Efficiently Computed Lexical Chains As an Intermediate Representation for Automatic Text Summarization

H. Gregory Silber*
University of Delaware

Kathleen F. McCoy†
University of Delaware

While automatic text summarization is an area that has received a great deal of attention in recent research, the problem of efficiency in this task has not been frequently addressed. When considering the size and quantity of documents available on the Internet and from other sources, the need for a highly efficient tool that produces usable summaries is clear. We present a linear time algorithm for lexical chain computation. The algorithm makes lexical chains a computationally feasible candidate as an intermediate representation for automatic text summarization. A method for evaluating lexical chains as an intermediate step in summarization is also presented and carried out. Such an evaluation was not possible before due to the computational complexity of previous lexical chains algorithms.

1 Introduction

The overall motivation for this research is a computationally efficient system to create summaries automatically. Summarization has been viewed as a two step process. The first step is the extraction of important concepts from the source text by building an intermediate representation of some sort. The second step uses this intermediate representation to generate a summary (Sparck Jones, 1993).

In this research we concentrate on the first step of the summarization process and follow Barzilay and Elhadad (1997) in employing lexical chains to extract important concepts from a document. We present a linear time algorithm for lexical chain computation and offer an evaluation that indicates that these chains are a promising avenue of study as an intermediate representation in the summarization process.

Barzilay and Elhadad (1997) proposed lexical chains as an intermediate step in the text summarization process. Determining the benefit of this proposal has been faced with a number of difficulties. First, previous methods for computing lexical chains have either been manual (Morris and Hirst, 1991), or automated, but with exponential efficiency (Hirst and St-Onge, 1997) (Barzilay and Elhadad, 1997). Because of this, computing lexical chains for documents of any reasonable size has been impossible. We present here an algorithm for computing lexical chains that is linear in space and time. This algorithm makes the computation of lexical chains computationally feasible even for large documents.

A second difficulty faced in evaluating Barzilay and Elhadad's proposal is that it is a proposal for the first stage of the summarization process, and it is not clear how to evaluate this stage independently of the second stage of summarization. The second contribution of this paper is a method for evaluating lexical chains as an intermedi-

* Department of Computer and Information Sciences, Newark, DE 19711
† Department of Computer and Information Sciences, Newark, DE 19711

ate representation. The intuition behind the method is as follows. The (strong) lexical chains in the document are intended to identify important (noun) concepts in a document. Our evaluation requires access to documents that have corresponding human generated summaries. We run our lexical chain algorithm on both the document and on the summary and examine: (1) how many of the concepts from strong lexical chains in the document also occur in the summary, and (2) how many of the (noun) concepts appearing in the summary are represented in strong lexical chains in the document.

Essentially, if lexical chains are a good intermediate representation for text summarization, we expect that concepts identified as important according to the lexical chains will be the concepts found in the summary. Our evaluation of 24 documents with summaries indicates that indeed lexical chains do appear to be a promising avenue of future research in text summarization.

1.1 Lexical Chains Described

The concept of lexical chains was first introduced by Morris and Hirst. Basically, lexical chains exploit the cohesion among an arbitrary number of related words (Morris and Hirst, 1991). Lexical chains can be computed in a source document by grouping (chaining) sets of words that are semantically related (i.e. have a sense flow). Identities, synonyms, and hypernyms/hyponyms[1] are the relations among words that might cause them to be grouped into the same lexical chain. Specifically, words may be grouped when:

- Two noun instances are identical, and are used in the same sense.
  (*The house on the hill is large. The house is made of wood.*)
- Two noun instances are used in the same sense (i.e., are synonyms).
  (*The car is fast. My automobile is faster.*)
- The senses of two noun instances have a hypernym/hyponym relation between them. (*John owns a car. It is a Toyota.*)
- The senses of two noun instances are siblings in the hypernym/hyponyn tree.
  (*The truck is fast. The car is faster.*)

In computing lexical chains, the noun instances must be grouped according to the above relations, but each noun instance must belong to exactly one lexical chain. There are several difficulties in determining which lexical chain a particular word instance should join. For instance, a particular noun instance may correspond to several different word senses and thus the system must determine which sense to use (e.g. should a particular instance of "house" be interpreted as sense 1: dwelling, or sense 2: legislature). In addition, even if the word sense of an instance can be determined, it may be possible to group that instance into several different lexical chains because it may be related to words in different chains. For example, the word's sense may be identical to that of a word instance in one grouping while having a hypernym/hyponym relationship with that of a word instance in another. What must happen is that the words must be grouped in such a way that the overall grouping is optimal in that it creates the longest/strongest lexical chains. It is our contention that words are grouped into a single chain when they are "about" the same underlying concept.

1 which together define a tree of "isa" relations between words.

2 Algorithm Definition

We wish to extract lexical chains from a source document using the complete method that Barzilay and Elhadad implemented in exponential time, but to do so in linear time. Barzilay and Elhadad define an interpretation as a mapping of noun instances to specific senses, and further of these senses to specific lexical chains. Each unique mapping is a particular "way of interpreting" the document, and the collection of all possible mappings defines all of the interpretations possible. In order to compute lexical chains in linear time, instead of computing every interpretation of a source document as Barzilay and Elhadad did, we create a structure that implicitly stores every interpretation without actually creating them, thus keeping both the space and time usage of the program linear. We then provide a method for finding that interpretation which is best from within this representation. As was the case with Barzilay and Elhadad, we rely on WordNet[2] to provide sense possibilities for, and semantic relations among, word instances in the document.

Before actually computing the interpretations, one issue we had to tackle was the organization and speed of the WordNet Dictionary. In order to provide expedient access to WordNet, we recompiled the noun database into a binary format and memory mapped it so that it can be accessed as a large array, changing the WordNet sense numbers to match the array indexes.

2.0.1 Chain Computation Before computation can begin, the system uses a part of speech tagger[3] to find the noun instances within the document. Processing a document involves creating a large array of "Meta-Chains", the size of which is the number of noun senses in WordNet plus the number of nouns in the document to handle the possibility of words not found in WordNet. (This is the maximum size that could possibly be needed). A "meta-chain" represents all possible chains that can contain the sense whose number is the index of the array. When a noun is encountered, for every sense of the noun in WordNet, the noun sense is placed into every meta-chain for which it has an identity, synonym, or hyperonym relation with that sense. These meta-chains represent every possible interpretation of the text. Note that each "meta-chain" has an index which is a WordNet sense number, so in a very real way, we can say that a chain has an "overriding sense". Figure 1 shows the structure of this array with each row being a "meta-chain" based on the sense listed in the first column. In each node of a given meta-chain, appropriate pointers are kept to allow fast lookup. In addition, (not shown in figure) in association with each word, a list of the meta-chains it belongs to is also kept.

The second step, finding the best interpretation, is accomplished by making a second pass through the source document. For each noun, look at each chain it belongs to and figure out, based on the type of relation and distance factors, which "meta-chain" the noun contributes to most. In the event of a tie, the higher sense number is used since WordNet is organized with more specific concepts indexed with higher numbers. The noun is then deleted from all other "meta-chains". Once all of the nouns have been processed, what is left is the interpretation whose score is maximum. From this interpretation, the best chains (highest scoring) can be selected. The algorithm in its entirety is outlined in figure 2.

2 WordNet is available at http://www.cogsci.princeton.edu/∼wn.
3 Available from http://www.rt66.com/gcooke/SemanTag.

Index	Meaning	Chain		
0	person	John	Machine	
1	unit	Computer	IBM	
2	device	Computer	Machine	IBM
3	organization	Machine	IBM	
4	unknown	IBM		
...				
N				

Assume the sentences "John has a computer. The machine is an IBM." and that the nouns have the following senses (meanings): John (0), computer (1,2), machine (0,2,3), IBM (1,2,3,4), and that words are put in a chain if they have an identity relation. Then this table depicts the meta-chains after the first step.

Figure 1
Example of "meta-chains"

Step 1	For each noun instance
	    For each sense of that noun instance
	        Compute all scored "meta-chains"

Step 2	For each noun instance
	    For each "meta-chain" which that noun belongs to
	        Keep word instance in the "meta-chain" to which it contributes most
	            Updating the scores of each other "meta-chain"

Figure 2
Basic linear time lexical chains algorithm

2.0.2 Scoring System Our scoring system allows different types of relations within a lexical chain to contribute to that chain differently. Further, our scoring system allows distance between word instances in the original document to affect the word's contribution to a chain. Figure 3 shows the scoring values used by our algorithm. These values were obtained through empirical testing, and while not optimal, appear to give good results.

	1 Sentence	3 Sentences	Same Paragraph	Default
Identical word	1	1	1	1
Synonym	1	1	1	1
Hyperonym	1	.5	.5	.5
Sibling	1	.3	.2	0

Figure 3
Scoring system tuned by empirical methods

Each noun instance is included in a chain because it is either the first noun instance to be inserted, or it is related to some word that is already in the chain. If it is the first word, then the "Identical Word" relation score is used. If not, then the type of relation is determined, and the closest noun in the chain to which it is related is found. Using the distance between these two words, and the relation type, we look up the contribution to the overall chain score of this word instance.

Once chains are computed, some of the high scoring ones must be picked as representing the important concepts from the original document. To select these, we use the idea of "strong chains" introduced by Barzilay and Elhadad (1997). They define a strong chain as any chain whose score is more than two standard deviations above the mean of the scores for every chain computed in the document.

2.1 Linear Runtime Proof

In this analysis, we will not consider the computational complexity of part of speech tagging, since it is quite fast. The runtime of the full algorithm will be O(pos tagging) + O(our algorithm). Also, as it does not change from execution to execution of the algorithm, we shall take the size and structure of WordNet to be constant. We will examine each phase of our algorithm to show that the extraction of these lexical chains can indeed be done in linear time. Figure 4 defines constants for this analysis.

2.1.1 Collection of WordNet information For each noun in the source document that appears in WordNet, each sense that the word can take must be examined. Additionally, for each sense, we must walk up and down the hypernym/hyponym graph collecting all parent and child information. It is important to note that we are not only interested in direct parents and children, but in all parents and children in the graph from most spe-

Value	Worst Case	Average Case
C_1=# of senses for a given word	30	2
C_2=parent/child "is a" relations of a word sense	45147	14
C_3=# of nouns in WordNet	94474	94474
C_4=# of synsets in WordNet	66025	66025
C_5=# of siblings of a word sense	397	39
C_6=# of chains a word instance can belong to	45474	55

Figure 4
Constants from WordNet 1.6

cific to most general. Lastly we must collect all of the senses in WordNet that are siblings (i.e. share immediate parents) with the word being processed. All of the complexity in this step is related to the size of WordNet which is constant. Lookups in WordNet use a binary search, hence a search in WordNet is $O(\log(C_3))$. The run-time is given as follows:

$$n*(\log_2(C_3)+C_1*C_2+C_1*C_5)$$

2.1.2 Building the graph The graph of all possible interpretations is nothing more than an array of sense values (66025+n in size) which we will call the sense array. For each word, we examine each relation computed as above from WordNet. For each of these relations, we modify the list that is indexed in the sense array by the sense number of the noun's sense involved in the relation. This list is modified by adding the word to the list, and updating the list's associated score. Additionally, we add the chain's pointer (stored in the array) to a list of such pointers in the word object. Lastly, we add the value of how this word affects the score of the chain based on the scoring system to an array stored within the word structure. The runtime for this phase of the algorithm is:

$$n*C_6*4$$

which is also clearly O(n).

2.1.3 Extracting the Best Interpretation For each word in the source document, we look at each chain to which the word can belong. A list of pointers to these chains are stored within the word object, so looking them up takes O(1) time. For each of these, we simply look at the maximal score component value in all of these chains. We then set the scores of all of the nodes that did not contain the maximum to 0 and update all the chain scores appropriately. The operation takes

$$n*C_6*4$$

which is also O(n).

2.1.4 Overall Run Time Performance The overall runtime performance of this algorithm is given by the sum of the steps listed above for an overall runtime of:

$$n*(\ 1548216+\log_2(94474)+45474*4) \text{ worst case.}$$

$$n*(326+\log_2(94474)+55*4) \text{ average case.}$$

Initially, we may be greatly concerned with the size of these constants; however, upon further analysis, we see that most synsets have very few parent child relations. Thus the worst case values may not reflect the actual performance of our application. In addition, the synsets with many parent child relations tend to represent extremely general concepts such as “thing” and “object”. These synsets will most likely not appear very often in a document.

While in the worst case these constants are quite large, in the average case they are reasonable. This algorithm is O(n) in the number of nouns within the source document. Considering the size of most documents, the linear nature of this algorithm makes it usable for generalized summarization of large documents (Silber and McCoy, 2000). For example, in a test, our algorithm has calculated a lexical chain interpretation of a 40,000 word document in 11 seconds on a Sparc Ultra 10 Creator. It was impossible to compute lexical chains for such documents under previous implementations because of computational complexity. Thus documents tested by Barzilay and Elhadad were significantly smaller in size. Our method affords a considerable speed up for these smaller documents. For instance, a document that takes 300 seconds using Barzilay and Elhadad’s method, takes but 4 seconds using ours (Silber and McCoy, 2000).

3 Evaluation Design

Our algorithm now makes it feasible to use lexical chains as the method for identifying important concepts in a document and thus form the basis of an intermediate representation for summary generation as proposed by Barzilay and Elhadad. An important consequence of this is to evaluate this proposal on documents of substantial size. We propose an evaluation of this intermediate stage that is independent of the generation phase of summarization. This said, the authors attempt to make no claim that a summary can actually be generated from this representation, however, we attempt to show that the concepts found in a human generated summary are indeed the concepts identified by our lexical chains algorithm.

The basis of our evaluation is the premise that if lexical chains are a good intermediate representation, then we would expect that if we had a summary, each noun in the summary should be used in the same sense as some word instance grouped into a strong chain in the original document. Moreover, we expect that all (most) strong chains in the document should be represented in the summary.

For this analysis, a corpus of documents with their human generated summaries are required. While there are many examples of document and summary types, for the purposes of this experiment, we focus on two general categories of summaries which are readily available. The first, scientific documents with abstracts, represent a readily available class of summaries often discussed in the literature (Marcu, 1999). The second class of document selected was chapters from University level textbooks that contain chapter summaries. To remove bias, textbooks from several fields were chosen.

In this analysis, we use the term “concept” to denote a noun in a particular sense (a given sense number in the WordNet database). Note, different nouns with the same sense number[4] are considered to be the same concept. It is important to note that for the purposes of this analysis, when we say sense of a word, we mean the sense as determined by our lexical chains analysis. The basic idea of our experiment is to try to determine if the concepts represented by (strong) lexical chains in the original document appear in the summary and if the concepts appearing in the summary (as determined by the lexical chain analysis of the summary) come from strong chains in the document. If both of these give 100% coverage this would mean that all and only the concepts iden-

4 Recall that synonyms in the WordNet database are identified by a synset (sense) number.

tified by strong lexical chains in the document occur in the summary. Thus the higher these numbers turn out to be, the more likely it is that lexical chains are a good intermediate representation of the text summarization task.

A corpus was compiled containing the two specific types of documents ranging in length from 2247 to 26320 words. These documents were selected at random, with no pre-screening by the authors. The scientific corpus consisted of ten scientific articles (five Computer Science, three Anthropology, and two Biology) along with their abstracts. The textbook corpus consisted of 14 chapters from 10 University level textbooks in various subjects (4 Computer Science, 6 Anthropology, 2 History, and 2 Economics) including chapter summaries.

For each document in the corpus, the document and its summary were analyzed separately to produce lexical chains. In both cases we output the sense numbers specified for each word instance as well as the overriding sense number for each chain. By comparing the sense numbers of (words in) each chain in the document with the computed sense of each noun instance in the summary, we can determine if the summary indeed contains the same "concepts" as indicated by the lexical chains. For the analysis, the specific metrics we are interested in are:

- The number and percentage of strong chains from the original text that are represented in the summary. Here we say a chain is represented if a word occurs in the summary in the same sense as in the document strong chain. (Analogous to recall)
- The number and percentage of noun instances in the summary that represent strong chains in the document. (Analogous to precision)

By analyzing these two metrics, we can determine how well lexical chains represent the information that appears in these types of human generated summaries. We will loosely use the terms recall and precision to describe these two metrics.

3.1 Experimental Results

Each document in the corpus was analyzed by running our lexical chains algorithm and collecting the overriding sense number of each strong lexical chain computed. Each summary in the corpus was analyzed by our algorithm and the disambiguated sense of each noun was collected[5]. Figure 5 shows the results of this analysis. The number of strong chains computed for the document is shown in column 2. Column 3 shows the total number of noun instances found in the summary. Column 4 shows the number, and percentage overall, of strong chains from the document that are represented by noun instances in the summary (recall). The number, and the percentage overall, of nouns of a given sense from the summary that have a corresponding strong chain with the same overriding sense number (representing the chain) in the original text are presented in column 5 (precision). Summary statistics are also presented.

In 79.12% of the cases, lexical chains appropriately represent the nouns in the summary. In 80.83% of the cases, nouns in the summary would have been predicted by the lexical chains. Two documents, ANTH Paper 3 and CS Chapter 4, perform badly under this analysis. Possible reasons for this will be discussed below, but our preliminary analysis of these documents leads us to believe that they contain a greater number of pronouns and other anaphoric expressions (which need to be resolved to properly compute lexical chains). These questions need to be examined further to determine why

5 This is the sense of the noun instance that was selected in order to insert it into a chain.

Document	Total Number of Strong Chains in document	Total Number of noun instances in summary	Strong chains with corresponding noun instances in summary	Noun instances with corresponding strong chains in document
CS Paper 1	10	22	7 (70.00%)	19 (86.36%)
CS Paper 2	7	19	6 (71.43%)	17 (89.47%)
CS Paper 3	5	31	4 (80.00%)	27 (87.19%)
CS Paper 4	6	25	5 (83.33%)	24 (96.00%)
CS Paper 5	8	16	6 (75.00%)	12 (75.00%)
ANTH Paper 1	7	20	7 (100.00%)	17 (85.00%)
ANTH Paper 2	5	17	4 (80.00%)	13 (76.47%)
ANTH Paper 3	7	21	6 (28.57%)	7 (33.33%)
BIO Paper 1	4	19	4 (100.00%)	17 (89.47%)
BIO Paper 2	5	31	5 (80.00%)	28 (90.32%)
CS Chapter 1	9	55	8 (88.89%)	49 (89.09%)
CS Chapter 2	7	49	6 (85.71%)	42 (85.71%)
CS Chapter 3	11	31	9 (81.82%)	25 (80.65%)
CS Chapter 4	14	47	5 (35.71%)	21 (44.68%)
ANTH Chapter 1	5	61	4 (80.00%)	47 (77.05%)
ANTH Chapter 2	8	74	7 (87.50%)	59 (79.73%)
ANTH Chapter 3	12	58	11 (91.67%)	48 (82.76%)
ANTH Chapter 4	13	49	11 (84.62%)	42 (85.71%)
ANTH Chapter 5	7	68	5 (71.43%)	60 (88.24%)
ANTH Chapter 6	9	59	8 (88.89%)	48 (81.36%)
HIST Chapter 1	12	71	10 (83.33%)	67 (94.37%)
HIST Chapter 2	8	65	7 (87.50%)	55 (84.62%)
ECON Chapter 1	14	68	12 (85.71%)	63 (92.65%)
ECON Chapter 2	9	51	7 (77.78%)	33 (64.71%)
Mean			79.12%	80.83%
Median			82.58%	85.35%

Figure 5
Analysis Results for each document.

these documents perform so poorly under our analysis. Without these two documents, our algorithm has a recall of 83.39% and a precision of 84.63% on average. It is important to note that strong chains only represent between 5% and 15% of the total chains computed for the document.

The evaluation presented here would be enhanced by having a baseline for comparison. It is not clear, however, what this baseline should be. One possibility would be to use straight frequency counts as an indicator and use these values for comparison.

4 Discussion and Future Work

Some problems which cause our algorithm to have difficulty, specifically proper nouns and anaphora resolution, need to be addressed. Proper nouns are often used in naturally occurring text, but since we have no information about them (People, Organization, Company, etc.), we can only do frequency counts on them. Anaphora resolution, especially in certain domains, is a bigger issue. Much better results are anticipated with the addition of anaphora resolution to the system.

Other issues which may affect the results stem from the WordNet coverage and the semantic information it captures. Clearly, no semantically annotated lexicon can be complete. Proper nouns, domain specific terms, as well as a number of other words likely to be in a document are not found in the WordNet database. The system defaults to word frequency counts for terms not found. Semantic distance in the "isa" graph, a problem in WordNet, does not affect our implementation, since we don't use this information. It is important to note that while our system used WordNet, there is nothing specific about WordNet per se, and any other appropriate lexicon could be "plugged in" and used.

Issues regarding generation of a summary based on lexical chains needs to be addressed and are the subject of our current work. Recent research has begun to look at the difficult problem of generating a summary text from an intermediate representation. Hybrid approaches such as extracting phrases instead of sentences and recombining these phrases into salient text have been proposed (Barzilay, McKeown, and Elhadad, 1999). Other recent work looks at summarization as a process of revision, where the source text is revised until the desired length summary is achieved (Mani, Gates, and Bloedorn, 1999). Additionally, some research has explored cutting and pasting segments of text from the full document to generate a summary (Jing and McKeown, 2000). It is our intention to use lexical chains as part of the input to a more classical text generation algorithm to produce new text which captures the concepts from the extracted chains. The lexical chains identify noun (or argument) concepts for the summary. We are examining ways for predicates to be identified and are concentrating on situations where strong lexical chains intersect in the text.

5 Conclusions

In this paper, we have outlined an efficient, linear time algorithm for computing lexical chains as an intermediate representation for automatic machine text summarization. This algorithm is robust in that it uses the method proposed by Barzilay and Elhadad, but it is clearly O(n) in the number of nouns in the source document.

The benefit of the linear time algorithm is its ability to compute lexical chains in documents significantly larger than could be handled by Barzilay and Elhadad's implementation. Thus, our algorithm makes lexical chains a computationally feasible intermediate representation for summarization. In addition, we have presented a method for evaluating lexical chains as an intermediate representation and have evaluated the method using 24 documents which contain human generated summaries. The results of these evaluations are promising.

An operational sample of our algorithm is available on the web; a search engine that uses our algorithm can be accessed there as well (Available at http://www.eecis.udel.edu/~silber/research.htm).

Acknowledgments

The authors wish to thank the Korean Government, Ministry of Science and Technology, whose funding, as part of the Bilingual Internet Search Machine Project, has made this research possible. Additionally, special thanks to Michael Elhadad and Regina Barzilay for their advice, and for generously making their data and results available, and to the anonymous reviewers for their helpful comments.

References

Barzilay, Regina and Michael Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop*, Madrid, Spain. ISTS'97.

Barzilay, Regina, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Conference of the Association for*

Computational Linguistics, College Park, MD. ACL.

Hirst, Gramme and David St-Onge. 1997. Lexical chains as representation of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *Wordnet: An electronic lexical database and some of its applications*. The MIT Press, Cambridge, MA.

Jing, H. and K. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of NAACL'00*, Seattle, WA.

Mani, Inderjeet, Barbara Gates, and Eric Bloedorn. 1999. Improving summaries by revising them. In *Proceedings of the 37th Annual Conference of the Association for Computational Linguistics*, College Park, MD. ACL.

Marcu, Daniel. 1999. The automatic creation of large scale corpora for summarization research. In *The 22nd international ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkley, CA. ACM.

Morris, J. and G. Hirst. 1991. Lexical cohesion computed by thesaural relations as an indecator of the structure of text. *Computational Linguistics*, 18:21–45.

Silber, H. Gregory and Kathleen F. McCoy. 2000. Efficient text summarization using lexical chains. In *2000 International Conference on Intelligent User Interfaces*, New Orleans, LA, January.

Sparck Jones, Karen. 1993. What might be in summary? *Information Retrieval*, pages 9–26.