Lexical Chains and Sliding Locality Windows in Content-based Text Similarity Detection

Thade Nahnsen, Özlem Uzuner, Boris Katz
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
{tnahnsen,ozlem,boris}@csail.mit.edu

Abstract

We present a system to determine content similarity of documents. Our goal is to identify pairs of book chapters that are translations of the same original chapter. Achieving this goal requires identification of not only the different topics in the documents but also of the particular flow of these topics.

Our approach to content similarity evaluation employs n-grams of lexical chains and measures similarity using the cosine of vectors of n-grams of lexical chains, vectors of tf*idf-weighted keywords, and vectors of unweighted lexical chains (unigrams of lexical chains). Our results show that n-grams of unordered lexical chains of length four or more are particularly useful for the recognition of content similarity.

1 Introduction

This paper addresses the problem of determining content similarity between chapters of literary novels. We aim to determine content similarity even when book chapters contain more than one topic by resolving exact content matches rather than finding similarities in dominant topics. Our solution to this problem relies on lexical chains extracted from WordNet [6].

2 Related Work

Lexical Chains (LC) represent lexical items which are conceptually related to each other, for example, through hyponymy or synonymy relations. Such conceptual relations have previously been used in evaluating cohesion, e.g., by Halliday and Hasan [2, 3]. Barzilay and Elhadad [1] used lexical chains for text summarization; they identified important sentences in a document by retrieving strong chains. Silber and McCoy [7] extended the work of Barzilay and Elhadad; they developed an algorithm that is linear in time and space for efficient identification of lexical chains in large documents. In this algorithm, Silber and McCoy first created a text representation in the form of metachains, i.e., chains that capture all possible lexical chains in the document. After creating the metachains, they used a scoring algorithm to identify the lexical chains that are most relevant to the document, eliminated unnecessary overhead information from the metachains, and selected the lexical chains representing the document. Our method for building lexical chains follows this algorithm.

N-gram based language models, i.e., models that divide text into *n*-word (or *n*-character) strings, are frequently used in natural language processing. In plagiarism detection, the overlap of n-grams between two documents has been used to determine whether one document plagiarizes another [4]. In general, n-grams capture local relations. In our case, they capture local relations between lexical chains and between concepts represented by these chains.

Three main streams of research in content similarity detection are: 1) shallow, statistical analysis of documents, 2) analysis of rhetorical relations in texts [5], and 3) deep syntactic

analysis [8]. Shallow methods do not include much linguistic information and provide a very rough model of content while approaches that use syntactic analysis generally require significant computation. Our approach strikes a compromise between these two extremes: it uses the linguistic knowledge provided in WordNet as a way of making use of low-cost linguistic information for building lexical chains that can help detect content similarity.

3 Lexical Chains in Content Similarity Detection

3.1 Corpus

The experiments in this paper were performed on a corpus consisting of chapters from translations of four books (Table 1) that cover a variety of topics. Many of the chapters from each book deal with similar topics; therefore, fine-grained content analysis is required to identify chapters that are derived from the same original chapter.

# translations	Title	# chapters
2	*20,000 Leagues under the Sea*	47
3	*Madame Bovary*	35
2	*The Kreutzer Sonata*	28
2	*War and Peace*	365

Table 1: Corpus

3.2 Computing Lexical Chains

Our approach to calculating lexical chains uses nouns, verbs, and adjectives present in WordNetV2.0. We first extract such words from each chapter in the corpus and represent each chapter as a set of these word instances $\{I_1, \ldots, I_n\}$. Each instance of each of these words has a set of possible interpretations, I_N, in WordNet. These interpretations are either the synsets or the hypernyms of the instances. Given these interpretations, we apply a slightly modified version of the algorithm by Silber and McCoy [7] to automatically disambiguate nouns, verbs, and adjectives, i.e., to select the correct interpretation, for each instance. Silber and McCoy's algorithm computes all of the scored metachains for all senses of each word in the document and attributes the word to the metachain to which it contributes the most. During this process, the algorithm computes the contribution of a word to a given chain by considering 1) the semantic relations between the synsets of the words that are members of the same metachain, and 2) the distance between their respective instances in the discourse. Our approach uses these two parameters, with minor modifications. Silber and McCoy measure distance in terms of paragraphs on prose text; we measure distance in terms of sentences in order to handle both dialogue and prose text.

Original document (underlined words are represented with lexical chains):
The furniture in the kitchen seems beautiful, but the bathroom seems untidy.

Intermediate representation (lexical chains):
03281101 03951013 02071636 00218842 03951013 02071636 02336718

Figure 1: Intermediate representation after eliminating words that are not nouns, verbs, or adjectives and after identifying lexical chains (represented by WordNet synset IDs). Note that {kitchen, bathroom} are represented by the same synset ID which corresponds to the synset ID of their common hypernym "room". {kitchen, bathroom} is a lexical chain. Ties are broken in favor of hypernyms.

Following Silber and McCoy, we allow different types of conceptual relations to contribute differently to each lexical chain, i.e., the contribution of each word to a lexical chain is dependent on its semantic relation to the chain (see Table 2). After scoring, concepts that are dominant in the text segment are identified and each word is represented by only the WordNet ID of the synset (or the hypernym/hyponym set) that best fits its local context. Figure 1 gives an example of the resulting intermediate representation, corresponding to the interpretation, S, found for each word instance, I, that can be used to represent each chapter, C, where $C = \{S_1, \ldots, S_m\}$.

Lexical semantic relation	Distance <= 6 sentences	Distance > 6 sentences
Same word	1	0
Hyponym	0.5	0
Hypernym	0.5	0
Sibling	0.2	0

Table 2: Contribution to lexical chains

3.3 Determining the Locality Window

After computing the lexical chains, we created a representation for text by substituting the correct lexical chain for each noun, verb, and adjective in each document. We omitted the remaining parts of speech from the documents (see Figure 1 for sample intermediate representation). We obtained ordered and unordered n-grams of lexical chains from this representation.

Ordered n-grams consist of *n* consecutive lexical chains extracted from text. These ordered n-grams preserve the original order of the lexical chains in the text. Corresponding unordered n-grams disregard this order. The resulting text representation is $T = \{gram_1, gram_2, \ldots, gram_n\}$, where $gram_i = [lc_1, \ldots, lc_n]$, where $lc_i \in \{I_1, \ldots, I_k\}$ (the chains that represent Chapter C). The elements in $gram_i$ may be sorted or unsorted, depending on the selected method. N-grams are extracted from text using sliding locality windows and provide what we call "attribute vectors". The attribute vector for ordered n-grams has the form $C = \{(e_1, \ldots, e_n), (e_2, \ldots, e_{n+1}), \ldots, (e_{m-n}, \ldots, e_m)\}$ where $(e_1, \ldots, e_n)$ is an ordered n-gram and e_m is the last lexical chain in the chapter. For unordered n-grams, the attribute vector has the form $C = \{sort[(e_1, \ldots, e_n)], sort[(e_2, \ldots, e_{n+1})], \ldots, sort[(e_{m-n}, \ldots, e_m)]\}$ where sort[…] indicates alphabetical sorting of chains (rather than the actual order in which the chains appear in the text).

We evaluated similarity between pairs of book chapters using the cosine of the attribute vectors of n-grams of lexical chains (sliding locality windows of width n). We varied the width of the sliding locality windows from two to five elements.

4 Evaluation

We used cosine similarity as the distance metric, computed the cosine of the angle between the vectors of pairs of documents in the corpus, and ranked the pairs based on this score. We identified the top *n* most similar pairs (also referred to as "selection level of *n*") and considered them to be similar in content.

We calculated similarity between pairs of documents in several different ways, evaluated these approaches with the standard information retrieval measures, i.e., precision, recall, and f-measure, and compared our results with two baselines. The first baseline measured the similarity of documents with tf*idf-weighted keywords; the second used the cosine of unweighted lexical chains (unigrams of lexical chains).

The corpus of parallel translations provides data that can be used as ground truth for content similarity; corresponding chapters from different translations of the same original title are considered similar in content, i.e., chapter 1 of translation 1 of *Madame Bovary* is similar in content to chapter 1 of translation 2 of *Madame Bovary*.

Figure 2 shows the f-measure of different methods for measuring similarity between pairs of chapters using ordered lexical chains, unordered lexical chains, and baselines. These graphs present the results when the top 100–1,600 most similar pairs in the corpus are considered similar in content and the rest are considered dissimilar (selection level of 100–1,600). The total number of chapter pairs is approximately 1,000,000. Of these, 1,080 (475 unique chapters with 2 or 3 translations each) are considered similar for evaluation purposes.

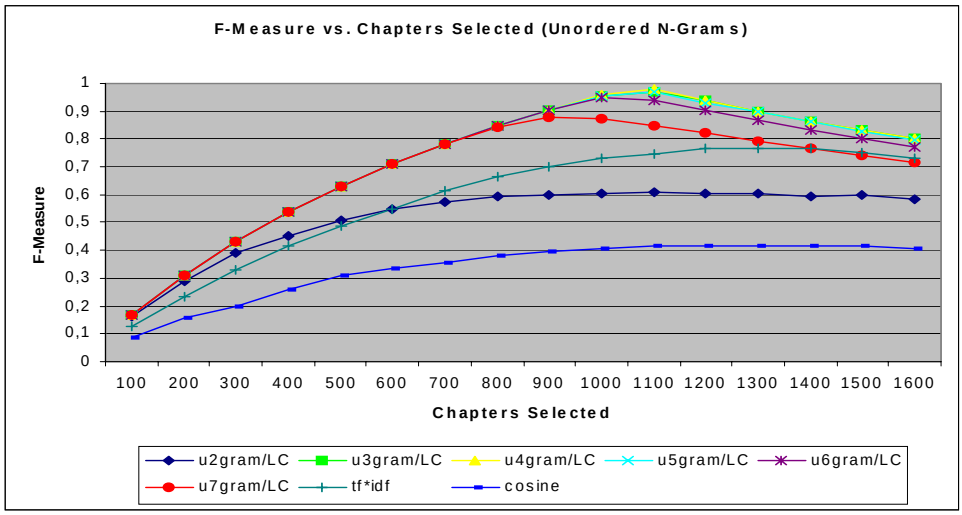
The results indicate that four similarity measures gave the best performance. These were tri-grams, quadri-grams, penta-grams, and hexa-grams of unordered lexical chains. The peak f-measure at the selection level of 1,100 chapter pairs was 0.981. Chi squared tests performed on the f-measures (when the top 1,100 pairs were considered similar) were significant at $p = 0.001$.

Closer analysis of the graphs in Figure 2 shows that, at the optimal selection level, n-grams of ordered lexical chains of length greater than four significantly outperformed the baseline at $p = 0.001$ while n-grams of ordered lexical chains of length less than or equal to four are significantly outperformed by the baseline at the same p. A similar observation cannot be made for the n-grams of unordered lexical chains; for these n-grams, the performance degradation appears at $n = 7$, i.e., the corresponding curves have a steeper negative incline than the baseline.

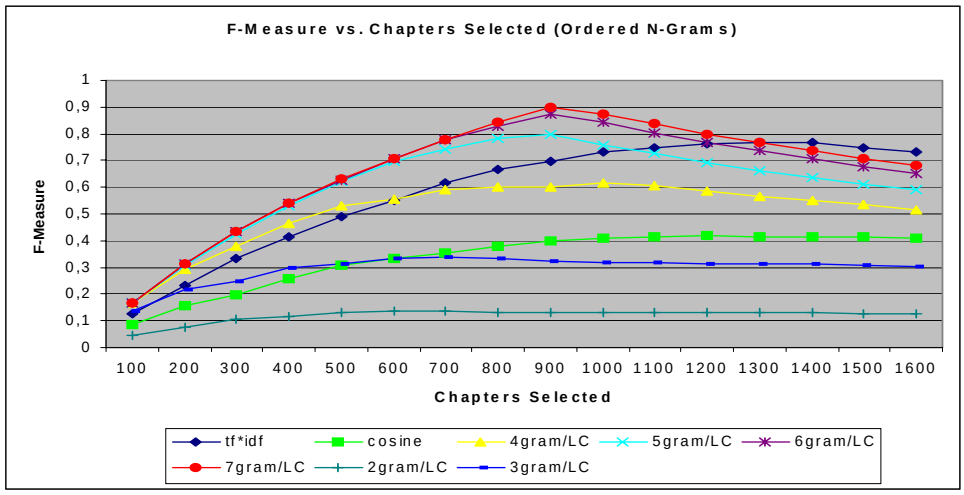
After the cut-off point of 1,100 chapter pairs, the performance of all algorithms declines. This is due to the evaluation method we have chosen: although the cut-off for similarity judgement can be increased, the number of chapters that are in fact similar does not change and at high cut-off values many dissimilar pairs are considered similar, leading to degradation in performance.

Figures 2a and 2b show that some of the lexical chain representations do not outperform the tf*idf-weighted baseline. A comparison of Figures 2a and 2b shows that, for n < 5, n-grams of ordered lexical chains perform worse than n-grams of unordered lexical chains. This indicates that between different translations of the same book the order of chains changes significantly, but that the chains within contiguous regions (locality windows) of the texts remain similar.

Interestingly, ordered n-grams of length 3 to 5 perform significantly better than unordered n-grams of the same length. This implies that, during translation, the order of the content words does not change enormously for three to five lexical chain elements. Allowing flexible order for the lexical chains (i.e., unordered lexical chains) in these n-grams therefore hurts performance by allowing many false positives. However, for longer n-grams to be successful, the order of the lexical chains has to be flexible.



(a) F-Measure: Unordered n-grams vs. the baselines



(b) F-Measure: Ordered n-grams vs. the baselines

ngram/LC – unordered n-grams of lexical chains are used in the attribute vector
ungram/LC – ordered n-grams of lexical chains are used in the attribute vector
tf*idf – tf*idf weighted words are used in the attribute vector
cosine – the standard information retrieval measure; words are used in the attribute vector

Figure 2: F-Measure

5 Future Work

Currently, our similarity measures do not employ any weighting scheme for n-grams, i.e., every n-gram is given the same weight. For example, the n-gram "be it as it has been" in lexical chain form corresponds to synsets for the words *be*, *have* and *be*. The trigram of these lexical chains does not convey significant meaning. On the other hand, the n-gram "the lawyer signed the heritage" is converted into the trigram of lexical chains of *lawyer*, *sign*, and *heritage*. This trigram is more meaningful than the trigram *be have be,* but in our scheme both trigrams will get the same weight. As a result, two documents that share the trigram *be have be* will look as similar as two documents that share *lawyer sign heritage*. This problem can be addressed in two possible ways: using a 'stop word' list to filter such expressions completely or giving different weights to n-grams based on the number of their occurrences in the corpus.

6 Conclusion

We have presented a system that extends previous work on lexical chains to content similarity detection. This system employs lexical chains and sliding locality windows, and evaluates similarity using the cosine of n-grams of lexical chains and tf*idf weighted keywords. The results indicate that lexical chains are effective for detecting content similarity between pairs of chapters corresponding to the same original in a corpus of parallel translations.

References

1. Barzilay, R., Elhadad, M. 1999. Using lexical chains for text summarization. In: Inderjeet Mani and Mark T. Maybury, eds., *Advances in AutomaticText Summarization*, pp. 111–121. Cambridge/MA, London/England: MIT Press.
2. Halliday, M. and Hasan, R. 1976. *Cohesion in English*. Longman, London.
3. Halliday, M. and Hasan, R. 1989. *Language, context, and text*. Oxford University Press, Oxford, UK.
4. Lyon, C., Malcolm, J. and Dickerson, B. 2001. Detecting Short Passages of Similar Text in Large Document Collections, *In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pp.118-125.
5. Marcu, D. 1997. *The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts* (Ph.D. dissertation). Univ. of Toronto.
6. Miller, G., Beckwith, R., Felbaum, C., Gross, D., and Miller, K. 1990. Introduction to WordNet: An online lexical database. *J. Lexicography*, 3(4), pp. 235-244.
7. Silber, G. and McCoy, K. 2002. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4).
8. Uzuner, O., Davis, R., Katz, B. 2004. Using Empirical Methods for Evaluating Expression and Content Similarity. In: *Proceedings of the 37th Hawaiian International Conference on System Sciences* (HICSS-37). IEEE Computer Society.