Efficient Text Summarization Using Lexical Chains

H. Gregory Silber
Computer and Information Sciences
University of Delaware
Newark, DE 19711
silber@ udel.edu

Kathleen F. McCoy
Computer and Information Sciences
University of Delaware
Newark, DE 19711
mccoy@cis.udel.edu

ABSTRACT

The rapid growth of the Internet has resulted in enormous amounts of information that has become more difficult to access efficiently. Internet users require tools to help manage this vast quantity of information. The primary goal of this research is to create an efficient and effective tool that is able to summarize large documents quickly. This research presents a linear time algorithm for calculating lexical chains which is a method of capturing the "aboutness" of a document. This method is compared to previous, less efficient methods of lexical chain extraction. We also provide alternative methods for extracting and scoring lexical chains. We show that our method provides similar results to previous research, but is substantially more efficient. This efficiency is necessary in Internet search applications where many large documents may need to be summarized at once, and where the response time to the end user is extremely important.

Keywords

Summarization, NLP, lexical chains, cohesion, linguistics, algorithm

1. INTRODUCTION

1.1 Motivation

Automatic text summarization has received a great deal of attention in recent research. The rapid growth of the Internet has resulted in enormous amounts of information that has become increasingly more difficult to access efficiently. The ability to summarize information automatically and present results to the end user in a compressed, yet complete form, would help to solve this problem. Further, for a proposed Internet application, efficiency, even on large documents, is of substantial importance.

1.2 Background Research

Current research in automatic text summarization has generally viewed the process of summarization in two steps. The first step of the summarization process is to extract the important concepts from the source text into some form of intermediate representation. The second step is to use the intermediate representation to generate a coherent summary of the source document [4].

Many methods have been proposed to extract the important concepts from a source text and to build the intermediate representation. Early methods were primarily statistical in nature and focused on word frequency to determine the most important concepts within a document [5].

The opposite extreme of such statistical approaches is to attempt true "semantic understanding" of the source document. Obviously the use of deep semantic analysis offers the best opportunity to create a quality summary. The problem with such approaches is that a detailed semantic representation must be created and a domain specific knowledge base must be available.

The major problem with purely statistical methods is that they do not account for context. Specifically, finding the aboutness of a document relies largely on identifying and capturing the existence of not just duplicate terms, but related terms as well. This concept, known as cohesion, links semantically related terms which is an important component in a coherent text [2].

The simplest form of cohesion is lexical cohesion. Morris and Hirst first introduced the concept of lexical chains [6]. Lexical chains represent the lexical cohesion among an arbitrary number of related words. Lexical chains can be recognized by identifying sets of words that are semantically related (i.e. have a sense flow). Using lexical chains in text summarization is efficient, because these relations are easily identifiable within the source text, and vast knowledge bases are not necessary for computation. By using lexical chains, we can statistically find the most important concepts by looking at structure in the document rather than deep semantic meaning. All that is required to calculate these is a generic knowledge base that contains nouns, and their associations. These associations capture concept relations such as synonym, antonym, and hyperonym (isa relations).

Barzilay and Elhadad have noted limitations in previous implementations of lexical chains. Because all possible senses of the word are not taken into account, except at the time of insertion, potentially pertinent context information that appears after the word is lost. The problem that results is referred to as "greedy disambiguation" [1]. Barzilay and Elhadad presented a less greedy algorithm that constructs all possible interpretations of the source text using lexical chains. Their algorithm then selects the interpretation with the strongest cohesion. They then use these "strong chains" to generate a summary of the original document. They also present an examination of the usefulness of these lexical chains as a source representation for automatic text summarization [1]. Barzilay and Elahadad used Wordnet as their knowledge base. Wordnet is a lexical database which captures all senses of a word and contains semantic information about the relations between words. The algorithm first segments the text, then for each noun in the segment, for each sense of the noun, it attempts to merge these senses into all of the existing chains in every possible way, hence building every possible interpretation of the segment. Next, the algorithm merges chains between segments that contain a word in the same sense in common. The algorithm then selects the chains denoted as "strong" (more than two standard deviations above the mean) and uses these to generate a summary.

This research defines a linear time algorithm for computing lexical chains based on the work of Barzilay and Elhadad.

2. A LINEAR TIME ALGORITHM FOR COMPUTING LEXICAL CHAINS

2.1 Research Design

The issue of time complexity arises if an algorithm such as Barzilay and Elhadad's is to be used effectively for information retrieval. This research differs in focus from that of Barzilay and Elhadad in that our major focus is on efficiency, while their focus was on the feasibility of such methodologies. We take this approach because our goal is to provide an efficient means of summarization for internet material that can still produce superior results. Additionally, in extending their research, care was taken to examine the effect of the pruning step that is not required by our algorithm. Lastly, a more complex scoring algorithm, which is machine trainable, was devised to allow more detailed research into the relative importance of different types of relations between words in the source text.

2.2 Issues related to Wordnet

Wordnet is a lexical database that contains substantial semantic information. In order to facilitate efficient access, the Wordnet noun database and tools were rewritten. The result of this work is that accesses to the Wordnet noun database can be accomplished an order of magnitude faster than with the original implementation.

2.3 The Algorithm

Our algorithm attempts to implement the methods of Barzilay and Elhadad, as well as an extended scoring system that we propose. The segmentation of the text is currently implemented to allow comparison; however, it does not run in linear time. The algorithm is described in detail in figure 1.

1) Tag the corpus using semantag. Semantag is a Brill style Part of Speech Tagger designed for efficiency available from http://www.rt66.com/gcooke/SemanTag on the Internet.
2) For each noun in the source document, form all possible lexical chains by looking up all relation information including synonyms, hyponyms, hypernyms, and siblings. This information is stored in an array indexed on the index position of the word from Wordnet for constant time retrieval.
3) For each noun in the source document, use the information collected by the previous step to insert the word in each "meta chain". A "meta chain" is so named, because it represents all possible chains whose beginning word has a given sense number. Meta-chains are stored by sense number. The Sense numbers are now zero based due to our reindexing of Wordnet. Again, the implementation details are important, as they allow us to retrieve the meta-chain in constant time.

Figure 1 – Linear time algorithm for computing lexical chains

This first phase of our implementation constructs an array of "meta chains." Each meta chain contains a score and a data structure which encapsulates the meta-chain. The score is computed as each word is inserted into the chain. While the implementation creates a flat representation of the source text, all interpretations of the source text are implicit within the structure. This concept is illustrated in Figure 3. Each dot represents a sense of a word in the document. Each line represents a semantic connection between two word senses. Each set of connected dots and lines represents a meta-chain. The gray ovals represent the list of chains to which a word can belong. The dashed box indicates the strongest chain in our representation.

Notice that in some senses of the word machine, it is semantically similar to friend, while in other senses, it is semantically similar to computer (i.e. in the same meta-chain). The algorithm continues by attempting to find the "best" interpretation from within our flat representation. We view the representation as a set of transitively closed graphs whose vertices are shared. In figure 3, the sets of lines and dots represent five such graphs. The set of dots within an oval represent a single shared node. That is to say, that while two of these graphs may share a node, the individual graphs are not connected. The "best" interpretation will be the set of graphs that can be created from the initial set mentioned above, by deleting nodes from each of the graphs so that no two graphs share a node, and the overall "score" of all the meta-chains is maximal.

```
1) For each word in the document
   a) For each chain that the word belongs to.
      i)  Find the chain whose score will be affected most
          greatly by removing this word from it.
      ii) Set the score component of this word in each of the
          other chains to which it belongs to 0, and update the
          score of all the chains to which the word belongs to
          reflect the word's removal.
```

Figure 2: Computation of best chain

The computation of the best interpretation is shown in figure 2.

With this method, we can find the set of chains which maximize the overall score without actually having to construct them all

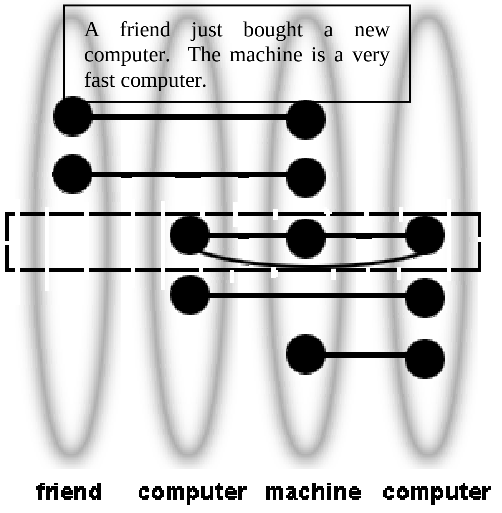
A friend just bought a new computer. The machine is a very fast computer.
friend
computer
machine
computer

Figure 3: Implicit interpretations within our representation.

explicitly. This fact is really the most important concept of this research. The fact that we can extract the interpretation (independent set of non-intersecting chains) of the text with the highest score without actually having to construct any other interpretations is the insight that allows this algorithm to run in linear time.

2.4 Runtime Analysis

In this analysis, we will not consider the computational complexity of part of speech tagging, since that is not the focus of this research. 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. For this analysis, we define constants in Table 1. 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. These synsets will most likely not appear very often as a direct synset for words appearing in a document.

Value	**Wrst**	**Avg**
C_1=# of senses	30	2
C_2=parent/child “is a” relations	45147	14
C_3=# of nouns in Wordnet	94474	94474
C_4=# of synsets in Wordnet	66025	66025
C_5=# of siblings	397	39
C_6=# of chains a word can belong to	45474	55

Table 1: Constants from Wordnet

2.4.1 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. Lastly we must collect all of the senses in Wordnet which share immediate parents with the word in question. All of the complexity in this step is related to the size of Wordnet which is constant. The run-time is given by the formula:

$$n*(\log(C_3)+C_1*C_2+C_1*C_5).$$

2.4.2 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 said relation. This list is modified by adding the word to the list, and updating the lists associated score. Additionally, we add the chains pointer (value stored in the array) to a list of such pointers in the word object. Lastly, we add the value of how this word effects the score of the chain based on the scoring system to an array stored within the word structure.

Clearly, because of the implementation all but the computing of the score component of the word are O(1). Because we also keep an array of sentences within each chain object with the words organized by their sentence number, we can easily find whether a word is within a certain distance of any other word in the chain in O(window size) time. The window size defaults, while adjustable, are taken as a constant and are generally small.

Consequently, the runtime for this phase of the algorithm is:

$$n*C_6*(4+\text{window size}) \text{ which is also clearly } O(n).$$

2.4.3 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).

2.4.4 Overall Run Time Performance

The overall runtime performance of this algorithm is given by the sum of the steps listed above. These steps give us an overall runtime of:

$$n * (1548216 + \log(94474) + 45474 * (4 + \text{window size})) \text{ worst case}$$

or

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

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.

3. EXPERIMENTS

3.1 Experimental Design

Experiments were conducted with the following research questions in mind. Does our linear time algorithm perform comparably with existing algorithms for computing lexical chains? How does a more complex scoring algorithm effect summarization?

These experiments were carried out on documents selected at random from the original set of documents tested by Barzilay and Elhadad.

3.2 Experimental Results

The results compared were strong chains (two standard deviations above the mean of the chains’ scores). Metrics were used to score chains. It is important to note that while flexible, these metrics were selected based on intuitive reasoning as to how lexical cohesion relationships might work in practice. More work is certainly necessary, and these values are by no means optimal. The values were merely selected to test the alternative scoring system approach. The results showed that although minor differences between results existed, they were relatively insignificant. Results of three selected documents are available online at http://www.eecis.udel.edu/~silber/results.htm.

4. DISCUSSION

4.1 Analysis of Our Algorithm

The experiments were conducted with the intention of determining how well our algorithm duplicates the experimental results of Barzilay and Elhadad. In conducting such an analysis, we must consider the known differences in our algorithms. The first, and possibly most apparent difference in our algorithms, is in the detection of noun phrase collocations. The algorithm presented by Barzilay and Elhadad uses a shallow grammar parser to detect such collocations in the source text prior to processing [1]. Our algorithm simply uses word compounds appearing in Wordnet (Wordnet stores such words connected by an underscore character). This difference may account for some of the differences observed in the results.

The next inherent difference between the algorithms is that Barzilay and Elhadad attempt to process proper nouns which our algorithm does not address. Although not clear how it is done, Barzilay and Elhadad do some processing to determine relations between proper nouns, and their semantic meanings.

Upon analysis, these differences seem to account for most of the differences between the results of our algorithm with segmentation, and the algorithm of Barzilay and Elhadad.

5. CONCLUSIONS

In this paper, we have outlined an efficient algorithm for computing lexical chains as an intermediate representation for automatic machine text summarization. In addition, several issues that affect efficiency were discussed.

The algorithm presented is clearly O(n) in the number of nouns present within the source document. While in the worst case, the constants that arise from this analysis are quite large, in the average case, the constants are manageable, and in practice, they are quite small. In tests conducted on a Sun Sparc Ultra10 Creator, a 40,000 word corpus was summarized in eleven seconds including generation.

Careful experiments reveal that our efficient method of text summarization produces similar results to the algorithm of Barzilay and Elhadad. Most of the inconsistencies between the output of the two algorithms can be attributed to differences in the ancillary components of the two summarization systems.

While not a major issue our algorithm does not prune the intermediate representation during its construction, and thus our algorithm provides the generation algorithm (in our case, a sentence extraction algorithm) with a more complete picture of the source text. In future work on the generation aspects of summarization, this factor may become more important.

An alternative scoring system to the one proposed by Barzilay and Elhadad was devised. This scoring system, while not currently optimized, provides good results, and allows the user to tune the importance of different types of relations, which in turn affect the summary. This scoring system also provides a notion of distance which allows for adjusting scores based on the size of "gaps" within the chains.

In their research, Barzilay and Elhadad showed that lexical chains could be an effective tool for automatic text summarization. By developing a linear time algorithm to compute these chains, we have produced a front end to a summarization system which can be implemented efficiently. An internet interface was developed to convert HTML documents into input to the summarizer. An operational sample of the summarizer is currently available on the World Wide Web for testing at http://www.eecis.udel.edu/~silber/research.htm.

6. FUTURE WORK

As this is ongoing research, there are many aspects of our work that have yet to be addressed. Issues regarding the extraction of lexical chains, segmentation, scoring, and eventual generation of the summary text must be examined further.

Segmentation, as implemented by Barzilay and Elhadad, is inefficient. It may be possible to incorporate segmentation information by making the distance metric of our new scoring system dynamic. By using segmentation information to determine the distance metric, we may be able to take advantage of segmentation without the expense of merging together chains computed from individual segments [1].

Examinations of the performance of our algorithm on larger documents should be conducted. Moreover, further analyses on the effects of pruning, as required by Barzilay and Elahadad, on these larger documents are also warranted

The scoring system proposed in this research requires optimization. Currently, its values are set based on the linguistic intuition of the authors. In future work, we hope to use machine learning techniques to train these values from human-created summaries.

Lastly, experiments to evaluate the effectiveness of a summary must be conducted. These experiments are necessary to examine how well our summary can assist a user in making a decision or performing a task. Since no two people would summarize the same document in precisely the same way, evaluation is one of the most difficult parts of text summarization.

7. 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.

8. REFERENCES

[1] Barzilay, Regina and Michael Elhadad. Using Lexical Chains for Text Summarization. in Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL Madrid, 1997.

[2] Halliday, Michael and Ruqaiya Hasan. Cohesion in English. Longman, London, 1976.

[3] Hearst, Marti A. Multi-paragraph segmentation of expository text. In Proceedings of the 32nd Annual Meeting of the ACL. 1994

[4] Jones, Karen Sparck. What might be in summary? Information Retrieval, 1993.

[5] Luhn, H.P. The automatic creation of literature abstracts. In H.P. Luhn: Pioneer of Information Science. Schultz, editor. Spartan, 1968.

[6] Morris, J. and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of the text. In Computational Linguistics, 18(1):pp21-45. 1991.

[7] Stairmond, Mark A. A Computational Analysis of Lexical Cohesion with Applications in Information Retrieval. Ph.D. thesis, Center for Computational Linguistics, UMIST, Manchester, 1999.