $score_{agreement}$ for a document is then computed as the product of the scores of all word-types that have been marked at least once in that document. The value of this metric falls in the range (0–1). A value approaching 0 indicates very poor inter-rater agreement along with a large number of circled word-types; whereas a value approaching 1 indicates very high inter-rater agreement and very few circled word-types.

Table 5 displays $score_{agreement}$ values computed for coherence annotations of the six test documents by the three judges. The highest score for each document is rendered in boldface. We see that documents chained by LexChains-Lin receive consistently higher scores compared to the corresponding documents chained by LexChains-Saif. This is consistent with the data in Tables 3 and 4.

Table 6 displays the number of words circled by judges J_D, J_E, and J_F per test document to indicate coverage problems. Again, all three judges marked, on average, significantly fewer (with a confidence interval of 97%) words in documents chained

Document	LexChains-Saif				LexChains-Lin			
	J_D	J_E	J_F	Average	J_D	J_E	J_F	Average
Doc_4	42	20	192	85	34	11	72	39
Doc_5	16	28	36	27	33	28	83	48
Doc_6	56	49	72	59	25	21	67	38
Doc_7	31	5	126	54	3	2	50	18
Doc_8	75	92	114	94	1	19	40	20
Doc_9	60	130	46	79	47	38	72	52
Average	47	54	98	66	24	20	64	36

Table 6: Number of words circled to indicate a lack of coverage by three judges (J_D, J_E, J_F) in six documents (Doc_4, Doc_5, Doc_6, Doc_7, Doc_8, Doc_9) as part of the main study.

Doc.	LexChains-Saif				LexChains-Lin			
	$J_D \cap J_E$	$J_E \cap J_F$	$J_F \cap J_D$	All	$J_D \cap J_E$	$J_E \cap J_F$	$J_F \cap J_D$	All
Doc_4	4	13	34	3	6	9	22	6
Doc_5	6	9	8	3	13	22	24	13
Doc_6	12	21	28	12	11	13	13	9
Doc_7	0	5	24	0	0	1	1	0
Doc_8	42	32	20	15	0	11	1	0
Doc_9	32	27	16	10	25	33	30	25
Average	16	18	22	7	9	15	15	9

Table 7: Overlap amongst the three judges three judges (J_D, J_E, J_F) in the number of words circled to indicate a lack of coverage in six documents (Doc_4, Doc_5, Doc_6, Doc_7, Doc_8, Doc_9) as part of the main study.

Document	LexChains-Saif	LexChains-Lin
Doc_4	$2.08e^{-18}$	$\mathbf{3.06e^{-09}}$
Doc_5	$\mathbf{4.15e^{-13}}$	$8.53e^{-16}$
Doc_6	$3.12e^{-18}$	$\mathbf{4.27e^{-16}}$
Doc_7	$2.88e^{-15}$	$\mathbf{1.42e^{-11}}$
Doc_8	$1.96e^{-24}$	$\mathbf{1.68e^{-11}}$
Doc_9	$2.26e^{-26}$	$\mathbf{1.15e^{-14}}$
$Average$	$6.96e^{-14}$	$\mathbf{5.15e^{-10}}$

Table 8: $Score_{agreement}$ values computed for coverage annotations of six documents (Doc_4, Doc_5, Doc_6, Doc_7, Doc_8, Doc_9) by three judges (J_D, J_E, J_F) in the main study. The higher the score, the fewer the number of words circled as coverage problems, and the higher the agreement amongst judges.

by LexChains-Lin compared to those chained by LexChains-Saif, except in document Doc_5. Table 7 shows pairwise and overall overlap in the three judges' coverage annotations in terms of word-tokens. If we consider words that any two judges agree upon, fewer words were marked in documents chained by LexChains-Lin than those chained by LexChains-Saif, as expected. Surprisingly, however, if we consider only those words that all three judges agree upon, we see that LexChains-Saif has fewer or equal words marked overall and in documents Doc_4, Doc_5, Doc_7 and Doc_9 compared to LexChains-Lin. Annotations by judges J_D and J_F show the highest overlap on average, indicating that they agree the most with each other on coverage-related problems.

As with coherence, Table 8 displays $score_{agreement}$ values (see equation 4) computed for coverage annotations of the six test documents by the three judges. The highest score for each document is rendered in boldface. We see that except for Doc_5, all documents chained by LexChains-Lin receive consistently higher scores than the corresponding documents chained by LexChains-Saif. This is consistent with the data in Table 3, but it seems to differ from the overlap data reported in Table 4 on documents Doc_4, Doc_7, and Doc_9. However, this is not an anomaly because the $score_{agreement}$

Document	Criterion	LexChains-Saif				LexChains-Lin			
		J_D	J_E	J_F	Average	J_D	J_E	J_F	Average
Doc_4	Coherence	1	1	1	1	6	5	6	**5.67**
	Coverage	1	6	1	2.67	6	7	4	**5.67**
Doc_5	Coherence	3	4	3	3.33	6	6	5	**5.67**
	Coverage	5	5	5	**5**	5	5	2	4
Doc_6	Coherence	5	3	3	3.67	6	4	3	**4.33**
	Coverage	4	5	4	**4.33**	4	6	2	4
Doc_7	Coherence	5	2	3	3.33	7	6	5	**6**
	Coverage	6	7	3	**5.33**	6	7	2	5
Doc_8	Coherence	5	1	3	3	3	5	4	**4**
	Coverage	4	2	3	3	6	5	5	**5.33**
Doc_9	Coherence	3	5	3	3.67	6	6	5	**5.67**
	Coverage	3	3	4	**3.33**	3	5	2	**3.33**
Average	Coherence	3.67	2.67	2.67	3	5.67	5.33	4.67	**5.22**
	Coverage	3.83	4.67	3.33	3.95	5	5.83	2.83	**4.56**

Table 9: Coherence and coverage scores assigned by three judges (J_D, J_E, J_F) to six documents (Doc_4, Doc_5, Doc_6, Doc_7, Doc_8, Doc_9) in the main study.

metric penalizes methods for a high number of circled words in addition to low agreement.

Table 9 shows the coherence and coverage scores assigned by judges J_D, J_E, and J_F to the test documents. The best score, averaged over judges, for each document is rendered in boldface. Except for coverage scores assigned to LexChains-Saif for Doc_4 and to LexChains-Lin for Doc_5, the kappa coefficient (κ) of inter-rater agreement falls within the range (0.6–1.0), allowing us to draw "tentative" to "replicable" conclusions (Carletta, 1996). We see that LexChains-Lin was assigned higher coherence scores than LexChains-Saif for all documents by all three judges. On the other hand, LexChains-Saif was assigned better or equal coverage scores on four of the six documents (Doc_5, Doc_6, Doc_7 and Doc_9). Surprisingly, averaging scores over documents shows that only judge J_F rated LexChains-Saif higher overall in terms of coverage.

We expected coverage and coherence scores to correlate inversely with the number of circled words. This was observed to be true for a majority of the cases. Interestingly, however, judge J_D assigned equal coverage scores to LexChains-Saif and LexChains-Lin for documents Doc_5, Doc_6 and Doc_7 in spite of circling, in LexChains-Saif, at least twice the number of words circled in LexChains-Lin. Similarly, judge J_F assigned a higher coverage score to LexChains-Saif for document Doc_7 in spite of circling more than twice the number of words circled in LexChains-Lin. In fact, we see that coverage and coherence scores averaged over judges show a better inverse-correlation with overlaps in the annotations of all three judges.

6 Conclusion

This section presents a summary of the results of the two experiments described in sections 4 and 5; the contributions and limitations of this work; and potential directions for future work.

6.1 Summary of Results

Text Segmentation Both variants of the lexical chaining method described in section 3 significantly outperform JTextTile (Choi, 1999), an improved version of TextTiling (Hearst, 1993, 1994, 1997). They also outperform or perform as well as C99 (Choi, 2000), a popular domain-independent text-segmentation algorithm. Of the two variants, LexChains-Saif performs better overall.

Human Judgement In both studies, lexical chains created by LexChains-Lin were unanimously rated significantly more coherent than those generated by LexChains-Saif. While chains created by LexChains-Saif were rated higher on average for coverage in the pilot study and in some documents in the main study, those by LexChains-Lin received higher coverage scores on average in the main study. The kappa coefficient (κ) of inter-rater agreement is reasonably high on the task of evaluating lexical chains. It lies in the range (0.6–1.0), with an average value of 0.7.

6.2 Contributions

Text Segmentation using Lexical Chains We implemented two variants of the general lexical chaining algorithm; LexChains-Lin uses Lin's WordNet-based measure (Lin, 1998b) and LexChains-Saif uses Lin's measure of distributional similarity (Lin, 1998a) computed under the framework of *distributional measures of concept distance*

(Mohammad and Hirst, 2006; Mohammad, 2008). When applied to the task of text segmentation, both significantly outperform TextTiling (Hearst, 1993, 1994, 1997), and perform as well as, if not better than, C99 (Choi, 2000).

Qualitative Evaluation of Lexical Chains Previous studies focused on the task of creating gold-standard lexical chains or comparing automatically-generated chains with such gold-standard chains, both of which resulted in very low inter-rater agreement. In contrast, this work explores the task of rating a stand-alone set of (possibly automatically-generated) lexical chains on some fixed criteria. The inter-rater agreement is observed to be sufficient to draw 'tentative' to 'replicable' conclusions.

6.3 Limitations and Future Work

Effects of Genre The ACL ARC corpus (Bird et al., 2008) represents the very constrained genre of research papers in the area of Computational Linguistics. It would be interesting to analyze the performance of different measures of semantic distance on a variety of genres; and to investigate the effect(s) of document genre on the qualitative evaluation task.

Interesting Chains Lexical chaining algorithms typically generate a much larger number of chains than can be processed comfortably by humans. Hence, there arises the need to select a small set of the "best" chains, which is then presented to the judges or annotators. "Best" is traditionally approximated to strongest, but it might be useful to select the most "interesting" chains instead. An interesting chain could, for instance, relate words using reasonable but unexpected or non-obvious relations.

Parameters In this work, $threshold_{add}$, $threshold_{merge}$ and α, the parameters of the lexical chaining algorithm, were tuned using a small development set. This in itself was difficult because the parameters are interrelated, making it hard to isolate their effects. It would be worthwhile exploring ways to determine their values automatically per set of documents or per genre.

User Interface In the main study, judges reported that it was hard to detect coverage problems because everything was presented in paper form. It would be helpful to build a GUI that allows easy navigation between the in-context and only-chains views.

References

Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization (ISTS'97)*, pages 10–17, Madrid, 1997.

Doug Beeferman, Adam Berger, and John Lafferty. Text segmentation using exponential models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 35–46, 1997.

J.R.L. Bernard, editor. *The Macquarie thesaurus*. Macquarie Library, Sydney, Australia, 1986.

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics. In *Proceedings of Language Resources and Evaluation Conference (LREC 08)*, Marrakesh, Morocco, May 2008.

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

Jean Carletta. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.

Freddy Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 26–33, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

Freddy Y. Y. Choi. JTextTile: A free platform independent text segmentation algorithm. Software, 1999. http://www.cs.man.ac.uk/~choif.

Kenneth W. Church. Char align: A Program for Aligning Parallel Texts at the Character Level. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Columbus, Ohio, USA, June 1993. Association for Computational Linguistics.

Kenneth W. Church and Jonathan I. Helfman. Dotplot: A Program for Exploring Self-Similarity in Millions of Lines of Text and Code. *Journal of Computational and Graphical Statistics*, 2(2):153–174, 1993.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication Series. MIT Press, Cambridge, 1998.

Barbara J. Grosz and Candace L. Sidner. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, 1986.

M. A. K. Halliday and Ruqaiya Hasan. *Cohesion in English*. Longman, London, 1976.

Ahmad Hasnah. *Full Text Processing and Retrieval: Weight Ranking, Text Structuring, and Passage Retrieval for Arabic Documents*. PhD thesis, Illinois Institute of Technology, 1996.

Marti A. Hearst. TextTiling: A quantitative approach to discourse segmentation. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1993.

Marti A. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, USA, June 1994. Association for Computational Linguistics.

Marti A. Hearst. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, 1997. ISSN 0891-2017.

Jonathan Helfman. Dotplot patterns: a literal look at pattern languages. *Theory and Practice of Object Systems*, 2(1):31–41, 1996. ISSN 1074-3227.

Donald Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 268–275, Pittsburgh, Pennsylvania, USA, June 1990. Association for Computational Linguistics.

Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database*, pages 305–332. The MIT Press, Cambridge, MA, 1998.

William Hollingsworth and Simone Teufel. Human Annotation of Lexical Chains: Coverage and Agreement Measures. In *Workshop proceedings "ELECTRA:Methodologies and Evaluation of Lexical Cohesion Techniques in Real-world Applications", SIGIR 2005*, Salvador, Brazil, 2005.

William A. Hollingsworth. *Using Lexical Chains to Characterise Scientific Text*. PhD thesis, Clare Hall College, University of Cambridge, 2008.

Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research on Computational Linguistics (ROCLING X)*, Taiwan, 1997.

Min-Yen Kan, Judith L. Klavans, and Kathleen R. McKeown. Linear segmentation and segment significance. In *Proceedings of the 6th International Workshop of Very Large Corpora (WVLC-6)*, pages 197–205, Montreal, Quebec, Canada, August 1998.

Lauri Karttunen, Kimmo Koskenniemi, and Ronald M. Kaplan. A compiler for two-level phonological rules. In Mary Dalrymple, editor, *Tools for Morphological Analysis*. Center for the Study of Language and Information, Stanford, CA, 1987.

G. R. Kiss, C. Armstrong, R. Milroy, and J. Piper. An associative thesaurus of English and its computer analysis. In A. J. Aitken, R. W. Bailey, and N. Hamilton-Smith, editors, *The Computer and Literary Studies*. Edinburgh: University Press, 1973.

Sylvain Lamprier, Tassadit Amghar, Bernard Levrat, and Frederic Saubion. On Evaluation Methodologies for Text Segmentation Algorithms. In *ICTAI '07: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, pages 19–26, Washington, DC, USA, 2007. IEEE Computer Society.

Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, New York, NY, USA, 1986. ACM.

Dekang Lin. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, Montreal, Quebec, Canada, August 1998a. Association for Computational Linguistics.

Dekang Lin. An Information-Theoretic Definition of Similarity. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA, 1998b. Morgan Kaufmann Publishers Inc.

R. E. Longacre. The paragraph as a grammatical unit. In Talmy Givón, editor, *Syntax and Semantics: Discourse and Syntax*, volume 12, pages 115–134. Academic Press, New York, 1979.

Saif Mohammad. *Measuring Semantic Distance using Distributional Profiles of Concepts*. PhD thesis, Department of Computer Science, University of Toronto, February 2008.

Saif Mohammad and Graeme Hirst. Distributional measures as proxies for semantic relatedness, 2005. Available at http://ftp.cs.toronto.edu/pub/gh/Mohammad+Hirst-2005.pdf.

Saif Mohammad and Graeme Hirst. Distributional measures of concept-distance: A task-oriented evaluation. In *Proceedings, 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, Sydney, Australia, July 2006.

Jane Morris and Graeme Hirst. Lexical cohesion, the thesaurus, and the structure of text. *Computational linguistics*, 17(1):21–48, March 1991.

Tadashi Nomoto and Yoshihiko Nitta. A Grammatico-Statistical Approach to Discourse Partitioning. In *Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING)*, pages 1145–1150, 1994.

Manabu Okumura and Takeo Honda. Word sense disambiguation and text segmentation based on lexical cohesion. In *COLING 1994 Volume 2: The 15th International Conference on Computational linguistics*, pages 755–761, Kyoto, Japan, 1994.

Rebecca J. Passonneau and Diane J. Litman. Intention-based Segmentation: Human Reliability and Correlation with Linguistic Cues. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 148–155, Columbus, Ohio, USA, June 1993. Association for Computational Linguistics.

Rebecca J. Passonneau and Diane J. Litman. Empirical analysis of three dimensions of spoken discourse: Segmentation, coherence and linguistic devices. In E. H. Hovy and D. R. Scott, editors, *Computational and Conversational Discourse: Burning Issues – An Interdisciplinary Account*, chapter 7, pages 161–194. Springer Verlag, Berlin, 1996.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet::Similarity – Measuring the Relatedness of Concepts. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Demonstration Papers*, pages 38–41, Boston, Massachusetts, USA, May 2004. Association for Computational Linguistics.

Fernando Pereira and Naftali Tishby. Distributional Clustering of English Words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio, USA, June 1993. Association for Computational Linguistics.

Lev Pevzner and Marti Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28:1–19, 2002.

Jeffrey C. Reynar. An automatic method of finding topic boundaries. In *Proceedings of the 32nd annual meeting of the Association for Computational Linguistics, Student Session*, pages 331–333, Las Cruces, New Mexico, 1994. Association for Computational Linguistics.

Jeffrey C. Reynar. *Topic segmentation: Algorithms and applications*. PhD thesis, Computer and Information Science, University of Pennsylvania, 1998.

P. Roget. *Roget's International Thesaurus, Fourth Edition*. Harper and Row Publishers Inc., 1977.

Gerard Salton, J. Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *SIGIR '93: Proceedings of the 16th Annual International*

ACM/SIGIR Conference on Research and Development in Information Retrieval, pages 49–58, Pittsburgh, PA, USA, 1993. ACM.

Heather Stark. What do paragraph markers do? *Discourse Processes*, 11(3):275–304, 1988.

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

Vivian Tsang and Suzanne Stevenson. Calculating Semantic Distance between Word Sense Probability Distributions. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of The Eighth Conference on Computational Natural Language Learning (CoNLL-2004)* , pages 81–88, Boston, MA, May 2004. Association for Computational Linguistics.

Vivian Tsang and Suzanne Stevenson. Context Comparison as a Minimum Cost Flow Problem. In *Proceedings of TextGraphs: The Second Workshop on Graph Based Methods for Natural Language Processing*, pages 97–104, New York, NY, June 2006. Association for Computational Linguistics.

Vivian Tsang and Suzanne Stevenson. A Graph-Theoretic Framework for Semantic Distance. *Computational Linguistics*, to appear.

J. E. Weeds. *Measures and applications of lexical distributional similarity*. PhD thesis, University of Sussex, September 2003.

Dongqiang Yang and David M. W. Powers. Word Sense Disambiguation Using Lexical Cohesion in the Context. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 929–936, Sydney, Australia, July 2006. Association for Computational Linguistics.