

An Efficient Text Summarizer Using Lexical Chains

H. Gregory Silber and Kathleen F. McCoy

Computer and information Sciences

University of Delaware

Newark, DE 19711

{*silber, mccoy*}@cis.udel.edu

Abstract

We present a system which uses lexical chains as an intermediate representation for automatic text summarization. This system builds on previous research by implementing a lexical chain extraction algorithm in linear time. The system is reasonably domain independent and takes as input any text or HTML document. The system outputs a short summary based on the most salient concepts from the original document. The length of the extracted summary can be either controlled automatically, or manually based on length or percentage of compression. While still under development, the system provides useful summaries which compare well in information content to human generated summaries. Additionally, the system provides a robust test bed for future summary generation research.

1 Introduction

Automatic text summarization has long been viewed as a two-step process. First, an intermediate representation of the summary must be created. Second, a natural language representation of the summary must be generated using the intermediate representation (Sparck Jones, 1993). Much of the early research in automatic text summarization has involved generation of the intermediate representation. The natural language generation problem has only recently received substantial attention in the context of summarization.

1.1 Motivation

In order to consider methods for generating natural text summaries from large documents, several issues must be examined in detail. First, an analysis of the quality of the intermediate representation for use in generation must be examined. Second, a detailed examination of the processes which link the intermediate representation to a potential final summary must be undertaken.

The system presented here provides a useful first step towards these ends. By developing a robust and efficient tool to generate these intermediate representations, we can both evaluate the representation

and consider the difficult problem of generating natural language texts from the representation.

1.2 Background Research

Much research has been conducted in the area of automatic text summarization. Specifically, research using lexical chains and related techniques has received much attention.

Early methods using word frequency counts did not consider the relations between similar words. Finding the aboutness of a document requires finding these relations. How these relations occur within a document is referred to as cohesion (Halliday and Hasan, 1976). First introduced by Morris and Hirst (1991), lexical chains represent lexical cohesion among related terms within a corpus. These relations can be recognized by identifying arbitrary size sets of words which are semantically related (i.e., have a sense flow). These lexical chains provide an interesting method for summarization because their recognition is easy within the source text and vast knowledge sources are not required in order to compute them.

Later work using lexical chains was conducted by Hirst and St-Onge (1997) using lexical chains to correct malapropisms. They used WordNet, a lexical database which contains some semantic information (<http://www.cs.princeton.edu/wn>).

Also using WordNet in their implementation, Barzilay and Elhadad (1997) dealt with some of the limitations in Hirst and St-Onge's algorithm by examining every possible lexical chain which could be computed, not just those possible at a given point in the text. That is to say, while Hirst and St-Onge would compute the chain in which a word should be placed when a word was first encountered, Barzilay and Elhadad computed every possible chain a word could become a member of when the word was encountered, and later determined the best interpretation.

2 A Linear Time Algorithm for Computing Lexical Chains

2.1 Overview

Our research on lexical chains as an intermediate representation for automatic text summarization follows the research of Barzilay and Elhadad (1997). We use their results as a basis for the utility of the methodology. The most substantial difference is that Barzilay and Elhadad create all possible chains explicitly and then choose the best possible chain, whereas we compute them implicitly.

2.2 Modifications to WordNet

As mentioned above, WordNet is a lexical database that contains substantial semantic information. In order to facilitate efficient access, the WordNet noun database was re-indexed by line number as opposed to file position and the file was saved in a binary indexed format. The database access tools were then rewritten to take advantage of this new structure. 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. No additional changes to the WordNet databases were made. The re-indexing also provided a zero-based continuous numbering scheme that is important to our linear time algorithm. This importance will be noted below.

2.3 Our Algorithm

| | |
|--------|---|
| Step 1 | For each word instance that is a noun For every sense of that word Compute all scored “meta-chains” |
| Step 2 | For each word instance Figure out which “meta-chain” it contributes most to Keep the word instance in that chain and remove it from all other Chains updating the scores of each “meta-chain” |

Figure 1: Basic linear time Algorithm for Computing Lexical Chains

Our basic lexical chain algorithm is described briefly in Figure 1. The algorithm takes a part of speech tagged corpus and extracts the nouns. Using WordNet to collect sense information for each of these noun instances, the algorithm then computes scored “meta-chains” based on the collected information. A “meta-chain” is a representation of every possible lexical chain that can be computed starting with a word of a given sense. These meta-chains are scored in the following manner. As each word instance is added, its contribution, which is dependent on the scoring metrics used, is added to the “meta-chain” score. The contribution is then stored within

| | Intra Pgrph. | Intra Segment | Adjacent Segment | Other |
|----------|-----------------|------------------|---------------------|-------|
| Same | 1 | 1 | 1 | 1 |
| Synonym | 1 | 1 | 0 | 0 |
| Hypernym | 1 | 1 | 0 | 0 |
| Hyponym | 1 | 1 | 0 | 0 |
| Sibling | 1 | 0 | 0 | 0 |

Table 1: Dynamic Scoring Metrics Set to Mimic B+E’s Algorithm

the word itself. These scores are dynamic and can be set based on segmentation information, distance, and type of relation.

Currently, segmentation is accomplished prior to using our algorithm by executing Hearst’s text tiler (Hearst, 1994). The sentence numbers of each segment boundary are stored for use by our algorithm. These sentence numbers are used in conjunction with relation type as keys into a table of potential scores. Table 1 denotes sample metrics tuned to simulate the system devised by Barzilay and Elhadad (1997).

At this point, the collection of “meta-chains” contains all possible interpretations of the source document. The problem is that in our final representation, each word instance can exist in only one chain. To figure out which chain is the correct one, each word is examined using the score contribution stored in Step 1 to determine which chain the given word instance contributes to most. By deleting the word instance from all the other chains, a representation where each word instance exists in precisely one chain remains. Consequently, the sum of the scores of all the chains is maximal. This method is analogous to finding a maximal spanning tree in a graph of noun senses. These noun senses are all of the senses of each noun instance in the document.

From this representation, the highest scored chains correspond to the *important concepts* in the original document. These important concepts can be used to generate a summary from the source text. Barzilay and Elhadad use the notion of strong chains (i.e., chains whose scores are in excess of two standard deviations above the mean of all scores) to determine which chains to include in a summary. Our system can use this method, as well as several other methods including percentage compression and number of sentences.

For a more detailed description of our algorithm please consult our previous work (Silber and McCoy, 2000).

2.4 Runtime Analysis

In this analysis, we will not consider the computational complexity of part of speech tagging, as that is not the focus of this research. Also, because the size

| | Worst Case | Average Case |
|-----------------------------------|------------|--------------|
| C_1 =No. of senses | 30 | 2 |
| C_2 =Parent/child isa relations | 45147 | 14 |
| C_3 =No. of nouns in WordNet | 94474 | 94474 |
| C_4 =No. of synsets in WordNet | 66025 | 66025 |
| C_5 =No. of siblings | 397 | 39 |
| C_6 =Chains word can belong to | 45474 | 55 |

Table 2: Constants from WordNet

and structure of WordNet does not change from execution to execution of the algorithm, we shall take these aspects 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 from WordNet 1.6 as denoted in Table 2.

Extracting information from WordNet entails looking up each noun and extracting all synset, Hyponym/Hypernym, and sibling information. The runtime of these lookups over the entire document is:

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

When building the graph of all possible chains, we simply insert the word into all chains where a relation exists, which is clearly bounded by a constant (C_6). The only consideration is the computation of the chain score. Since we store paragraph numbers represented within the chain as well as segment boundaries, we can quickly determine whether the relations are intra-paragraph, intra-segment, or adjacent segment. We then look up the appropriate score contribution from the table of metrics. Therefore, computing the score contribution of a given word is constant. The runtime of building the graph of all possible chains is:

$$n * C_6 * 5$$

Finding the best chain is equally efficient. For each word, each chain to which it belongs is examined. Then, the word is marked as deleted from all but the single chain whose score the word contributes to most. In the case of a tie, the lower sense number from WordNet is used, since this denotes a more general concept. The runtime for this step is:

$$n * C_6 * 4$$

This analysis gives an overall worst case runtime of:

$$n * 1548216 + \log(94474) + 227370$$

and an average case runtime of:

$$n * 326 + \log(94474) + 275$$

While the constants are quite large, the algorithm is clearly $O(n)$ in the number of nouns in the original document.

At first glance, the constants involved seem prohibitively large. Upon further analysis, however, 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.

2.5 User Interface

Our system currently can be used as a command line utility. The arguments allow the user to specify scoring metrics, summary length, and whether or not to search for collocations. Additionally, a web CGI interface has been added as a front end which allows a user to specify not just text documents, but html documents as well, and summarize them from the Internet. Finally, our system has been attached to a search engine. The search engine uses data from existing search engines on the Internet to download and summarize each page from the results. These summaries are then compiled and returned to the user on a single page. The final result is that a search results page is returned with automatically generated summaries.

2.6 Comparison with Previous Work

As mentioned above, this research is based on the work of Barzilay and Elhadad (1997) on lexical chains. Several differences exist between our method and theirs. First and foremost, the linear run-time of our algorithm allows documents to be summarized much faster. Our algorithm can summarize a 40,000 word document in eleven seconds on a Sun SPARC Ultra10 Creator. By comparison, our first version of the algorithm which computed lexical chains by building every possible interpretation like Barzilay and Elhadad took six minutes to extract chains from 5,000 word documents.

The linear nature of our algorithm also has several other advantages. Since our algorithm is also linear in space requirements, we can consider all possible chains. Barzilay and Elhadad had to prune interpretations (and thus chains) which did not seem promising. Our algorithm does not require pruning of chains.

Our algorithm also allows us to analyze the importance of segmentation. Barzilay and Elhadad used segmentation to reduce the complexity of the problem of extracting chains. They basically built chains within a segment and combined these chains later when chains across segment boundaries shared a word in the same sense in common. While we include segmentation information in our algorithm, it

is merely because it might prove useful in disambiguating chains. The fact that we can use it or not allows our algorithm to test the importance of segmentation to proper word sense disambiguation. It is important to note that on short documents, like those analyzed by Barzilay and Elhadad, segmentation appears to have little effect. There is some linguistic justification for this fact. Segmentation is generally computed using word frequencies, and our lexical chains algorithm generally captures the same type of information. On longer documents, our research has shown segmentation to have a much greater effect.

3 Current Research and Future Directions

Some issues which are not currently addressed by this research are proper name disambiguation and anaphora resolution. Further, while we attempt to locate two-word collocations using WordNet, a more robust collocation extraction technique is warranted.

One of the goals of this research is to eventually create a system which generates natural language summaries. Currently, the system uses sentence selection as its method of generation. It is our contention that regardless of how well an algorithm for extracting sentences may be, it cannot possibly create quality summaries. It seems obvious that sentence selection will not create fluent, coherent text. Further, our research shows that completeness is a problem. Because information extraction is only at the sentence boundary, information which may be very important may be left out if a highly compressed summary is required.

Our current research is examining methods of using all of the important sentences determined by our lexical chains algorithm as a basis for a generation system. Our intent is to use the lexical chains algorithm to determine what to summarize, and then a more classical generation system to present the information as coherent text. The goal is to combine and condense all significant information pertaining to a given concept which can then be used in generation.

4 Conclusions

We have described a domain independent summarization engine which allows for efficient summarization of large documents. The algorithm described is clearly $O(n)$ in the number of nouns in the original document.

In their research, Barzilay and Elhadad showed that lexical chains could be an effective tool for automatic text summarization (Barzilay and Elhadad, 1997). 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 operational sample of this demo is available on the web at <http://www.eecis.udel.edu/~silber/research.htm>.

While usable currently, the system provides a platform for generation research on automatic text summarization by providing an intermediate representation which has been shown to capture important concepts from the source text (Barzilay and Elhadad, 1997). The algorithm's speed and effectiveness allows research into summarization of larger documents. Moreover, its domain independence allows for research into the inherent differences between domains.

5 Acknowledgements

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.

References

- Regina Barzilay and Michael Elhadad. 1997. Using lexical chains for text summarization. *Proceedings of the Intelligent Scalable Text Summarization Workshop, ACL Madrid*.
- Michael Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman, London.
- Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. *Proceedings of the 32nd Annual Meeting of the ACL*.
- Gramme Hirst and David St-Onge. 1997. Lexical chains as representation of context for the detection and correction of malpropisms. *Wordnet: An electronic lexical database and some of its applications*.
- J. Morris and G. Hirst. 1991. Lexical cohesion computed by thesaural relations an an indecator of the structure of text. *Computational Linguistics*, 18:21–45.
- H. Gregory Silber and Kathleen F. McCoy. 2000. Efficient text summarization using lexical chains. *Conference on Intelligent User Interfaces 2000*.