

APPROACH OF TOPICAL MULTI-DOCUMENT SUMMARIZATION

CS359 Project

Xuehai Zhang

Dept of CS, Univ. of Chicago
hai@cs.uchicago.edu
<http://people.cs.uchicago.edu/~hai/cs359/>

Abstract

This paper describes an innovative algorithm that aims to solve the topical multi-document summarization problems. Given a user input topical query and a large unlabeled document collections, the algorithm first clustering the documents into a set of clusters by using the efficient spherical k -means algorithm and ranks each document and the cluster on the basis of the approximation to the topic. Thus the documents collection is curtailed to several high-ranked clusters, in each of which only an appropriate number of high-ranked documents are reserved. Then some existing merge, reduction and single-document summarization techniques are employed to generate the final summary. We also present the implementation (incomplete) of the algorithm and some experimental results. Evaluation of this approach currently is not available and needs for future work.

1. Introduction

The high-speed networking, large volume storage and other techniques make the large collections of the documents become increasingly popular. Like Google, the largest search engine on the web currently, it catalogues over 1.39 billion web pages. A large amount of scientific data is generated every minute and stored as documents. How to deal with these documents is a very big problem we face today. Several research approaches could be applied on them, and text summarization is one of the hot topics and has aroused a lot of research interest.

Multi-document summarization is a more challenging field in text summarization comparing with single-document summarization. Basically, it works on a collection of documents and generate the comprehensive summary based on the content of part of or all the documents.

Different forms of summarization are useful in different situations, depending on the intended purpose of the summary and on the types of documents summarized. The multi-document summarizing problem studied in this paper is named topical multi-document summarization. It is aiming to answer such a question, can you / how can give me as much concise but valuable information as possible from a collection of documents to help me find a decent job? From this simple scenario we notice an important characteristic of this kind of multi-document summarization, a topic is initiated first and applied to the entire summarizing process. Topical multi-document summarization is very interesting, because typically, the same information is described by many different documents, hence

summaries that synthesize common information across documents but preserve the differences would significantly help people. In this paper, we present research on the topical summarization across multiple documents using clustering and reduction techniques.

We propose an algorithm by extending and modifying the algorithm presented in Stein's "Multi-Document Summarization" Methodologies and Evaluations" paper. The new algorithm shares the similar characteristic and employs similar techniques for the subcomponents with Stein's algorithm, but it differs greatly from the old one by significantly rearranging and optimizing the processing stages from a different angle of thinking.

In this paper, our main concern is obtaining a more efficient approach for summarizing a topic on a very large document collection. Due to the tight time schedule, we could not do performance evaluation by comparing with the other approaches and these are left for future work. But through the implementation and the experiments we conducted, we could tell the algorithm we proposed is a feasible approach to meet our expectation.

The paper is organized as follow. In section 2, we will review different approaches for multi-document summarizations and some ongoing projects, but we emphasize on Stein's work that gives our inspiration on our work. Section 3 describes the algorithm presented in Stein's "Multi-Document Summarization" Methodologies and Evaluations" paper. In section 4, we propose our algorithm for topical multi-document summarization. We will give the motivation of the approach and the comparison with Stein's algorithm; also techniques used in each stage of the algorithm will be discussed in details. Section 5 is the implementation of the algorithm as well as several experimental instances. In section 6, we talk a little bit of the performance and the evaluation. The last section is the conclusion and future work.

2. Related work

Regina Barzilay and Kathleen R. McKeown (1999) have presented a method to automatically generate a concise summary by identifying and synthesizing similar elements across related text from a set of multiple documents. The approach is unique in its usage of language generation to reformulate the wording of the summary. The Carnegie Group's work on multi-document summarization (Carbonell and Goldstein 1998) depends on the maximal marginal relevance measure to organize the final summary and detect redundancy. In this research work, clusters are formed and a representative segment is presented to the user. Gees Stein and Amit Bagga carry out a similar but more recent work. The system they proposed produces multi-document summaries using clustering techniques to identify common themes across the set of documents. For each theme, the system identifies representative passages that are included in the final summary. As we said before, we develop our algorithm and system by extending and modify the algorithm in Stein's research, this piece of work is more closely to ours than any other, so we will discuss more in the later sections.

3. Algorithm from Stein's paper

The goal of the algorithm in Stein's "Multi-Document Summarization: Methodologies and Evaluations" paper is to give people the gist of the original documents by creating cross-document, indicative summaries. The algorithm they proposed tries to generate a final summary by getting rid of those similar, identical, or redundant information across the documents. What is more, the information in the final summary should be arranged in a proper way that keeps the logic of the documents and makes sense to the users.

The algorithm consists of several important sub-tasks and they are arranged and executed in a certain order. The algorithm is as follows:

- Summarize each document

The first step is to create individual single-documents summaries for all documents in the collection.

- Group the summaries in cluster

The second step is the grouping of the individual summaries into clusters. Documents are clustered on the basis of the contents of their summaries where a cluster consists of summaries that describe a similar topic. The researcher experimented with a variety of approaches for producing the clusters, including several graph-theoretic approaches.

- For each cluster select representative passage(s) that will contribute to the final summary

The third step of Stein's algorithm involves selecting a member of a cluster as the representative summary for that cluster. When a user wants a topical summary, the topic description is used to pick the summary that has most similarity to the topic.

- Organize these passages in a logical way

The last step of the multi-document summarization process involves organizing the selected passages in an order that makes sense to the reader. Currently, they organize the selected passages based upon topic similarity.

The researcher also implemented a system named XDOCTOOL, to support this algorithm.

4. Our proposed algorithm

4.1 The approach

Since our work is inspired by Stein's research, especially the algorithm described on section 3, it is easy to notice that our approach is based on similar initial assumption as Stein's system, like all the documents in the experiments should be text only and well formatted, etc. But our proposed algorithm is quite different from the algorithm above.

The basic idea is: we will not do summarizations towards individual documents at the beginning stage; on the contrary, we employ document-clustering techniques to group the documents into clusters. Documents in each of the cluster are more related to each other than the other documents in another cluster. During the process of clustering, the rank of each document is also calculated and marked. Here we define rank as an index to indicate the approximation between the content of this document and the given user-input topic. The higher the rank, the more approximately the document accords to the given topic. When finishing calculating of the documents' ranks, we also need to decide the ranks of the clusters. Here the rank of a cluster means the possibility that the documents in this cluster contribute to the final cross-documents summary. For simplicity, the rank of a cluster is a kind of aggregate function with all the inside documents' ranks as inputs.

Then we may choose the first n high ranked clusters, and in each of these selected clusters, I choose the first m high ranked documents. Here n and m are threshold values defined by the users or by the program itself. One alternative here is that we choose a degressive sequence of m_i for a certain selected cluster i . The next step also has two alternatives. The first one is that we directly employ Stein's algorithm step 1) and 3) to work on the selected m_i documents in the selected cluster i where i between 1 and n ; then we use step 4) to integrate all these resulted passages to form the final summary. The other choice for this step is that we first employ certain merge and reduction techniques to merge all the m_i documents within cluster i into one single aggregated document D_i from cluster 1 to cluster n ; then we use the step 1), 3) and 4) in Stein's algorithm to work on the new documents collection $\{D_1, D_2, \dots, D_n\}$ and integrate all these resulting passages into the final summary. The two approaches for the final step we describe here are not both developed in the implementations and applied to the experiments, only the first approach is realized.

4.2 Motivation/Comparison

Several things contribute to the development of our thinking of the new algorithm described above. First, we notice Stein's algorithm is an approach more general than directly solving the topical multi-document summarization but we are looking for the methods for the latter task; second, the efficiency of Stein's algorithm for topical multi-document summarization is good for argument. In Stein's paper, it claims that "when a user wants a topical summary, the topic description is used to pick the summary that has most similarity to the topic", but the paper lacks the details of the "topic description" mentioned here. We think it is important to know the approximation between the content of a certain document and the given topic, that is the reason why we table a proposal to calculate the rank of each document and each cluster; third, because Stein's algorithm is not for topical multi-document summarization by default, it performs the single-document summarization for all the documents in the collection first. We think it is a kind of cost to do so because those documents that is irrelevant to the given topic will add no piece of information to the final summary, thus there is no need to do summarization for them at all. Our approach is based on the assumption that given a specified topic for multi-document summarization, we may want to get rid of those topic-irrelevant

documents as earlier as possible while keeping them intact and away from the single-document summarization.

4.3 Documents clustering

The document clustering techniques used in our algorithm are the memory-efficient multi-threaded preprocessing scheme and a fast clustering algorithm named spherical k-means algorithm, which are depicted in the research of Inderjit Dhillon and James Fan, 2000.

The starting point for applying clustering algorithms to unstructured text data is to create a *vector space model*, also know as *bag-of-words model*. In the preprocessing algorithm we used, the output vector space model is represented as a highly sparse word-by-document matrix and stored using the Compressed Column Storage (CCS) format. In this format, we record the value of each non-zero element, along with its row and column index.

Given the vector space model, the document vectors may be represented by x_1, x_2, \dots, x_d . A clustering of the document collection is its partitioning into the disjoint subsets $?_1, ?_2, \dots, ?_k$ where $\bigcup_{j=1}^k ?_j = \{x_1, x_2, \dots, x_d\}$. Any clustering algorithm needs an objective notion

of similarity between documents. Cosine similarity is a common choice. We define the “goodness” of cluster $?_j$ as

$$\sum_{x_i \in ?_j} X_i^T C_j \quad (4.3)$$

where each x_i is normalized such that $|x_i| = 1$ and C_j is the normalized centroid of cluster

$$?_j, C_j = \frac{\sum_{x_i \in ?_j} X_i}{|\sum_{x_i \in ?_j} X_i|}.$$

Aggregating over all clusters, we can measure the goodness of any given partitioning $\{?_j\}_{j=1}^k$ using the following *objective function*:

$$O(\{?_j\}_{j=1}^k) = \sum_{j=1}^k \sum_{x_i \in ?_j} X_i^T C_j \quad (4.4)$$

The *spherical k-means* algorithm is a variant of the classical *k-means* that can quickly find a good local maximum for (4.4). The details of the algorithm could be found at the paper.

Experimental data indicates that the spherical *k-means* algorithm is very effective and it could preprocess and cluster 113,716 NSF award abstracts in 23 minutes on a Sun workstation with modest memory consumption.

4.4 Document/cluster ranking

As said before, that ranking of a document is an index to indicate the approximation between the content of this document and the given user-input topic and we have know the importance to integrate the ranking calculation and processing into our algorithm. In our algorithm, we select a very simple solution to calculate the rank of a document. Assume the give topic is a kind of query string consisting of a set of words $\{w_1, w_2, \dots, w_q\}$ and at this stage, the document-clustering algorithm has finished and returned an optimal partitioning, that is, we get cluster sets $\{C_1, C_2, \dots, C_m\}$. First we set the rank for each

document x_j as $rank(x_j) = \sum_{i=1}^q f(w_i, x_j), 1 \leq j \leq d$, here $f(w_i, x_j) = c$, c is the number of times that word w_i appears in document x_j . Then we calculate the following cost

$R = \sqrt{\sum_{j=1}^d (rank(x_j))^2}$, and then we recalculate the rank for each document x_j by dividing

R , so $rank(x_j) = \frac{Rank(x_j)}{R}$. Now we get all the ranks for documents. The rank of a

cluster is based on the ranks of the documents within this cluster. For any cluster C_i , $1 \leq i \leq m$, the documents within C_i is a subset of the whole collection, we mark it as $\{x'_1, x'_2, \dots, x'_n\}$ and $\{x'_1, x'_2, \dots, x'_n\} \subseteq \{x_1, x_2, \dots, x_d\}$. We calculate the rank of the cluster

in this way, $rank(C_i) = \frac{\sqrt{\sum_{j=1}^n (rank(x'_j))^2}}{n}$.

There are possible better ways to calculate or optimize the ranks besides the above approach. Remember when we calculate the original rank of a document, we only take the distinct words in the query string into consideration, but (1) different word might have different weight based on the type of it such as nouns, verbs, etc; (2) there could be relations between these words, also indicating a kind weights; (3) a word may have a close relation to a set of words which do not appear in the query string but could be get by consulting a global knowledge database, thus we could extend the query string by adding this set of words.

4.5 Document collection reduction

Through the ranks calculation in section 4.4, we sort both the cluster set and the document set within each cluster by the degrading of the corresponding ranks. Then we must make a decision what clusters, and what documents in those clusters might contribute to the final summary.

Two possible means could be applied here: First, we may choose the first n high ranked clusters from the total m clusters, and in each of these selected clusters, I choose the first l high ranked documents. Here n and l are threshold values defined by the users or by the program itself. Second alternative is that we choose a degressive sequence of l_i for a certain selected cluster i . The implementation employs the second approach.

4.6 Multi-documents summarization on the reduced documents subset

We discuss this aspect in more details in section 4.1.

5. Implementation

The implementation of the algorithm for topical multi-document summarization we propose above is a combinational work including several existing free software developed by other researcher and a big amount of coding we did by ourselves. By integrating these program together using UNIX shell scripts, we get 80% of the functionalities of our algorithm running automatically and smoothly. The disjoint part is the functionality depicted in section 4.6, Multi-documents summarization on the reduced documents subset. Because the tight time schedule, I could not find a source code or proper binary version of document summarizer to integrate into the whole working process. (We do find some nice interface online, such as the ProSum online and we once downloaded a summarizer software from IBM website but that is only a demo) Thus I have to do the summarization work by importing the documents into Microsoft Word and using the Microsoft Autosummarize functionality. Although we could not perform the last step decently, we think our implementation is fairly well because it satisfies the first three steps and these three steps are the most important ones to the algorithm. In the following we would have some brief explanation about what tools we employed in our implementation.

5.1 Tools borrowed

- MC - A Toolkit for Creating Vector Models from Text Documents

Developed by Univ of Texas at Austin

Available at <http://www.cs.utexas.edu/users/jfan/dm/index.html>

MC is a C++ program that creates vector-space models from text documents that can be used for text mining application. *MC* provides an efficient multi-threaded implementation that can process very large document collection.

- "spkmeans"

Developed by Univ of Texas at Austin

Available at <http://www.cs.utexas.edu/users/yguan/datamining/spkmeans.html>

"spkmeans" is a fast clustering C++ program that fully exploits the scarcity of the data set. At the heart of the program is the (spherical) K-means clustering algorithm, which is enhanced by a technique for significantly reducing the number of similarity computations required.

"spkmeans" can also reduce the dimensionality of the original word-document matrix through concept decomposition or QR decomposition of the concept vectors. This may be useful for classification and query retrieval.

- Cluster Browser - A Java program to illustrate clustering results

Developed by Univ of Texas at Austin

Available at <http://www.cs.utexas.edu/users/yguan/datamining/clusterBrowser.html>

This Java program generates a sequence of web pages to illustrate clustering results.

Input: document (and word) clusters produced by “spkmeans”

Output: Web pages that show the structure of the clusters.

- Microsoft Autosummarize
Developed by Microsoft

5.2 Tools developed by ourselves

- GetRank (400 lines)

GetRank.c is a C++ file that will take the query string as input, generating the ranks for each documents and each clusters. The rank information is written into the related files generated by MC and “spkmean” program.

- Modified Cluster Browser (300 lines)

Cluster Browser is modified to meet the need to integrate the rank information in the final web pages, and it also generate a new web interface to show the reduced clusters and documents which is used directly for summarization.

- C (100 lines)

C is a UNIX shell script to integrate all the related programs in a batch.

- stoplist (100 lines)

stoplist is a text file containing all the stop words needed in the preprocessing of the clustering.

5.3 Experiment Instance

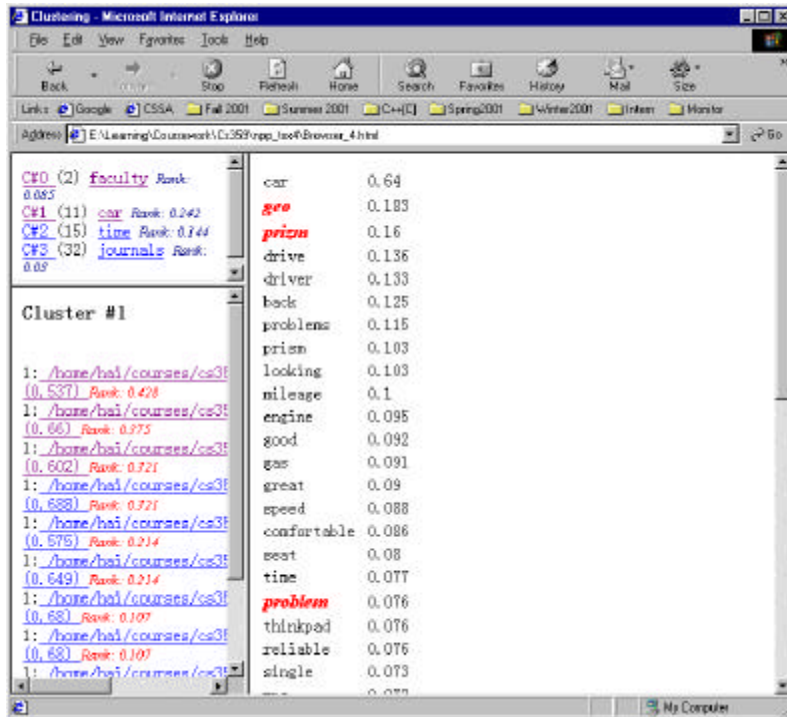
5.3.1 Text source for experiments

In our experiment, we select experimental documents from the following three text sources; the total number of documents is 60, and these documents is located at /home/hai/courses/cs359/project/data/npp or <http://people.cs.uchicago.edu/~hai/cs359/> .

- 30 selected Brown documents in <http://www.csi.uottawa.ca/tanka/Brown/original/index.html>
- Documents about the debate of profit or non-profit publishing, <http://www.nature.com/nature/debates/e-access/>
- Documents about car survey, http://www.epinions.com/auto_Make-Passenger_Cars-Geo

We conducted the following experiment by letting the topical query string be “geo prizm problem”, which means we want to generate a summary about the problem of GEO prizm model’s problem; and we let the cluster be initialized to 4.

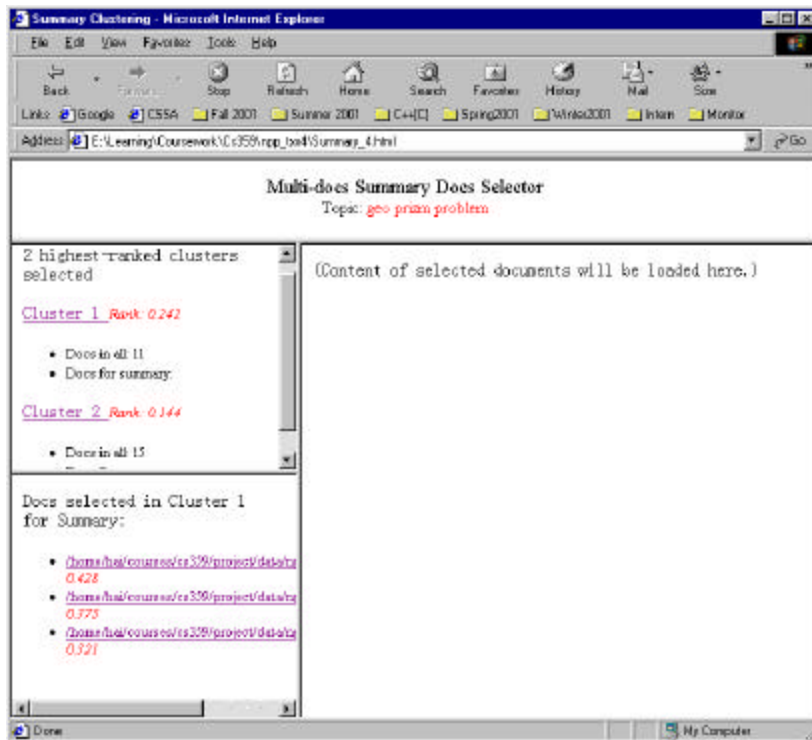
5.3.2 Instance – Clusters and documents web page



The above web page is called Browser_4.html, which is generated from the Cluster Browser program. The upper left frame has the clusters information. For example, cluster 0 has 2 files, and the most frequent word in this cluster is faculty, and the rank for this cluster is 0.242. The lower left frame has the document information for certain cluster if you click C#1 for example. It lists all the file names in this cluster, with the link to the real files and the rank of this document. The right panel has the information either about the content of a document if you click the file link on the lower left panel, or the words information if you click the frequent word link in the upper left panel. You might notice the words “geo”, “prizm”, and “problem” in this panel is marked as red, because they are part of the topical query string.

5.3.3 Instance – Reduced clusters and document web page (for summary)

Below web page is called Summary_4.html, which provides the information for the selected clusters, select documents in the selected clusters used for the final summary. From the example showed there, we see only Cluster 1 and Cluster 2 are selected for the final summary, and in cluster 1, only 3 documents are selected for final summary.



6. Discussion and future work

In this paper, we have presented an algorithm for topical multi-document summarization. We initiate our work by extending and modifying the multi-document summarization algorithm described in Stein's paper. We discuss in details of each rule in our algorithm, and provide our reasoning why we employ it comparing with Stein's algorithm. We also talk about the implementation of our algorithm and present several experimental results.

In future work, we will plan to focus on the following aspects:

- Give a feasibility proof of the algorithm based on both theory and experiment.
- Try to find the appropriate summarizer software that could be integrate into our existing implementation smoothly.
- Performance and evaluation.

Reference:

1. Gees Stein, Amit Bagga and G. Wise. 2000. Multi-Document Summarization: Methodologies and Evaluations. Conference TALN 2000.
2. <http://www.summarization.com/>
3. <http://www.cs.columbia.edu/nlp/projects.html>
4. <http://www.cs.columbia.edu/~hjing/summarization.htm>
5. Regina Barzilay, Kathleen R. McKeown, Michael Elhadad. 1999. Information Fusion in the Context of Multi-Document Summarization.
6. Carbonell, Jame and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. SIGIR '98.
7. Inderjit Dhillon and James Fan, 2001. Efficient Clustering of Very Large Document Collection.
8. www.cs.utexas.edu/users/dml/
9. <http://www.site.uottawa.ca/tanka/ts.html>
10. <http://transend.labs.bt.com>
11. CCS format, http://www.netlib.org/linalg/html_templates/node92.html
12. K-means clustering, http://www.engr.sjsu.edu/~knapp/HCI RDFSC/C/k_means.htm
13. Text source, <http://www.csi.uottawa.ca/tanka/Brown/original/index.html>
14. Text source, <http://www.nature.com/nature/debates/e-access/>
15. Text source, http://www.epinions.com/auto_Make-Passenger_Cars-Geo

Appendix:

- Getrank.c
- stoplist
- C (unix shell script)
- ClusterBrowser.java (too big, available as request; also available at <http://people.cs.uchicago.edu/~hai/cs359/>)