A Query-Focused Multi-Document Summarizer Based on Lexical Chains

Jing Li[*†], **Le Sun**[*]
Institute of Software[*]
Chinese Academy of Sciences
{lijing05, sunle}@iscas.cn

Chunyu Kit[†], **Jonathan Webster**[†]
Dept. of Chinese, Translation & Linguistics[†]
City University of Hong Kong
{ctckit, ctjjw}@cityu.edu.hk

Abstract

This paper presents our work on query-focused multi-document summarization with the enhanced IS_SUM system. We focus on improving its lexical chain algorithm for efficiency enhancement, applying the WordNet for similarity calculation and adapting it to query-focused multi-document summarization. We present its performance in terms of its official DUC2007 evaluation results together with some other improvements.

1 Introduction

The IS_SUM system was initially designed for DUC2005 [2] and was further developed for DUC2006 [3]. For DUC2007 we tried to carry out a number of necessary improvements aimed at enhancing its efficiency and performance. We continue to use the existing lexical chain algorithm as our major means for extraction. Several modifications have been made to the algorithm for ameliorating its efficiency and adapting it to query-focused multi-document summarization. The method for calculating word similarity is replaced by a more reasonable one.

In the remaining sections of this paper, we will first introduce the modified lexical chain algorithm for summarization in Section 2. Section 3 will present our system's architecture and Section 4 its evaluation results. Section 5 concludes the paper and discusses future work.

2 Lexical Chain Algorithm

In 1991, Morris and Hirst first gave a logical description of the implementation of lexical chain using Roget dictionary [4]. Hirst and Onge used WordNet for lexical chain construction and adopted a strategy to choose a word's sense with respect to those words occurring ahead of it [5]. An optimization strategy was put forward by Barzilay and Elhadad in [6] to insure that all senses of a candidate word be properly considered. It was applied to generate coherent summaries for single document summarization.

Our DUC2007 task is to carry out query-focused multi-document summarization using lexical chain. Following our previous work for DUC 2006, we move on to dealing with a few specific problems concerning the application of Barzilay and Elhadad's strategy, including the time/space cost increasing dramatically along the size of input documents, and the weight of query that the strategy does not handle. Therefore we need to take care of them in our implementation.

2.1 Candidate Words

After text preprocessing, words are extracted from input documents as candidate words for building lexical chains. Three types of words are selected, namely, nouns, noun compounds and name entities. We identify nouns with the aid of Stanford tagger [7] for POS tagging, develop a simple script for recognizing adjective-noun and noun-noun compounds in each document, following the practice in [6], and use GATE [8] to annotate name entities. Nouns and noun compounds are used to build lexical chains directly, whereas name entities are used for sentence selection.

2.2 Similar Senses

For lexical chain building we have to determine whether a word is semantically related to any others. We use the WordNet to find out such relationships between words. In the WordNet, nouns with the same meaning are grouped into a synset, and synsets are organized into a hierar-

chical lexical tree in terms of their semantic relations, as exemplified in Figure 1 below [9].

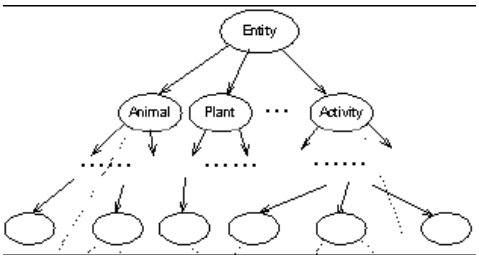

Figure 1. WordNet's lexical tree for nouns

Any synset as a leaf node in the tree has a path through its super-ordinate synsets back to the root, e.g., oak @-> tree @-> plant @-> organism @-> entity. The "@->" operation is an upward move in the lexical tree, going from specific to generic [9].

There are two methods to calculate the similarity between two words with respect to the lexical tree.

1) The similarity of two words is assessed by the depth of their maximal co-ancestor, i.e., the root of the minimal tree that subsumes both words. That is,

$$sim(c_i, c_j) = \begin{cases} \dfrac{d(c^*)}{max_{c_k}[d(c_k)]}, & if \quad c_i \neq c_j \\ 1, & otherwise \end{cases}$$

where c_i and c_j are the synsets to which the two words belong, respectively, c^* is their maximal co-ancestor, $d(c)$ stands for the depth of the synset c, and c_k ranges over all synsets in the synset hierarchy. Intuitively, if $c_i = c_j$, their similarity is 1. [9]

2) The similarity of two words is calculated as the ratio of their commonality weight and positional weight, formally defined as

$$\mathrm{sim}(w_1 \ldots w_n) = \frac{\mathrm{com}(w_1 \ldots w_n)}{\mathrm{pos}(w_1 \ldots w_n)}$$

where com($w_1 \ldots w_n$) and pos($w_1 \ldots w_n$) are the commonality and positional weights of words $w_1 \ldots w_n$, respectively. The weight of a word is defined as its depth in the lexical tree. This similarity ranges from 0 to 1, with 1 for two identical words and 0 for words with no common ancestor.

A problem with the first method is that the similarity of c_i and c_j may equal that of c^* and c_j, if $d(c_i) < d(c_j)$. To avoid this, we opt for the second choice.

2.3 Lexical Chain Building

Lexical chain building is conducted in two steps, single document chain building and multi-chain building.

The purpose of the first step is to generate the strongest lexical chains to represent the main themes of a document [6]. It is carried out by the following procedure.

1) Sort the candidate words in the descending order of frequency.
2) Retrieve the next most frequent word's sense set from the WordNet and create a chain for each sense.
3) For each chain so created, search through the remaining words for those that can be inserted into the chain according to the similarity between their synsets.
4) Repeat 2) and 3) until half of the candidate words have been processed.
5) Merge the resulting chains; score the final chains.

We give a higher priority to the words with a higher frequency, because they can strengthen the score of their chains. The strongest chains often contain the most frequent words. In this way, we can get the strongest chains for a document. Also, every word in a chain has a specific sense, bearing no ambiguity.

In the second step for multi-chain building, we merge all strongest chains from each document into a new chain set as the final result of lexical chain building. It is implemented as follows:

1) Select the strong chains from the existing chain sets following the "Strength Criterion" [6].
2) Merge these chains and remove some of them to insure that one word occurs once in each chain. Score and sort the resulting chains.

Repeated words are not removed until this step. Because keeping more distinct words in the previous steps would keep a better chance in this step for merging multi-chains with at least a common word and arriving at a better representation of the main themes of the input documents.

2.4 Summary Generation

Our summarizer is based on sentence extraction with the aid of lexical chains. We score a candidate sentence with the following formula

and extract sentences with the highest scores:

$$Score=\alpha P(chain)+\beta P(query)+\gamma P(nameentity)$$

where $P(chain)$ is the sum of the scores of the chains whose words come from the candidate sentence, $P(query)$ is the sum of the co-occurrences of key words in a topic and the sentence, and $P(nameentity)$ is the number of name entities existing in both the topic and the sentence. Each score is normalized first. We select the sentence with the next highest score until reaching the word number limit for a summary.

In our experiment, $P(query)$ and $P(nameentity)$ are found to affect the system's performance remarkably. Empirically, the three coefficients α, β and γ are set to 0.2, 0.3 and 0.5, respectively.

Comparing the manual summaries with our machine-generated ones, we find that the manual ones contain more short sentences and, accordingly, cover more content within a preset number of words. We resort to sentence compression for possible improvement. A sentence is first parsed into a syntactic tree using Charniak's parser [12], and then passed to a sentence compressor [11] for core word extraction. Unfortunately, the sentence compressor sometimes destroys the backbone structure of a sentence or returns too few words. Even so, our submission to DUC2007 could have been better if such sentence compression were applied. Certainly, a better sentence compressor is needed for our summarizer.

3 System Architecture

The IS_SUM system consists of 3 basic modules for preprocessing, modeling and summarization, respectively, as illustrated in Figure 2, where the arrows show the data flow in the system.

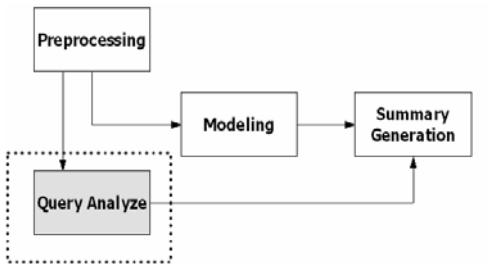

Figure 2. IS_SUM modules

3.1 Text Preprocessing

Each input document undergoes necessary text preprocessing, including sentence splitting, lemmatization, word counting, POS tagging, noun compound recognition and name entity annotation. The preprocessing outputs a candidate set of words for lexical chain construction.

3.2 Modeling

A set of lexical chains are built for each document with the algorithm given above. Then, the resulting chain sets are merged for multi-document summarization. Two chains are merged if they have at least a common word with the same sense. Finally, we got a merged chain set for each input document.

3.3 Summarization

Each candidate sentence is scored first with the scoring scheme given above in Section 2.4. Then, the sentence with the next highest score is recursively selected until the total number of words has reached a given limit, i.e., 250. If sentence compression were applied, long sentences (>30 words) would be parsed by a syntactic parser and then compressed into short ones by a sentence compressor as a preparation for sentence selection.

4 Evaluation

The two official evaluation results for our system in DUC2007 are presented in this section as follows.

4.1 Manual Evaluation (Linguistic Quality)

Table 1 and Table 2 present the linguistic quality scores for our system in DUC2006 and DUC2007 respectively. The linguistic quality is judged in terms of grammaticality, non-redundancy, referential clarity, focus, structure and coherence, identified respectively by L1, L2, ... and L5 in the tables.

Table 1 IS_SUM's average linguistic quality in DUC2006

System	L1	L2	L3	L4	L5
IS_SUM	3.96	4.36	3.24	3.49	2.67

Table 2 IS_SUM's average linguistic quality in DUC2007

System	L1	L2	L3	L4	L5
IS_SUM	1.69	3.44	2.22	2.47	1.4

The average linguistic quality scores in DUC2007 are worse than before. Lacking sentence compression is one of the causes. In general, shorter sentences are more preferable in a

summary, as indicated by the comparison of our linguistic quality evaluation results in DUC 2005, 2006 and 2007.

4.2 ROUGE Evaluation

Table 3 shows the ROUGE scores for our summarizer. In general, the performance of our system this year is about the same as last year. A main reason is that many modifications for improvement were not finished in time.

Table 3 Our ROUGE scores

System	ROUGE-2	ROUGE-SU4
Highest	0.12448	0.17711
IS_SUM	0.08039	0.13503
Baseline	0.06039	0.10507

Table 4 gives the ROUGE evaluation of our system after the modifications. This performance could probably have put it very close to rank 17 among 32 in DUC2007.

Table 4 Our ROUGE scores after modifications

System	ROUGE-2	ROUGE-SU4
Highest	0.12448	0.17711
IS_SUM	0.09848	0.14801
Baseline	0.06039	0.10507

5 Conclusion and Future Work

Our work for DUC2007 shows that lexical chain method is useful in extracting important sentences. Unfortunately, it has not illustrated any advantage in eliminating redundant content units and/or responding to queries. In our future work, we will try some other query analysis methods like agent finding and domain analysis and also explore effective approaches to query expansion for performance enhancement.

6 Acknowledgment

The research work described in this paper was partially supported by City University of Hong Kong through the Strategic Research Grant #7001879.

Reference

[1] DUC 2005/2006/2007 http://duc.nist.gov.

[2] Zhou, Q., L. Sun, and J.-Y. Nie. 2005. IS_SUM: A multi-document summarizer based on document index graphic and lexical chains. DUC2005.

[3] Zhou, Q., L. Sun and Y. Lv. 2006. ISCAS at DUC 2006. DUC 2006.

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

[5] Hirst, G., and D. St. Onge. 1998. Lexical chains as representation of context for the detection and correction of malapropisms. In Fellbaum, C. (ed.), *WordNet: An electronic lexical database*, pp. 305-332. Cambridge, MA: MIT Press.

[6] Barzilay, R., and M. Elhadad. 1997. Using lexical chains for summarization. *ACL/EACL-97 Summarization Workshop*, pp.10-18, Madrid.

[7] The Stanford Natural Language Processing Group http://nlp.stanford.edu/software/tagger.shtml.

[8] GATE http://gate.ac.uk.

[9] Huang, G., and W. Liu. 2004. An online adaptive method for personalization of search engines. W*eb Information Systems – WISE 2004*, pp.422-427, LNCS 3306.

[10] Maynard, D., and S. Ananiadou. 2000. TRUCKS: A model for automatic multi-word term recognition, *Journal of Natural Language Processing*, 8(1):101-126

[11] Balazer, J. 2005. Sentence compression using a machine learning technique, http://www.eecs.umich.edu/~balazer/sc/.

[12] Charniak, E., S. Goldwater, and M. Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pp.127-133, Montreal.