Topic Segmentation of Meetings Using Lexical Chains

Francisco Adarve
Helen Kwong
Mike Speriosu

CS224N
Spring 2007
Final Project

Abstract

Topic segmentation attempts to divide a document into segments, where each segment corresponds to a particular discourse topic. Lexical chains are a disambiguation tool often used for text summarization, and more recently in topic segmentation. A lexical chain encapsulates the concept of a single word (or group of closely related words) that occurs repeatedly across some portion of a document. While it might be uninteresting to attempt topic segmentation on news articles, which often revolve around just a single topic for the duration of the article, meeting conversations typically move across several topics and are more interesting to segment by topic. Some work has been done using lexical chains to perform topic segmentation on transcribed meeting corpora (Galley et al., 2003), but this work used a very simple implementation of lexical chains that only counted identical words as belonging to one lexical chain. We present here an implementation of topic segmentation on meeting text using more advanced lexical chains, utilizing synonymy and other relationships between distinct words.

Introduction

The natural language task known as topic segmentation (or discourse segmentation) involves deciding where transitions from one topic to another are within a document. In some genres, such as news articles, a single document often discusses only a single topic. One way to apply topic segmentation to this genre is to concatenate many articles on different subjects into one large transcript, and then attempt to determine the boundaries between the articles. However, another interesting genre to apply topic segmentation to is that of conversation or meeting text. In these types of text, speakers typically discuss a variety of different topics as time progresses.

There are difficulties that arise from working with transcribed spoken text. For instance, a large fraction of spoken language is ungrammatical due to the spontaneous manner in which it is generated. Words may be unnecessarily repeated, sentences restarted or abandoned altogether, or speakers interrupted mid-utterance. However, the usefulness of natural language applications capable of processing spoken text is obvious, and we decided to attempt topic segmentation on a transcribed meeting corpus.

One method that is often used in text summarization, a related but different problem, is known as lexical chains. A lexical chain represents the repeated occurrence of a single word or of several closely related words over the course of some fraction of a document. Barzilay and Elhadad (1997) showed an implementation of lexical chains as applied to summarization.

Galley et al. (2003) presented a lexical chain approach to topic segmentation of the ICSI meeting corpus. In their experiment, they used a very simple conception of lexical chains where each item in a chain is identical to all the other items. In order to generalize somewhat, however, they performed stemming on each token in the transcript as a preprocessing step. In this way, “rat” and “rats” would belong to the same lexical chain, but “mouse” would not.

Following our intuition that lexical chains should encapsulate an entire concept, rather than just a particular word that can be used for that concept, we decided to use more advanced lexical chains such as those implemented by Silber and McCoy (2002). In this version, synonymy, hypernymy, hyponymy, and other relations are used to group words into a lexical chain. We used WordNet (Miller et al.), a well known lexical information database, to retrieve these relationships.

Since lexical chains are intended to contain concepts or ideas, we only formed lexical chains out of nouns for many of our experiments. Verbs, prepositions, and other such parts of speech tend to connect concepts together in various ways rather than introduce new concepts. We used a part-of-speech tagger to determine which words were nouns. As we will discuss later, we also used adjectives in some experiments.

In the following sections, we will first discuss preprocessing of the corpus that we performed. Next, we will give a detailed explanation of the two different types of advanced lexical chains we implemented. Then we will explain the methods we used to determine topic boundaries given lexical chains. Finally, we will present our results according to two common evaluation measures, followed by concluding remarks.

Data Preprocessing

For this paper we used the same 25 meetings from the ICSI meeting corpus as Galley et al. (2003). The meetings are transcribed as speaker turns (i.e. one instance of one person speaking without significant pause) rather than sentences. For some of our experiments, we wanted to work in terms of sentences, so a separate copy of the corpus broken into sentences instead had to be made. We used the simple heuristic that periods (.), question marks (?), and exclamation marks (!) usually serve as sentence boundaries. A common problem with this method is that abbreviations (e.g. “O.K.”) tend to have periods in them, but the data was preprocessed to handle these cases (e.g. “O_K”), which helped with sentence boundary determination.

Since we used only nouns (and adjectives in some experiments) in our lexical chains, we needed to run a part-of-speech tagger on the data. We used the Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al., 2003). It did not make a significant difference to the output of the tagger whether we ran it on the speaker turns or sentences.

To aid in successfully looking up words in WordNet, we used the first step of Porter Stemmer, the one that gets rid of plurals. We only performed stemming on nouns, as the most important suffix we wanted to remove was pluralization. When attempting a lookup in WordNet, we first queried the stemmed form of the noun. If that stemmed word was found in WordNet, we used the information from WordNet for it. If it wasn’t found, we tried querying the unstemmed form. If the unstemmed form wasn’t found as a noun, we queried it as an adjective. If it still wasn’t found, we discarded that particular word. In summary, WordNet helps us with the stemming of words. Looking at the stemmed word in WordNet is not satisfactory, since some roots can adopt many meanings.

We thought it might help make more accurate lexical chains if we performed anaphora resolution (AR) on the transcriptions before they were read in by the chain builder. Pronouns could be replaced by their antecedent, and algorithms for building lexical chains would be better able to track the discussion of a single noun in the case where it is referred to by a pronoun after first mentioned. We attempted a simple approach to AR where each of the pronouns “it”, “they”, “he”, and “she” (along with these words in various other grammatical cases) was replaced by the most recent antecedent in the text that could be its correct antecedent. “It” looked for the most recent singular non-proper noun, “s/he” for the most recent singular proper noun, and “they” for the most recent plural noun of any kind. This follows the intuition that a pronoun should refer to the most recent noun that makes sense as its antecedent. Unfortunately, “makes sense” turns out to mean something quite different from “is most recent” in a very large portion of cases. For instance, our simple AR algorithm turned phrases like “put your name on it” to “put your name on name”. With no gender information encoded by the POS tagger, it turned text like “Fey arrived today. Jerry wants to work with her” to “Fey arrived today. Jerry wants to work with Jerry.” It did especially poorly with instances of “it” that have no real linguistic antecedent, such as in exchanges like, “‘I just moved in today, so I borrowed your ladder.’ ‘Oh, it’s nice that you found a new place’”, which gets turned into “‘Oh, ladder’s nice that you found a new place.’” Clearly, this method of AR is too simplistic to provide any real help, and we did not end up running this preprocessing step for our final results. More advanced methods of AR that consider the full syntactic parse of sentences would be required.

Another dataset we used was the “Wall Street Journal” (WSJ) articles corresponding to the year 1994. We concatenated random selected articles, between 4 and 21 of them, in order to see if our segmenter could detect the change of topic, i.e. points of concatenation.

Computing Lexical Chains

A lexical chain is a set of semantically related words inside a document. In some sense, lexical chains capture the cohesion that a person consciously or unconsciously follows when she writes or talks. As an example, if we are interested in felines, we can start talking about cats, and later elaborate the discourse to talk about wild cats, such as leopards, tigers, etc. All these nouns will be chained together into a lexical chain that represents the concept “felines”. However, and more likely, if we are interested in Apple operating systems, we will probably mention tiger and leopard in a very different context. The method to compute lexical chains performs “Word Sense Disambiguation” and will group tiger and leopard with the Apple chain, and not with the “felines” chain.

Lexical chains are a tool frequently used in natural language processing, especially in the text summarization method proposed by Barzilay and Elhadad (1997) and the topic segmentation method in Galley et al. (2003). Interestingly, recent publications propose lexical chains for annotation of images, which allows searching over images, extracting relationships between images, etc. If we compute the lexical chains in the text surrounding the images, we can attach to the image the concept represented by the most important (longest) lexical chains.

As mentioned before, we intended to modify the lexical chain segmenter of Galley et al. (2003) by adding more powerful lexical chains. Galley’s original implementation of the segmenter uses lexical chains defined as repetitions of the same word. For example, all appearances of the word “mole” in the document will appear under a lexical chain with meaning “mole”, whatever the meaning of that word is in every case. For clarity in the exposition, we refer to Galley et al. (2003) lexical chains as “simple” lexical chains and to Silber and McCoy (2002) and Galley and McKeown (2003) as “complex” lexical chains.

However, later we realized that Galley’s lexical chains are, in fact, not that simple. Since Galley performs stemming as a previous step before building these simple lexical chains, they are actually grouping several possibly related concepts under the same lexical chain. For example, appearances of the words “president”, “presidency”, “presidential”, etc., will be grouped together in a chain with root “presid”. This might explain why our complex lexical chains do not always perform better on topic segmentation than this approach.

Referring back to complex chains, we first implemented Silber and McCoy chains. Those authors propose a linear space and time implementation of lexical chain extraction. They argue that the previous implementation by Barzilay and Elhadad (1997) was exponential in time and not usable for long documents. They compute lexical chains of nouns in two steps:

1. **Building metachains**: Initially we have an array with an entry for every noun sense in WordNet. For every noun in the document, and for every sense of that noun, we insert the noun in those entries for which it has an identity, synonym, or hypernym relationship with that sense. The noun is linked to the previous noun in the chain with a score that depends on the type of relationship between them (identity, synonym, hypernym, hyponym or sibling) and the distance between those nouns in the text. After reading all words, the non-empty entries of the array are called metachains.

2. **Disambiguation**: For every noun, we keep the noun in one metachain, the one where its score contributes the most. In the case of a tie, the word is kept in the more specific (higher WordNet sense number) metachain. Later, we eliminate that noun from every other metachain. After this process, every metachain is a valid lexical chain. It is clear that they use a greedy approach for disambiguation.

We found that this greedy approach did not produce much variety in the lexical chains. Most of them continue being repetitions of the same noun, since ties are frequent and always the more specific sense is used in that case.

That is why we implemented Galley and McKeown (2003) lexical chains. The main differences between this method and the previous are two:

- All appearances of a noun in a document are considered to have the same sense, in opposition to Silber and McCoy, where every word could end up with a different sense.
- As a first step, the method builds a disambiguation graph: every noun is inserted in all its possible senses and edges to the hypernyms, hyponyms and siblings, weighted by a score that depends on the relationship and distance in the text, are drawn. After inserting every noun, we performed disambiguation that, in this case, for every noun keeps the sense (all the words in the document) that has maximum sum of edges from those words toward it.

The lexical chains we get are much richer than with the previous method, because ties are less frequent, so lexical chains with general senses appear more often.

As examples, we show some actual computed lexical chains:

- Idea, goal, intention, sake
- building, door, story, level, complex

The second example is illustrative, since in that case the conversation was about architecture, and story gets the right meaning.

Very interestingly, in our data, when they talk about “mike”, the method is good enough to realize they are talking about “microphones”, so “mike” is actually added to the lexical chain with that meaning.

Unknown words are added to different lexical chains with “unknown” meaning. These words could be people, places, and other important words in our task.

Finally, our data were very rich in meaningful adjectives. For example, during a conversation segment, if we talk about lexical chains, we keep repeating lexical a lot. That adjective has a great importance in the cohesion of the segment. Therefore, we ran some experiments building both noun chains and adjective chains. However, we do not perform word sense disambiguation with adjectives, instead building only simple lexical chains.

Lexical Chain Segmenting Algorithm

Algorithm description

The topic segmentation algorithm we used is the *LCseg* algorithm, developed by Galley et al. (2003). The main hypothesis behind the algorithm is that topic changes are likely to occur where strong lexical cohesion starts and ends, or where it changes most sharply. As mentioned, the original algorithm uses simple lexical chains consisting of only term repetitions, whereas we also applied more complex lexical chains to the algorithm. After the lexical chains are built, we further divide each into smaller chains whenever there is a hiatus of *h* sentences without a term belonging to the chain appearing. Each of these chains is then scored according to a variant of the TF.IDF metric. Our score depends on both term frequency and chain compactness, the hypothesis being that these are good indicators of lexical cohesion. Formally, for each chain R_i, its score is

$$score(R_i) = freq_i * \log(L/L_i)$$

where $freq_i$ is the frequency of the chain's term(s), L is the length of the conversation or text, and L_i is the length of the chain, where length is measured in terms of speaker turns in conversations and sentences in text.

From this we can then compute the lexical cohesion score at each speaker turn or sentence break, where we consider two analysis windows of size *k* adjacent to the break. We do so by calculating the cosine similarity between the two windows (which we will call *A* and *B*), with the definition

$$\cos ine(A,B) = \frac{\sum_i w_{i,A} \cdot w_{i,B}}{\sum_i w_{i,A}^2 \sum_i w_{i,B}^2}$$

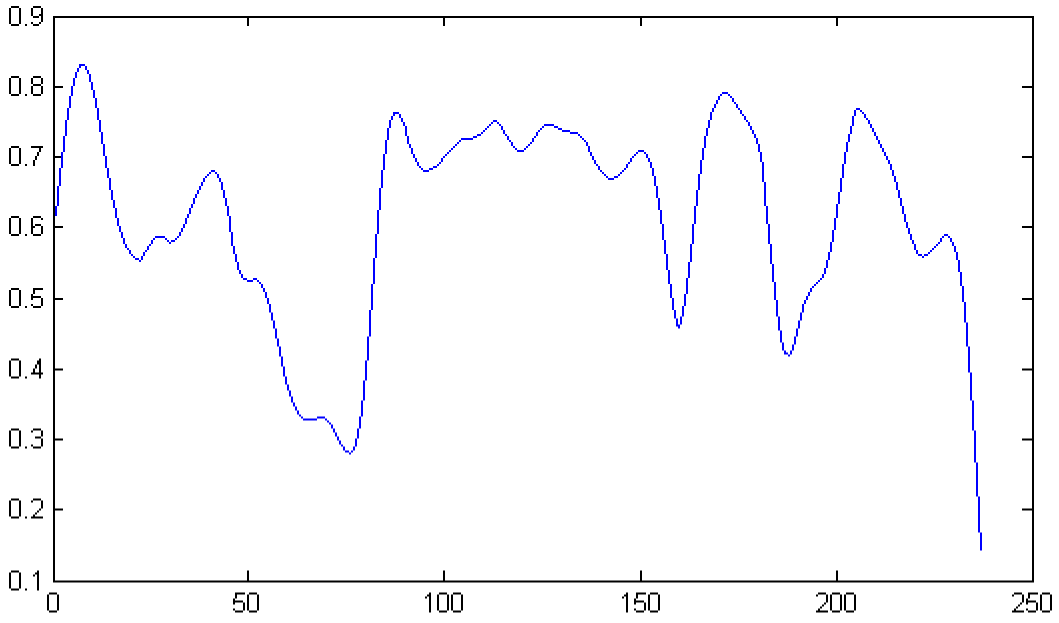
where

$$w_{i,\Gamma} = \begin{cases} score(R_i) & \text{if } R_i \text{ overlaps } \Gamma \in \{A,B\} \\ 0 & \text{otherwise} \end{cases}$$

The lexical cohesion scores give us a lexical cohesion function (LCF) over all the breaks (and thus over time). We then smooth the function using a moving average filter, and consider the resulting local minima as potential topic segment boundaries. The probability $p(m_i)$ of each local minimum m_i being a segment boundary is determined by how sharply the LCF changes there. More precisely, $p(m_i)$ is the sum of the differences in LCF between m_i and the local maximum left of m_i and between m_i and the local maximum right of m_i.

We show in the following figure an example of a lexical cohesion function graph.

Figure 1: This is a concatenation of nine WSJ articles, with the articles beginning at lines 24, 31, 61, 65, 73, 80, 162 and 188. The horizontal axis represents line numbers and the vertical axis represents the lexical cohesion score. We can see sharp changes at some of these points or near them, such as the big dips that seem to correspond roughly to lines 75, 160 and 190.



Finally, after determining the LCF, we can guess the boundaries. When the number of boundaries is known, we simply select the n - 1 m_i's with the highest $p(m_i)$'s, where n is the number of topics. When the number of boundaries is not given, we compute a threshold based on the mean and variance of the $p(m_i)$'s and select only those points whose probabilities are above the threshold.

Implementation description

The first step is building the lexical chains, described in more detail above. For the simple chains with simple repetitions done in the original algorithm, because of the simplicity, this is done in the `LCseg` class, whereas for the more complex chains, this is done outside first using `ConversationReader` and then fed to `LCseg`. The implementation of the algorithm is straightforward and follows the process described above. We use the same h, k and threshold parameters proposed by Galley *et al*. as those giving the best performance. They do not, however, specify parameters for the moving average filter used to smooth the LCF, although the size f of the filter window seems to matter, with some values of f working better for one type of lexical chain than it does for another. Thus we tune the parameter f using the development dataset for each type of lexical chain in each experiment setup.

Running experiments is done using `LCsegTests`, which does parameter tuning and then tests using the tuned parameters. As building the complex chains is time consuming, the way we did the testing was to first run ConversationReader and save all the complex chains to file, and then whenever we test `LCsegTests` will read the chains from file and then feed them to `LCseg` for topic segmentation.

In our implementation of the algorithm, we use the same *h*, *k* and threshold parameters proposed by Galley *et al*. as those giving the best performance. They do not, however, specify parameters for the moving average filter used to smooth the LCF, although the size *f* of the filter window seems to matter, with some values of *f* working better for one type of lexical chain than it does for another. Thus we tune the parameter *f* using the development dataset for each type of lexical chain in each experiment setup.

Testing and Results

Evaluation method

We evaluated the performance of our segmenters on both the "known" task and the "unknown" task, i.e., the task in which the number of topic boundaries is already given, and the task in which the number of topic boundaries is not given. For evaluating segmentation accuracy, we use the P_k and WindowDiff (*WD*) error metrics, the same as those used by Galley *et al.* (2003) and Purver *et al.* (2006). P_k is the probability that speaker turns or sentences *k* units apart are incorrectly determined as being in the same segment or in different segments, i.e., an error occurs whenever two speaker turns or sentences *k* units apart are not separated by any hypothesized boundaries although they are separated by some reference boundary, or when they are separated by a hypothesized boundary but are not separated by any reference boundaries. Note that *k* here is not the same as the lexical cohesion analysis window size *k* mentioned in the previous section, but taken to be half of the average length of the segments. WindowDiff was proposed by Pevzner and Hearst (2002) as a better error metric than P_k. Pevzner and Hearst noted several problems with the P_k metric, such as that it penalizes false negatives more than false positives and that it sometimes allows extra boundaries to go unpenalized, and WindowDiff solves these problems. It is the probability that for any two speaker turns or sentences *k* units apart, the number of hypothesized boundaries that are in between the two is equal to the number of reference boundaries that fall in this interval.

About one fifth of the corpus is used as used for development and the remainder for testing.

Preliminary evaluation for implementation details

There are details in the implementation of the *LCseg* algorithm that Galley *et al*. do not specify, and we used our development set to see what seems to work best. For instance, we found that we do slightly better if we ignore the points at which sentences become empty strings after the stop words are taken out when we are computing the lexical cohesion score at each sentence break. When we incorporated the complex lexical chains with *LCseg*, we also tried to see whether performance would improve if in computing the score $score(R_i) = freq_i * \log(L / L_i)$ of each chain R_i, we replace the log term with the IDF measure of the complex chain, which takes into account other conversations in the corpus instead of just the current conversation. This, however, did not improve performance. Also, as mentioned, the smoothing filter window size *f* is tuned for each chain in each experimental setup.

Experiment 1: Whole conversations from the ICSI corpus

As mentioned, for each conversation we also have a version that is considered in terms of sentences instead of speaker turns. We can see this as a heuristic to limit the number of speaker turn breaks that we consider as potential topic boundaries, since sometimes one sentence would be separated into several speaker turns, as there are pauses in between. We ran *LCseg* on both the conversations with the original speaker turns and with the sentences, with the results shown below. "Orig" below stands for the original *LCseg* algorithm with the simple term repetition lexical chains, "SMC" for the Silbery & McCoy lexical chains, "GMK" for Galley & McKeown chains with only nouns, and "GMK-a" for Galley & McKeown chains with both nouns and adjectives. "Orig-snt" stands for the original *LCseg* algorithm applied to sentences, and so on. We also provide the baseline for each task (where for the known task with *n* boundaries, we randomly select *n* boundaries and for the unknown task, we first randomly select *n* from the range of zero to twice the average number of boundaries).

Table 1: Performance of *LCseg* with different lexical chains on entire conversations

	Known		Unknown	
	P_k	*WD*	P_k	*WD*
Orig	29.85%	39.71%	28.32%	41.15%
SMC	31.50%	37.02%	30.10%	36.06%
GMK	29.65%	37.04%	29.69%	38.76%
GMK-a	29.58%	37.14%	29.15%	38.40%
Orig-snt	29.55%	38.26%	30.84%	47.17%
SMC-snt	30.99%	37.43%	31.16%	37.15%
GMK-snt	29.52%	37.26%	29.51%	38.24%
GMK-a-snt	29.75%	37.38%	29.34%	38.24%
baseline	41.21%	48.35%	44.61%	51.28%

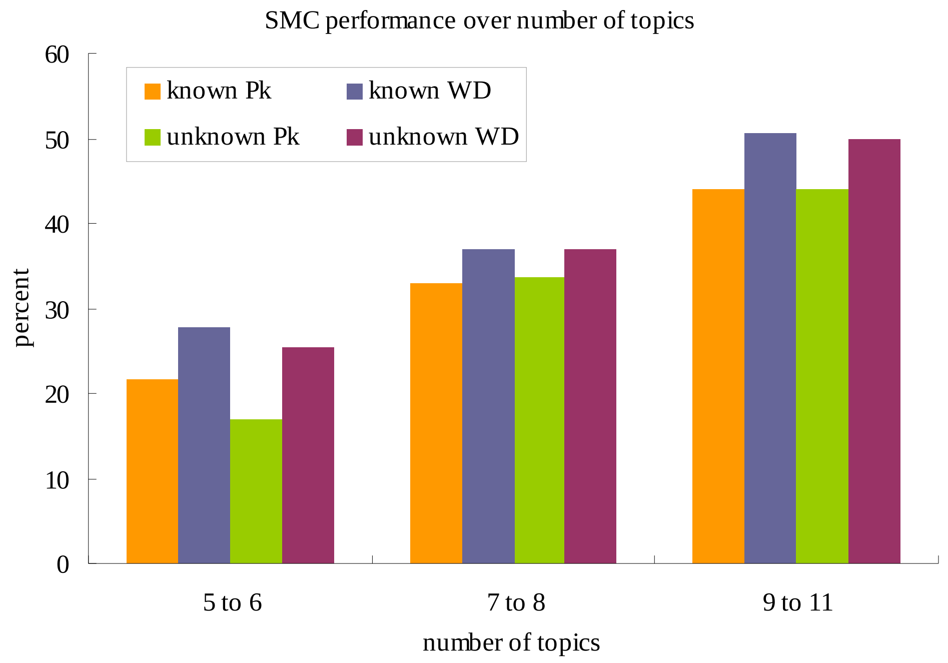
We can see that incorporating the complex chains into *LCseg* does not make it do worse than the original algorithm in any category, but it also does not improve performance according to the P_k measure. However, in terms of WindowDiff, the three more complex chains performed very slightly better than the original algorithm in the known task, and in the unknown task, *LCseg* with Silber and McCoy chains incorporated did considerably better than the original algorithm in terms of WindowDiff, and the other two more complex chains also performed slightly better than the original with this metric.

Unfortunately, we also observe that two things we tried adding did not help with performance here. Whether or not we treat the conversation in terms of sentences instead of speaker turns does not seem to affect performance, nor does adding adjectives to the Galley and McKeown lexical chains.

Analysis: Number of topics vs. accuracy

We wondered whether it was the case that conversations or documents with fewer boundaries generally do better than those with a higher number of boundaries. We ran *LCseg* with the Silber & McCoy chains on the whole conversations dataset and put the results into several topic boundaries number bins, producing the following graph, where we show the average errors for each group.

Figure 2: Performance of *LCseg* using Silber & McCoy chains on the whole conversations dataset over the number of topics in the conversations.



We can see that each of the four error metrics increase as the size of topic boundaries increase, so it seems that when there are more topics, it is harder to identify their boundaries accurately.

Experiment 2: Cutting out the off-topic segments

The ICSI conversations contain segments that are off-topic and do not pertain to the content of the meeting, often at the beginning and end of conversations. These segments include informal chat, agenda-discussion, and ICSI "digit-task" sections in which speakers read aloud digits to provide data for speech recognition experiments. These sections are annotated in the data, and they were actually removed from the data used by Galley et al., Therefore, we also evaluated our lexical chains on modified conversations with these off-topic segments cut out, with the results shown below.

Table 2: Performance of *LCseg* with different lexical chains on only content topics

	Known		Unknown	
	P_k	*WD*	P_k	*WD*
Orig	27.31%	30.22%	29.72%	34.59%
SMC	29.41%	29.69%	29.23%	30.09%
GMK	29.53%	29.72%	29.76%	29.95%
GMK-a	29.22%	29.72%	29.76%	29.95%
Orig-snt	29.10%	30.56%	33.10%	34.14%
SMC-snt	28.73%	28.89%	29.07%	29.23%
GMK-snt	28.53%	28.72%	29.11%	29.30%
GMK-a-snt	28.70%	29.00%	29.02%	29.19%
baseline	36.36%	40.09%	38.97%	43.14%

These results support what the results on the whole conversations suggested: it seems that the three complex chains do better than the original algorithm in the unknown task, at least according to the WindowDiff measure, which is supposed to be a less biased measure than P_k.

We also note that our numbers for the original algorithm are very close to those given in the Galley *et al.* paper, a good indication that we implemented the algorithm correctly.

Experiment 3: The WSJ corpus

Besides the ICSI meeting corpus, we also put together a corpus from WSJ news stories. Instead of conversations with different topics, we have concatenations of news articles, treating each article as a separate topic, with the number of articles in each concatenation ranging from 4 to 22. We have 119 concatenations in total. Instead of sentences, we simply treated them as the lines that they appeared as in the data. Here are our results.

Table 3: Performance of *LCseg* with different lexical chains on the WSJ corpus

	Known		Unknown	
	P_k	*WD*	P_k	*WD*
Orig	24.75%	30.61%	25.29%	31.31%
SMC	28.61%	34.54%	28.47%	34.50%
GMK	24.28%	30.22%	25.27%	30.42%
GMK-a	24.88%	30.84%	24.83%	29.93%
baseline	45.46%	51.36%	46.06%	54.88%

Here the Silber & McCoy chain actually seems to not perform so well compared to the other complex lexical chains and the simple lexical chain, which give very similar performances. Using complex chains appears to give no improvement on the WSJ task.

Conclusion and Further Work

We started the project very motivated by the possible better performance of full lexical chains in topic segmentation, compared with simple chains based on word repetition.

However, we did not achieve a significant improvement of the performance of the original method. It seems that stemming plus grouping of roots performs as well as word sense disambiguation. However, and the reason is still not clear to us, word sense disambiguation improves the segmentation results in the meeting data when the number of segments is unknown. In addition, our results support the fact that we could only use nouns for topic segmentation with lexical chains and still get results similar in quality to those obtained by the original algorithm, which uses roots of nouns, adjectives and verbs.

As future work, we could use full anaphora resolution to build more robust lexical chains. However, we are skeptical that these extended lexical chains will improve the results. More likely to work are approaches that use other features in the conversations, such as pauses, phrases such as “let’s change topic”, etc., to extend our previous method.

Performing named entity recognition (NER) as a preprocessing step might also help since it is likely that names of people, places, organizations, etc. are frequently topics of conversation. Running a robust NER system could add to the validity of our lexical chains, rather than having to rely entirely on WordNet.

Additionally, we believe our complex lexical chains might perform better topic segmentation on a genre where using many different words to refer to the same concept is considered good writing, such as literature. In such a domain, complex lexical chains would hopefully find nuances that are lost on the simple chains.

Collaboration Statement

Luis, Mike, and Helen all worked together discussing what we should do for the project and investigating existing work. For programming, Mike focused on reading and preprocessing the data as well as an attempt at simple anaphora resolution, Luis focused on getting the lexical chains built, and Helen focused on implementing the algorithm to determine topic boundaries. All three worked on the write-up together.

All the late days will be distributed evenly among the members of the team, allowing the project to be turned in 3 days late with no penalties.

References

Regina Barzilay, Michael Elhadad (1997). Using Lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summmarization Workshop (ISTS-97),* Madrid, Spain.

Michel Galley, Kathleen McKeown (2003). Improving Word Sense Disambiguation in Lexical Chaining. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*. Poster paper. August 9-15, 2003. Acapulco, Mexico.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, Hongyan Jing (2003). Discourse Segmentation of Multi-Party Conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*. July 21-26, 2003. Sapporo, Japan.

George A. Miller et al. WordNet. *Cognitive Science Laboratory, Princeton University*. http://wordnet.princeton.edu. 2007.

Lev Pevzner and Marti Hearst. (2002) A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, Vol. 28, No. 1, pages 19-36.

Matthew Purver, Konrad Körding, Tom Griffiths and Josh Tenenbaum. (2006) Unsupervised Topic Modelling for Multi-Party Spoken Discourse. In *Proceedings of COLING/ACL 2006*, pages 17-24, Sydney, Australia, July 2006.

Gregory Silber and Kathleen F. McCoy (2002). Efficiently Computed Lexical

Chains as an IntermediateRepresentation for Automatic Text Summarization. In *Computational Linguistics*.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer (2003). Feature Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL 2003* pages 252-259.