Text Summarization Using Lexical Chains

Meru Brunn Yllias Chali Christopher J. Pinchak
Department of Mathematics and Computer Science
University of Lethbridge
4401 University Drive
Lethbridge, Alberta, Canada, T1K 3M4
E-mail: {brunnm9, chali, pinchak9}@cs.uleth.ca

Abstract

Text summarization addresses both the problem of selecting the most important portions of text and the problem of generating coherent summaries. We present in this paper the summarizer of the University of Lethbridge at DUC 2001, which is based on an efficient use of lexical chains.

1 Introduction

Text summarization addresses both the problem of selecting the most important portions of text and the problem of generating coherent summaries. We describe in this paper the summarizer of the University of Lethbridge at the Document Understanding Conference (DUC) 2001. It is based on an approach for identifying the most important portions of the text which are *topically* most *salient*. This identification also takes into consideration the degree of *connectiveness* among the chosen text portions so as to minimize the danger of producing summaries which contain poorly linked sentences. These objectives can be achieved through an efficient use of *lexical chains*.

The overall architecture of the system is shown in *Figure 1*. It consists of several modules organized as a pipeline. The paper is organized as follows. The next sections are devoted to each of the system's modules. Finally, we conclude by quickly analyzing the performance of the system at DUC evaluation, and outlining some future works.

2 Preprocessing

2.1 Segmentation

To start the summarization process, the original text is first sent to the text segmenter. The role of the text segmenter is to divide the given text into segments that address the same topic. To fulfill the text segmentation requirement, we chose the text segmenter described in (Choi, 2000). This text segmenter typically generated multiple segments for a document, some of which were more substantial in content than others. The purpose of segmentation in our system is to obtain multiple sub-source texts, each containing sentences that discuss the same topic. This segmentation allows later modules to better analyze and generate a summary for a given source text.

2.2 Tagging

The tagging module, which is essential for using the parser, involves classifying words in the segment according to the part of speech they represent. In this process, known as *part-of-speech tagging*, words are considered individually, and no semantic structure is considered or assigned by the tagger. The tagger used in our system is described in (Ratnaparkhi, 1996), and was chosen because of its impressive accuracy in assigning tags. This tagger comes pre-trained on sections 0 to 18 of the Penn Treebank Wall Street Journal corpus [http://www.cis.upenn.edu/~treebank/home.html], and there was no reason to retrain the tagger using a significantly different corpus.

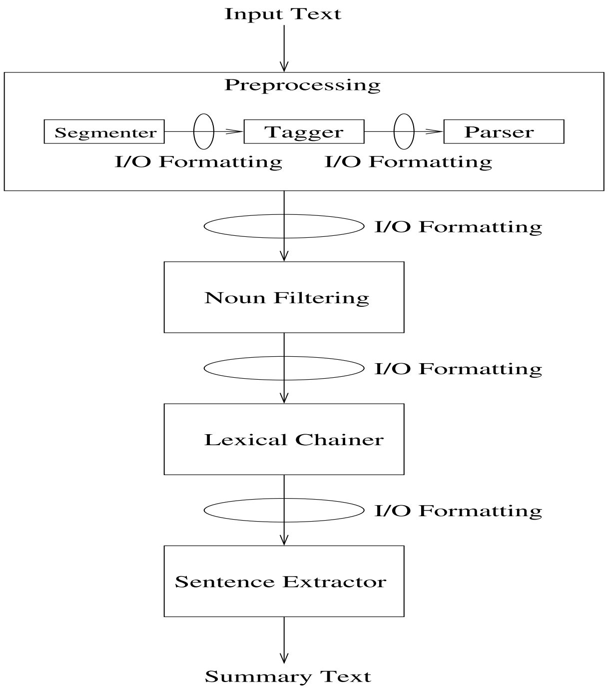

Figure 1: System Overview

Also included in the package containing Ratnaparkhi's MXPOST tagger is a utility for detecting sentence boundaries. This utility is described in (Reynar and Ratnaparkhi, 1997), and is useful for breaking up text such that each sentence appears on a line by itself. The tagger also relies on these sentences being tokenized according to the Penn Treebank conventions, found at [http://www.cis.upenn.edu/˜treebank/tokenization.html]. A small sed script, provided by the University of Pennsylvania, converts plain text to the appropriately tokenized version [http://www.cis.upenn.edu/˜treebank/tokenizer.sed]. These utilities are responsible for converting segments into an acceptable format for the tagger.

2.3 Parsing

Parsing is the most significant and time-consuming of the modules in our system. In this module, tagged words are collected and organized into their syntactic structure. We chose the parser presented in (Collins, 1997) for our system. A parsed representation of the text allows for the selection of various components (or phrases) depending on their syntactic position within a sentence. For example, we could decide to select all noun phrases within a given text. Finding these noun phrases would be a trivial task using a parsed representation of the text. In fact, the next phase of our system relies upon the ability to select particular syntactic elements of sentences.

It must be noted that the parser and tagger are not entirely compatible with respect to input/output. The parser expects each tagged sentence to be preceded by a count of the number of words (tags) within that sentence. Therefore, a small utility was written to count the number of words on a line output by the tagger. The parser also expects that the tagged words will be of the form 'word TAG'. The tagger outputs tagged words with the form 'word_TAG', and so the underscore is simply removed.

3 Noun Filtering

The noun filtering component is an optimization, rather than a required component, of our text summarization system. Noun filtering improves the accuracy of text summarization by selectively removing nouns from the parsed text. A lexical chainer, described in the following section, considers a set of nouns received as input. These nouns come from the source text and are identified by the tagger. However, there are nouns that both contribute to and detract from the subject of the text.

Consider an analogy to analogue data transmission. During data transmission, there is both a signal component and a noise component. Data transmission conditions are ideal when there is a strong signal and low noise. It is when the signal is overcome by noise that it becomes difficult to detect. This is similar to the presence of nouns within the source text. Those nouns that contribute to the subject of the text are part of the 'signal', and those that do not are part of the 'noise'. The noun filter's job is to reduce the 'noise' nouns while still retaining as many 'signal' nouns as possible.

There are a number of different heuristics that could be used to filter out the 'noise' nouns. We have designed a heuristic using the idea that nouns contained within *subordinate clauses* are less useful for topic detection than those contained within *main clauses*. However, these main and subordinate clauses are not easily defined. For our system, we selected a relatively simple heuristic. Because the parser creates the syntactic structure of the sentences, we are able to select different types of phrases from the source text. We chose to identify the first noun phrase and the noun phrase included in the first verb phrase from the first sub-sentence of each sentence as the main clause, with other phrases being subordinate. Results of text summarization using this heuristic can be found in the DUC evaluation section. A number of other, more complex, heuristics could be developed that would seek to identify other phrases as part of the main or subordinate clauses within a sentence. Experimentation will be necessary to determine their benefit to the summary.

4 Lexical chainer

The notion of *cohesion*, introduced by Halliday and Hasan (1976), is a device for 'sticking together' different parts of the text to function as a whole. It is achieved through the use of *grammatical cohesion* (i.e., reference, substitution, ellipsis and conjunction), and *lexical cohesion* (i.e., semantically related words). Lexical cohesion occurs not only between two terms, but among sequences of related words, called *lexical chains* (Morris and Hirst, 1991). The steps of the algorithm for lexical chain computation are as follows:

1. We select the set of candidate words. A candidate word comes from an open class of words that function as a noun phrase or proper name as results of the noun filtering process (cf. section 3).

2. The set of the candidate words are exploded into senses, which are obtained from the thesaurus. In this experiments, we used WordNet thesaurus (Miller et al., 1990). At this step all senses of the word are considered, and each word sense is represented by distinct sets considered as levels. The first one constitutes the set of synonyms and antonyms, the second one constitutes the set of first hypernyms/hyponyms and their variations (i.e., meronyms/holonyms, etc.), and so on.

3. We find the semantic relatedness among the set of senses according to its representations. A semantic relationship exists between two word senses if, when comparing between two sense representations of two distinct words, a matching exists, i.e., a non-empty intersection exists between the sets of words. Each semantic relationship is associated with a measure that indicates the length of the path taken in the matching with respect to the levels of the two compared sets.

4. We build up chains that are sets such as

$$\begin{array}{ll} \{ & (word_1[sense_{11}, sense_{12}, \ldots]), \\ & (word_2[sense_{21}, sense_{22}, \ldots]), \\ & \ldots \qquad\qquad\qquad\qquad\qquad \} \end{array}$$

   in which $word_i$-$sense_{ix}$ is semantically related to $word_j$-$sense_{jy}$ for $i \neq j$

5. We retain the longest chains by relying on the following preference criterion:

$$\begin{array}{l} word\ repetition \gg \\ \quad synonym/antonym \gg \\ \qquad ISA-1/INCLUDE-1 \gg \\ \qquad\quad ISA-2/INCLUDE-2 \gg \\ \qquad\qquad \ldots \end{array}$$

In our implementation, this preference is handled by assigning scores to each pairwise semantical relation in the chain, and then summing those pairwise scores. Hence, the score of a chain is based on its length and on the type of relationships among its members.

In the lexical chaining method, the relationship between the words in a chain is pairwise mutual. That is, each word-sense has to be semantically related to every other word-sense in the chain. The order of the open class words in the document does not play a role in the building of chains. However, it turned out that the number of lexical chains could be extremely large, and thus problematic, for larger segments of text. To cope with this, we reduced the word-sense representation to synonyms only when we had long text segments. This reduction has another benefit, in the sense that a lexical chain based only on synonyms could be better than one based on ISA-2/INCLUDE-2. This reduction also narrows down the set of lexical chains stemming from a single segment in the case where there are too many.

The lexical chainer method integrates a process of word sense disambiguation as the set of candidates are represented by their senses. The preference criteria, stated in step 5 of the algorithm, retain only some chains in which members are senses. Those senses are deemed to be indicators of the cohesion of the text.

Lexical chains are computed for each text segment, and the sentence extractor module proceeds next.

5 Sentence extractor

The sentence extractor module has two steps: segment selection and sentence extraction.

5.1 Segment selection

This phase aims at selecting those text segments that are closely related to the topics suggested by the text. The segment selection algorithm is based on a scoring of segments following the formula:

$$score(seg_j) = \sum_{i=1}^{m} \frac{score(chainMember_i, seg_j)}{s_i}$$

where $score(chainMember_i, seg_j)$ is the number of occurrences of $chainMember_i$ in seg_j, m is their number, and s_i is the number of segments in which $chainMember_i$ occurs.

The top n segments - with the highest scores - are chosen for the process of sentence extraction.

5.2 Sentence extraction

Each sentence is ranked with reference to the total number of lexical cohesion scores collected. The objective of such a ranking process is to assess the importance of each score and to combine all scores into a rank for each sentence. In performing this assessment, provisions are made for a threshold which specifies the minimal number of links required for sentences to be lexically cohesive, following Hoey (1991)'s approach. Ranking a sentence according to this procedure involves summing the lexical cohesion scores associated with the sentence which are above the threshold. According to our empirical investigation, a threshold value of two was retained for our experiments.

Each sentence is ranked by summing the number of shared chain members over the sentence. More precisely, the score for $sentence_i$ is the number of words that belong to $sentence_i$ and also to those chains that have been considered in the segment selection phase.

The summary consists of the ranked list of top-scoring sentences, according to the desired compression ratio, and ordered in accordance with their appearance in the source text.

6 DUC Evaluation

We participated in the single document DUC evaluation. The task consisted of, given a document, creating a generic summary of the document with a length of approximately 100 words. Thirty sets of approximately 10 documents each were provided as system input for this task. According to our analysis, the results seem promising. The grammaticality of our summaries scored an average of 3.73/4. Similarly, the cohesion and organization scores were, on average, of 2.55/4 and 2.66/4, respectively. The evaluation results of our summarizer to identify important information is summarized in Table 1.

number of model units	950
number of peer units	600
number of peer units marked with model units	412
number of unique peer units marked with model units	350

Table 1: Evaluation results

We find that the top ranked sentences are most of the time the most important ones. Furthermore, extraction of sentences that contain referring expressions is still problematic in our system.

7 Conclusions and future work

We have presented in this paper an efficient implementation of the lexical cohesion approach as the driving engine of the summarization system. The ranking procedure, which handles the text 'aboutness' measure, is used to select the most salient and best connected sentences in a text corresponding to the summary ratio requested by the user. In the future, we plan to investigate the following problems:

- Our methods extract whole sentences as single units. The use of compression techniques will increase the condensation of the summary and improve its quality (Barzilay, McKweon, and Elhadad, 1999; Mani, Gates, and Bloedorn, 1999; Jing and McKeown, 2000; Knight and Marcu, 2000).
- Our summarization method uses only lexical chains as representations of the source text. Other clues could be gathered from the text and considered when generating the summary.
- In the noun filtering process, our hypothesis eliminates the terms in subordinate clauses. Rather than eliminating them, it may also prove fruitful to investigate weighting terms according to the kind of clause in which they occur.

Acknowledgments

This work was supported by a research grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada and Research Excellence Envelope (REE) funding from the Alberta Heritage Foundation for Science and Engineering Research (AHFSER). We would like to thank Lesley Schimanski for her helpful comments during the preparation of this paper.

References

Barzilay, Regina and Michael Elhadad. (1997). Using lexical chains for text summarization. In *ACL/EACL Workshop on Intelligent Scalable Text Summarization*, pages 10-17, Madrid.

Barzilay, Regina, Kathleen McKweon, and Micheal Elhadad. (1999). Information fusion in the context of multi-document summarization. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 550-557, College Park, Maryland.

Choi, Freddy Y. Y. (2000). Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics*, pages 26 - 33, Seattle, Washington.

Collins, Michael. (1997). Three generative, lexicalized models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, Spain.

Halliday, Michael and Ruqaiya Hasan. (1976). *Cohesion in English*. Longman, London.

Hoey, Michael. (1991). *Patterns of Lexis in Text*. Oxford University Press.

Jing, H. and K. R. McKeown. (2000). Cut and paste based text summarization. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics*, Seattle.

Knight, K. and D. Marcu. (2000). Statistics-based summarization - step one: Sentence compression. In *Proceedings of the 17th National Conference on Artificial Intelligence*, Austin.

Mani, Inderjeet, Barbara Gates, and Eric Bloedorn. (1999). Improving summaries by revising them. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 558-565, Maryland.

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. (1990). An on-line lexical database. *International Journal of Lexicography*, 3(4):235-312.

Morris, Jane and Graeme Hirst. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21-48.

Ratnaparkhi, Adwait. (1996). A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, University of Pennsylvania.

Reynar, Jeffrey C. and Adwait Ratnaparkhi. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C.