Cosine Similarity Based Non-Redundant Summarizer

Swan Pye Zaw, Kyaw Zar Zar Phyu
University of Computer Studies, Yangon
swanpye007@gmail.com, kyawzarzarphyu@gmail.com

Abstract

As summarization inherently assign more weights to the more important sentences in an IT news article, this may improve the precision and recall score of the article. This paper uses the redundant reducing algorithm to improve the summarization technique. Redundancy in summaries was reduced to different levels and its effect on summarization meaning was investigated. This paper presents the summarization system: an efficient method to extract the most important and non-redundant "n" sentence segments based on sum of similarity. In this system, the characteristics are listed as follows: 1) using preprocessing to delete the additional information that won't turn up in the summarization; 2) getting sentence segment by syntax; 3) redesigning the vector similarity between a pair of sentences by using sum of similarity. Because of all of above properties, experimental results show that this system's approach compares favorably with auto summarizer of MS word.

1. Introduction

Document summarization has been a research topic since the 1950s; however, it became active in the second half of the 1990s. Nowadays, with the development of the Internet, it is becoming more and more important. There are two major types of text summary: extract and abstract. Extract summarization means the summarized text is extracted from the original text on a statistical basis or by using heuristic methods or a combination of both. Abstract summarization means the summarized text is an interpretation of the original text. The quality of an extract summary might not be as good as an abstract summary, but it is considered well for a reader to understand the main ideas of a document, so many works in this area are based on extraction.

In addition to extracts and abstracts, summaries may differ in several other ways. Some of the major types of summary that have been identified include indicative (keywords indicating topics) vs. informative (content-laden); generic (author's perspective) vs. query-oriented (user-specific); background vs. just-the-news; single- document vs. multi-document; neutral vs. evaluative. A full understanding of the major dimensions of variation, and the types of reasoning required to produce each of them, is still a matter of investigation. This makes the study of automated text summarization an exciting area in which to work.

This paper is structured as follows. Section 2 presents the overview of text summarization. Section 3 describes the preprocessing steps, used in our summarization system. Section 4 describes initial summarization processes. Section 5 presents redundancy reducing on initial summary. Section 6 expresses the implementation of summarization. Section 7 evaluates the performance of the system. Section 8 concludes the paper.

2. Overview of the Summarizer

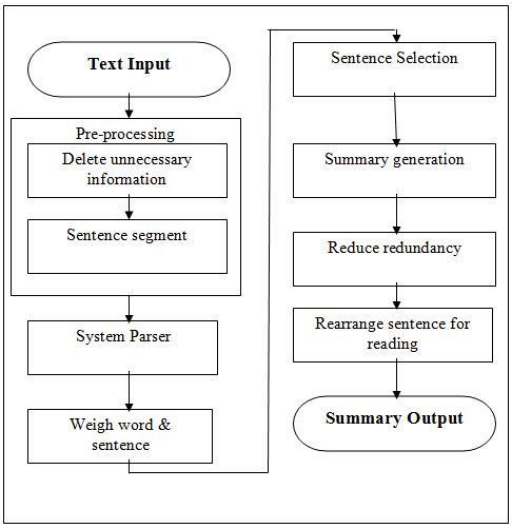

Figure: 1. Overview of the summarizer

Our summarization system is to employ IR techniques after preprocessing as far as they can take us, and then to augment them with statistical methods. At last we use a way to reduce the redundancy. In this system, deletion of comma parentheses and sentences segmentation will be performed as preprocessing step. After the text preprocessing, initial summarization step will be carried out. In this step, stop word removal, porter stemming algorithm, weighting the terms, and calculating sentences similarity will be included. After building initial summary, the redundancy will be reduced. Finally the non-redundant summary will be produced as the result of this system. Precision and recall will be calculated by standardizing on Microsoft Word summary. Figure 1 is the overall processes of system.

3. Preprocessing

In the preprocessing phase, removing comma parenthesis and sentence segmentation will be performed.

3.1. Deleting Comma Parenthesis Text

Words enclosed by curly braces ")" are called comma parenthesis, which are considered as additional information inside a segment, and won't turn up in summarization. So we use the preprocessing to delete the content between “(” and “)”, then let the remaining text as the input text.

3.2 Segmenting Sentences

Most of the experiments show that the sentence segmentation can affects the performance of the summarization system. In our system, we first divide the text to sentence by sentence. We don’t only use a short list of end-of-sentence punctuation marks, such as periods, question mark, and exclamation point, colon, semicolon etc, since these forms of punctuation are not used exclusively to mark sentence breaks, sentence boundaries are ambiguous. For example a period can be used in an abbreviation, as a decimal point, in e-mail addresses and to indicate ellipsis. Some examples are shown below.

(a) She needs her car by 5 p.m. on Saturday evening.
(b) At 5 p.m. I had to go to the bank.
(c) “I have a dream!” is very famous.
(d) Decimal, e-mail addresses, abbreviations, initials in names, honorific titles.

We correct the condition to detect the sentence: When the punctuation was met, the next character is detected whether space or not, if it is yes, this punctuation assume to be sentence boundary. If there is no space, this punctuation isn’t the end of the sentence. By this way we can solve above (c) and (d), such as the middle period in a.m. / p.m. / A.C / B.D.

Every complete sentence starts with an uppercase letter, so after (1) we check the next character is whether uppercase, now we can solve the problem like this above (a), but we can’t still solve the problem (b) by use this way. So we decide to use more semantic ways to solve in future.

Till now we can get the number of the sentence and sentence information in the input text.

4. Initial Summarization

After the preprocessing steps summarization process will be started by removing worthless words, applying porter stemmer, calculating term document frequency and cosine similarity on the sentences.

4.1 Removing Stop words

Many of the most frequently used words in English are worthless in IR and text mining. These words are called *stop words* e.g. the, of, and, to, etc. Typically about 400 to 500 such words were used in a document. For an application, an additional domain specific stop words list may be constructed. In this system, we used the BOW library for stop word removal [13]. Since stop words were not useful for searching of text mining and have a large number of hits, removing stop words will reduce indexing (or data) file size and improve efficiency.

4.2. Applying Porter Stemmer Algorithm

Most of the words in a text document have various morphological variants. Since the variants have a similar semantics they can be considered as equivalent for the purpose of many retrieval tasks. Consider for example the words \connecting" and \connect": they are not recognized being equivalent without having them reduced to their stem. A stem is not further analyzable into meaningful elements and carries the principle portion of meaning of the words in which it functions. Stemming is the process of reducing a word to its stem, and a stemmer or a stemming algorithm is a computer program that automates the task of stemming [12].

In our system, we use Porter Stemmer algorithms. The Porter Stemmer is based on the idea that the suffixes in the English language are mostly made up of a combination of smaller and simpler suffixes. Porter Stemmer is most widely used in IR research, and can be thought of as a lexicon-free stemmer because it uses the cascaded rewrite rules that run very quickly and do not require the use of a lexicon. For the stemmer, the input is text; the output is the number of the term and whole term contents. After that we can count the frequency of the occurrence *tf* for each term in the text, and we can get the term-id (term identification number). These are the contents included in the future term table.

4.3 Calculating the Weight of the Term

Our summary is meant to use simple and efficient term weighting algorithms. “Word features” employed by many other systems which may be helpful in a specific text domain, but they would have to be redesigned whenever it were ported to a new domain. We also find that words in the title/topic and/or appearing in the first/last few sentences can be given more weight by means of an alterable parameter. At the same time, we find only the capital letter feature can show important information in the text. Our system reads each sentence and gives value to the capitalized words,

we define the value P_{upcase}, if the word is capital letter, and it receives a score grater than 1, otherwise 1. The *tf* × *idf* method proved itself better than all the other methods of weight computation. Term frequency of the K_{th} term in a document, tf_k , times the logarithm of the number of documents in a collection N, divided by the number of documents where this term occurs n_k : $tf_k \times \log(N/n_k)$. This formula measures how frequently the term in a sentence occurs relatively to their occurrence in the entire document. We define as term weight k_w.

$$tf_k \times \log(N / n_k) \times P_{upcase} \quad (1)$$

4.4 Finding Cosine similarity

After obtaining the term weights of all terms, it's easy to apply traditional similarity measures like the cosine similarity to compute the similarity of two sentences. The cosine similarity between two sentences S_i and S_l is defined as:

$$\cos(S_i, S_l) = \frac{\sum_{j=1}^{m} w_{ij} w_{lj}}{\sqrt{\sum_{j=1}^{m} w_{ij}^2 \cdot \sum_{j=1}^{m} w_{lj}^2}}, i, l = 1, ..., n, \quad (2)$$

where w_{ij} is the term weight of the term t_j in the sentence $S_i = (w_{i1}; w_{i2}; :::; w_{im})$, $i = 1; 2; :::; n$, $j = 1; 2; :::; m$.

Finally, as to selection of sentences to generate a summary all sentences are ranked according to their relevance scores calculated from formula (2), and a designated number of top-weight sentences are picked out to form the summary.

5. Reduce redundancy

Our way highly improves the diversity of information coverage of summary, it is robust and transparent. To minimize summary redundancy, we do not consider sentence dissimilarity; we remove terms that have already been included in other important sentence. Figure 2 is the algorithm of our way to reduce redundancy [8].

After generating of initial summarization, we extract the *n* top-ranked sentences. We use the whole summary as a baseline against any individual sentence, then we choose first sentence in initial summary because we think the first sentence in initial summary is the most important sentence. Then terms that appear in the first sentence are stricken out of the baseline, so as to remove content from the document. The process repeats until no more term can be stricken out from the baseline string.

Input: Initial summary
Output: non-redundant summary
1: Baseline (B) ← Initial summary
2: Repeat
3: $S^* \leftarrow S_i$ with max sim(B,S_i), ($1 \le i \le n$, but we initialize i=1)
4: Output ← *Output* ∪ S^*
5: Remove from B all terms occurring in S^*
6: until B not changed

6. Implementation of Summarized System

In this system, the IT news article to be summerized will need to be loaded for sentence segmentation as shown in figure 3.

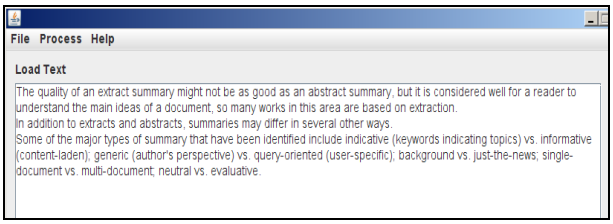

Figure: 3. Document loader for text preprocessing

The loaded text will be segmentated by using above methods expressed in section 3. Figure 4 illustrates the sentence segmentation/splitting process.

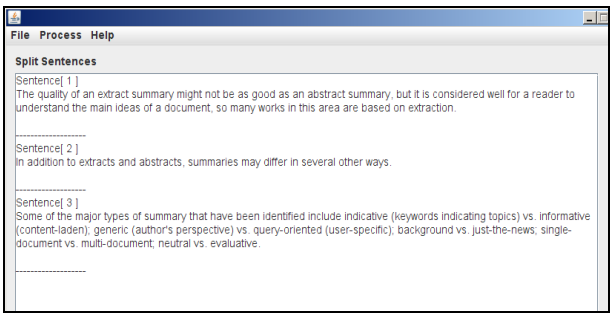

Figure: 4. Sentence splitting process

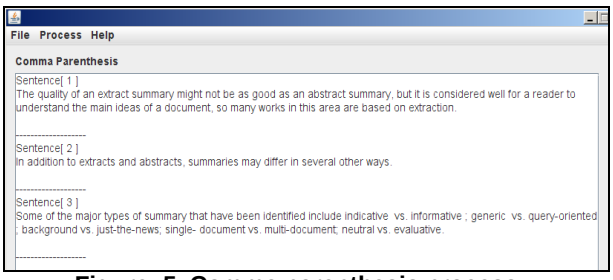

Figure: 5. Comma parenthesis process

After splitting the loaded sentences, we will delete the content between "(" and ")", then let the remaining text as the input text, as shown in figure 5.

After tf-idf calculation we will reduce the redundancy sentences from the initial summary. If original document has no redundancy, the initial summary phase and reduce redundancy phase will not be different.

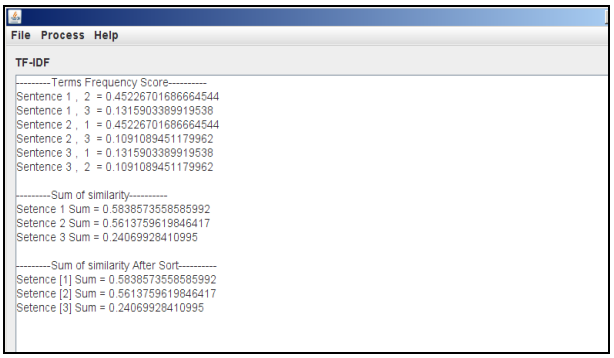

Figure: 7. TF-IDF Calculation process

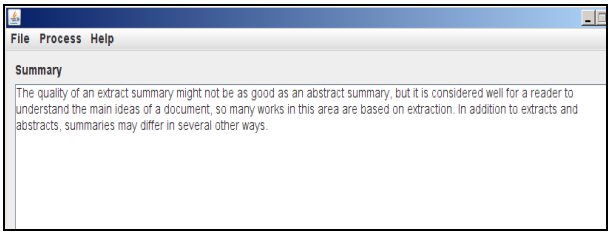

Figure: 8. Generated summary

7. Performance Evaluation

In our paper to evaluate the importance of the sentences we use classical IR precision and recall measures. To calculate the similarity measure each sentence must at first be represented in a suitable form. In our method, a sentence S_i is represented as bag-of-terms $S_i = (t_1; t_2; :::; t_{mi})$, instead of the term-based frequency vector. Here *mi* denotes the number of terms in a sentence S_i:

The similarity between pair of sentences S_i and S_l is evaluated to determine if they have similarity relation [14].

The formulas (3 and 4) are classical IR precision and recall measures. In our system, these formulas can be taken as follow:

$$Precision = \frac{\#\text{of sentences extracted by the system which are in the target S}}{\text{total \# of sentences extracted by the system}}$$

$$P(S_i, S_l) = \frac{|S_i \bigcap S_l|}{|S_i|} = \frac{|S_i \bigcap S_l|}{m_i}, i \neq l = 1, 2, ..., n, \qquad (3)$$

$$Recall = \frac{\#\text{of sentences extracted by the system which are in the target S}}{\text{total \# of sentences in the target summaries}}$$

$$R(S_i, S_l) = \frac{|S_i \bigcap S_l|}{|S_l|} = \frac{|S_i \bigcap S_l|}{m_l}, i \neq l = 1, 2, ..., n, \qquad (4)$$

We implemented the summarization system as described in the previous sections and submitted the results of our system. We employ the standard measures to evaluate the performance of summarization, i.e. precision and recall measure. We assume that MS word would be able to identify the most important sentences in a document most effectively. The evaluation results are shown in Table 1.

Table 1 Precision and Recall Result

Document	Precision	Recall
2600dcr1:	60.448	73.193
2600dcr2:	73.373	72.093
Agntsnfo:	66.772	71.284
Aotd1_a2:	50.598	63.819
Aotd1_a6:	64.216	74.857
Aotd3_2:	71.053	73.469
Aotd3_8:	72.941	73.810
Aotd4_5:	54.106	78.322
Aotd5_11:	65.649	77.477
Aotd6_3:	56.250	58.065

MS word summarized 14 sentences for document number 2600dcr1 and 5 sentences for document number 2600dcr2. We generate summary with our summarizer for these documents and compute the performance by using formula 3 and 4. Our summarizer gain 60.448, 73.193 and 73.373, 72.093 for precision and recall score of document 2600dcr1 and 2600dcr2 respectively.

8. Conclusion

We have described a practical automatic text summarizer based on sentence extraction. Our method adds pre-processing to delete additional information that should not be included in the summarization before stemming, and uses the sum of similarity. Our text summarization method creates generic summaries by scoring and extracting sentences from the source documents. We provide experimental evidence that our approach achieves reasonable performance. Finally our system generates summarization process well by combining cosine similarity and reduce-redundancy methods.

Reference:

[1] A. M. Lam-Adesina and G. J. F. Jones, "Applying summarization techniques for term selection in relevance

feedback", Proceedings of the 24th ACM SIGIR, pages 1–9, 2001.

[2] Canasai Kruengkrai and Chuleerat Jaruskulchai, "Generic Text Summarization Using Local and Global Properties of Sentences", Proceedings, IEEE/WIC International Conference, Pages:201 – 206. (2003)

[3] D. Marcu, "The Rhetorical Parsing, Summarization, and Generation of Natural Language Texts", PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1997.

[4] G. Salton, A. Singhal, M. Mitra, and C. Buckley, "Automatic text structuring and summarization", Advances in automatic text summarization, MIT Press, 1999.

[5] H. P. Luhn, "The automatic creation of literature abstracts", IBM Journal of Research and Development, pages 159–165, 1959.

[6] H. P. Edmundson, "New methods in automatic extraction", Journal of the ACM, 16(2):264–285, 1969.

[7] I. Mani and M. Maybury, "Advances in Automatic Text Summarization", MIT Press, 1999.

[8] J. Carbonell and J. Goldstein, "The use of MMR, diversity-based reranking for reordering documents and producing summaries", In Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval, pages 335-336,1998.

[9] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell, "Summarizing text documents: Sentence selection and evaluation metrics", Proceedings of the 22nd ACM SIGIR, pages 121–128, 1999.

[10] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Seeing the whole in parts: Text summarization for web browsing on handheld devices", WWW10, May 2-5 2001.

[11] G. Salton, A. Singhal, M. Mitra, and C. Buckley, "Automatic Text Structuring and Summarization", Advance in Automatic Text Summarization, Cambridge Massachusetts, MIT press, 341-355,1997 .

[12] http://www.tartarus.org/~martin/PorterStemmer/

[13] http://www.rainbow.com/bowlib/

[14] http://www.hsl.creighton.edu/hsl/Searching/Recall-Precision.html