Multi-Documents Text Summarization Using Natural Language Processing Techniques

by

Bhargav Prabhala

A Project Report Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by

Dr. Rajendra K.Raj

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

Month Year

The project "Multi-Documents Text Summarization Using Natural Language Processing Techniques" by Bhargav Prabhala has been examined and approved by the following Examination Committee:

Dr. Rajendra K.Raj
Professor
Project Committee Chair

Dr. Leon Reznik
Professor

Dedication

To whoever...

Acknowledgments

I am grateful for ...

Abstract

Multi-Documents Text Summarization Using Natural Language Processing Techniques

Bhargav Prabhala

Supervising Professor: Dr. Rajendra K.Raj

In this age of the Internet, Natural Language Processing (NLP) techniques are the key sources to provide information required by users. However, with extensive usage of available data, a secondary level of wrappers that interact with NLP tools have become necessary. These tools must extract a concise summary from the primary data set retrieved. The main reson for using text summarization techniques is to obtain this secondary level of information. Text summarization using NLP techniques is an interesting area of research with various implications for information retrieval.

This report deals with the use of Latent Semantic Analysis (LSA) for generic text summarization and compares it with other models available. It proposes text summarization using LSA with usage of open-source NLP frameworks like Mahout and Lucene. LSA algorithm can be scaled to multiple large sized documents using these frameworks. The performance of this algorithm is compared with other models commonly used for summarization and Recall-Oriented Understudy of Gisting Evaluation (ROUGE) scores. This project implements a text summarization framework, which uses available open-source tools and cloud resources to summarize documents from english and other foreign laungages, in this case, Hindi.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Automatic text summarization is an interesting research area in which considerable amount of work has been done by researchers. With the amount of information available on the Internet, the need for summarized presentation of information has become essential. Many summarization methods were suggested in the past and worked efficiently on single documents [9] or small set of articles on specific topics [6]. Many browser applications and search engines like Google, Bing, Yahoo provide brief descriptions of results using these methods.

Multi-document summarization (MDS) has gathered attention from researchers lately due to the challenges it presents in providing well summarized reports. The issues in MDS are size constraints of documents, limited memory in computing resources, redundancy of similar sentences in multiple documents. This project resolves issues in MDS by using NLP techniques and open-source machine learning frameworks like Mahout [11] and Lucene [7].

1.1 Background

NLP is a developing area in computational linguistics which uses many machine learning algorithms to evaluate, analyze, replicate textual information. NLP is a means to simplify human-computer interaction through learning mechanisms like pattern recognition, parts-of-speech (POS) tagging and textual summarization. NLP plays a key part in allowing machines to understand and interact with humans. IBM's Watson, a system designed for answers, uses DeepQA [5] software for question analysis, decomposition, hypothesis generation filtering, synthesis for answer questions raised in native languages. Text summarization, the goal of which is to produce an abridged version of a text that is important or relevant to a user, is an important step in question answering.

1.1.1 What is Text summarization?

A good summary should be concide and relevant to the original context, incur minimal information loss. There are different types of summaries. The variants in summaries are as follows:

- Extraction vs Abstraction
- Generalized vs Query-model

Based on how a summary is created, it can be categorized into an abstractive or extractive summary. With the extractive approach, key sentences or phrases are selected from the original document to reflect the theme. The output generated gives an idea about the actual text. In the abstractive approach, the input document is analyzed and the generated summary reflects the sentiment of the original text but is represented in different words.
If a summary is generated based on a query by user and is relevant only to a specific topic, it is considered a query-based summarization. If a summary tries to obtain a high-level understanding of the input document and caters to all key points in the document, it is considered a generic summary.
Sentence and sentiment analysis is in the developing stage, and hence, abstractive summary generation using machine learning is difficult. Many attempts have been made by researchers to improve abstractive summaries by using POS tagging, WordNet, Named entity recognition and other NLP techniques. The first step toward an abstractive summary is

to obtain a well formed extractive summary. In this project, we worked towards obtaining well-defined extractive generic summaries.

1.1.2 Multi-Document summarization

MDS is a summary of collection of articles on a specific topic. The Text Analysis Conference (TAC) conducts evaluation of summarization models presented by researchers each year. A model should be able to minimize the redundancy of context and the compression ratio should be less than 10 percent to qualify for TAC evaluation. Complexity of extracting sentences in MDS over single documents is also a criteria. When dealing with multiple articles on the same topic, there may be some overlap in sentences from different articles. For example, headlines from two different papers on a Russian space mission reported the following.

"Russian Soyuz rocket starts mission to space station with 3-person international crew onboard." - Washington Post

"Soyuz taking Russian-US-Japanese team to international space station." - Reuters

When these two articles are summarized, both the sentences have high term frequency and sentence similarity and could end up in the summary if the learning algorithm does not include the proper constraints. In MDS, limiting the size of a summary is also a tedious task; in two articles on a topic there could be different opinions, which have to be captured in the summary, and this would only increase the file size. In this project, we addressed the above mentioned issues.

Most of the summarization mechanisms generate output in four stages-pre-processing, evaluation, information selection, output generation(see figure 1). The pre-processing stage involves removing meta-data, titles, figures from the original corpora for further analysis and the evaluation of information. The evaluation stage involves the analysis of datasets using machine learning algorithms to get information required for the next steps. Sentences are reviewed and selected using sentence clustering algorithms in the information selection stage, the selected sentences are put together for output generation.

1.1.3 Multilingual summarization

Due to the technological advancements, books and literature from various languages around the world have been digitized. Even though the literature is available in digital libraries to audiences across the globe, accessiblity to those documents is limited. End users would like to get some information about a foreign book before getting it translated into language known to them. Multilingual MDS is very useful in these type of situations. Multilingual summarization will take a very similar approach to normal MDS when extracting summaries and then adding a few additional steps in the preprocessing stage. The additional steps in the preprocessing stage involve language/encoding detection, lemmatization, stemming, indexing. We will discuss each of these steps further in the section on the design and implementation of summarization methods for documents.

1.2 Related Work

Since the early 1960s, several methods have been proposed by researchers. The initial works primarily focused on extracting sentences based on prominent terms, sentence lengths, random selection. The most prominent study was conducted by H.P. Luhn using term frequencies and word collections from *The Automatic Creation of Literature Abstracts* [9]. The research aimed to obtain generic abstracts for several research papers. This approach was able to handle only single documents with less than 4000 words total. Another important approach in the early stages was proposed by Edmundson [4] using term frequencies and emphasizing the location of the sentences. Sentences at the beginning and end were given priority over other sentences. In the recent past, several methods have been proposed based on statistical, graph based and machine learning approaches.

1.2.1 Statistical approach

In the statistical approach, features in sentences are selected based on concepts like co-occurrences of words and classification. Summarist [**?**] is a system proposed by Hovy and Lin, which is based on a statistical approach for sentence extraction from single documents. It defines optimal position policy by selecting words based on a certain distribution and extracts sentences containing the filtered words. Nomoto [?] proposed using bayesian classification in text summarization, the system was defined for SDS in Japanese and, in some cases, evaluated better than other systems. There are some other papers [3] that used lexical chains and bayesian method for sentence extraction. These approaches used supervised algorithms, which had to be trained in a specific domain to obtain good summary results. This approach was not efficient for new corpora from a different domain.

1.2.2 Graph-based approach

In the graph based approach each node represents a sentence from the text and nodes are connected to each other using edges. Nodes are connected to each other only when there are common terms between two nodes and the consine similarity between them is above certain threshold. From the graph based approach, topic-driven summarization can be done by obtaining a sub-graph which is similar to the topic. For a generic summary, the most connected node from each sub-graph is selected for the summary. This approach was used in ranking web pages by Google's PageRank, [?] which serves to index and search web pages. Summarization systems like *TextRank [?], LexRank [?], Hypersum [?]* use this approach for text. These systems obtain good summaries but get complicated when introduced to MDS.

1.2.3 Machine learning approach

Summarization techniques using machine learning algorithms combined with NLP have increased lately. Machine learning algorithms like Naive-Bayes, special clustering, classification methods, decision tress, markov models are being used in the implementation of summaries. Some of their noteworthy approaches includes using the Hidden Markov Models(HMM) [2] and Naive-Bayes [?]. These approaches work well with SDS but do not scale well to multi-documents. Learning algorithms like Latent Semantic Analysis(LSA) and Latent Dirichlet Allocation(LDA) [1] work well with single documents and have been highly recommended by TAC. LSA will be explained in detail in the next section.

LSA in summarization

Latent Semantic analysis is the most prominent algebric learning algorithm used for Information Retrieval(IR) from textual data. LSA, is widely used in various applications for the dimension reduction of large multi-dimensional data. Singular Value Decomposition(SVD) is the most commonly used learning algorithm for information retrieval. It is known by different names in different fields, Principal Component Analysis(PCA) in signal processing, KL Transform in image processing, LSA in textual processing. LSA works very well in single document summarization [**?**] and has been evaluated highly by TAC. Because LSA is an unsupervised learning algorithm it can be used across different domains without any corpora training. LSA obtains summaries in three stages: *Input matrix, singular value decomposition, and sentence selection*.
Input matrix(A): In this stage the original documents are transformed into a matrix which can be processed in SVD step. Transformation of a document involves many NLP methods like segmentation, stemming, word filtering, sentence removal. Values in the matrix signify the importance of terms in a document and are calculated using different weighting metrics. The selection of a weighting metric can affect the end results. Some of the weighting metrics are term frequency(tf), term frequency-inverse document frequency (tf-idf), log entropy and binary representation(0, 1). All these steps are explained in design implementation section.

Singular value decomposition, is a statistical tool for dimension reduction and feature selection. SVD factorizes a given input matrix into three matrices as shown above (Figure 2).

$$A = U\Sigma V^T \tag{1.1}$$

A:	Input matrix (m*n)
U:	Orthogonal left singular matrix (m*k)
Σ:	Scalar values matrix (k*k)
V^T:	Orthogonal right singular matrix (k*n)

The input matrix comprises values which are defined by the weighting metric and the number of times a term has appeared in a sentence. Term frequency per sentence is usually zero or one if input matrix is a big (m x n) matrix with rows defined by terms and columns by sentences. After the decomposition of the input matrix, we obtain two orthogonal matrices and one diagonal matrix. SVD is similar to eigen value decomposition, the only difference being SVD can be applied on rectangular matrices, but eigen decomposition is strictly used for square matrices.

The values of each matrix can be obtained from the original input matrix.

- left singular matrix U is comprised of eigen vectors of AA^T
- diagonal matrix Σ is comprised of eigen values from $\sqrt{(AA^T)(A^TA)}$
- right singular matrix V^T is comprised of eigen vectors of A^TA

The left matrix U provides a relationship between terms and features. Any information related to terms can be extracted from matrix U. For information related to sentences and features, right matrix V^T can be used. Reduction of dimension is based on number of features (k) required. SVD works well in dimension reduction and text summarization, but it has a few disadvantages. SVD is a time consuming process and it has to be reworked every time there are any changes to the original matrix. In addition, it cannot identify homonyms and words with different meanings. These issues can cause problems for large text analysis and summarization.

In the sentence selection stage, different algorithms can be used on the right singular matrix(V^T) obtained from the SVD on input matrix. V^T is comprised of *features x sentences*, where *features* are represented by values in rows and *sentences* by columns. In his paper [**?**], Gong discusses usage of sentences with highest value for each feature. Steinberger *et al.* [**?**] used top k sentences with sum of feature values. Clustering algorithms can be used to select sentences similar to centroids.

1.3 Hypothesis

The hypothesis of this project is that a text summarization framework using available open source tools and NLP techniques will provide optimized summary results in terms of quality and scalability. The implementation of such a generic framework will allow support to multi-document and multi-lingual summaries. The LSA based text summary framework will provide summaries with reasonable quality compared to existing systems but will have an edge in trms of scalability.

1.3.1 Summary Quality

Summary quality is the primary criteria for determining the efficiency of summary framework. Determining the quality of summary and comparisons with results obtained from other frameworks are necessary steps.

1.3.2 Scalability

As the current systems available do not provide much flexibility in the area of scaling summaries to larger set of documents, this framework will attempt to bridge that gap. Usage of hadoop based NLP tools and cloud resources in the design, augments handling of scalability issues better in this framework.

1.4 Roadmap

The remainder of this report discusses the design, implementation and analysis of the text summarization method. Section 2 and Section 3 describe the detailed design and implementation of the framework, respectively. Section 4 analyzes the summaries and evaluated against metrics. Text summaries obtained from other systems will be compared with our result set in this section. Section 5 provides future work and conclusion.

Chapter 2

Design

- How you designed your solution
- Rationale for decisions
- Compare and contrast design with other approaches (related work)

The general design for text summarization implementation will include several components. The key components in the three stages (*input matrix, SVD, sentence selection*) are explained below. A generic design involving components in each stage can be seen in Figure 3. NLP frameworks used in each of the stages are explained below.

2.1 Input matrix generation

In the input matrix generation phase, The input is text dataset which has to be summarized and the intermediate output is a sparse tf-idf matrix.

2.1.1 Dataset

Datasets for text summarization have to match certain criteria, the datasets publicly available and specific news related datesets do not provide content redundancy. Handling redundancy in documents is an important feature in our system. For this reason we use Apache Nutch, [8] an open-source web search engine to get similar datasets on a specific topic. Using nutch, we crawl the web and fetch content from blogs, articles and websites related to the topic. The segments obtained from the crawler are parsed and the required text is

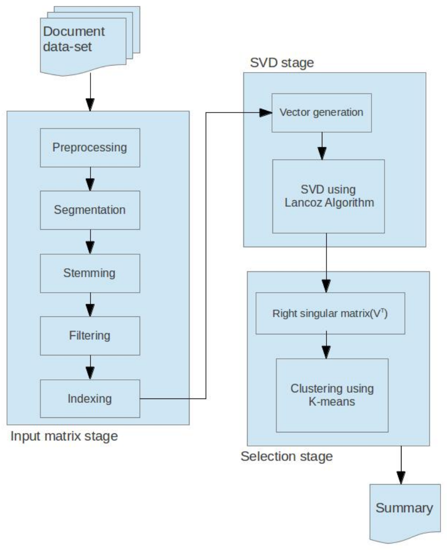

Figure 2.1: System Design.

extracted. Content obtained from web can be in different formats like .pdf, .txt, .html, raw text is extracted from these file formats using Tika [10].

2.1.2 TF-IDF matrix

Lucene API supports indexing of text in different formats like .html, .txt, several other document formats. Preprocessing will involve removal of html tags, extraction of text from different formats, and conversion of text from different languages to unicode. Segmentation of content is done based on groups of phrases, collection of sentences or single sentences. It is known that usage of single sentences provides better results. In this project, we will segment text based on senteces. Using different segmentation methods will affect the nature of matrix, i.e., it will be sparse or dense. Word grouping is done through stemming where words like “running,” “runner,” “run” are considered as “run”. Using lucene API, words and sentences are filtered to match pre-defined criteria. Lucene is used to index terms in each sentence. Lucene stores meta-data from documents in the form of key-value pairs, where key is the term and value is number of times it occurs in the sentence. Information related to sentences is provided in the index. The output of this phase is tf-idf sparse matrix which will be used for SVD.

2.2 SVD on input matrix

In this phase, SVD is used on the input matrix to obtain feature-sentence relation.

2.2.1 SVD using Mahout

Mahout API provides access to machine learning algorithms, which can be run on hadoop framework to parallelize the jobs. Indexed terms from each sentence are converted to vectors based on a selected weighting metric (*tf-idf, tf, log-entropy*). Lancoz algorithm which is used for eigen value decomposition is modified to obtain SVD term vectors and extracted features-sentences matrix (V^T).In the sentence selection, the extracted V^T matrix is used against different metrics to obtain different sentence selection processes. Some of the procedures followed in this report are clustering using K-Means, Extracted sentence threshold. Based on different sentence selection approaches we end up with different group of sentences being included into the summary.

2.2.2 Parallelization strategy

Right and left singular eigen vectors can be obtained from matrix A by matrix multiplication AA^T, A^TA respectively. If A is not a square matrix, a few optimizations have to be done to make it a square matrix. Matrix multiplication can be parallelized very effectively as a specific row does not have dependency over other rows or columns. Map-reduce framework will be used to parallelize the indexing of documents and the lancoz algorithm. As most of the underlying algorithms, be it Lancoz, matrix multiplication, KMeans clustering all of these are modified to take advantage of hadoop framework.

2.3 Evaluation

The obtained summary from system has to be evaluated against proven metrics and models to determine the quality compared to required standards. A summary can be categorized as good one if it reflects the tone and content of the original report. In this system, we evaluate the obtained summary against ROUGE-N, ROUGE-LCS and Tesla summarization metrics. Each of these metrics compares the extracted summary against summaries extracted by humans. Rouge-N and Tesla are N-gram based recall measures and ROUGE-LCS uses longest common subsequence of extracted words for benchmarking. Detailed explanation of the metrics is provided in chapter 4. Analysis.

2.4 Computation resources

For better performance, speed and consistency, experiments were run on Amazon EC2 AMIs. All the experiments had been conducted on Ubuntu 12.04 LTS (precise), and host will have minimum memory of 6GB. Performance and speed of algorithms on multiple EC2 instances has been studied.

Chapter 3

Implementation

(Note: this chapter may be merged with Chapter 2 to have a combined Design and Implementation chapter, if more appropriate.)

- Software details (use as many section as needed for class design, database tables, middleware, etc.)
- Make sure you present and comment on any interesting issues about your implementation that you are proud of or unhappy with
- Skip code listing and specific UML diagrams, etc. to an appendix

Chapter 4

Analysis

- How did you analyze your hyptothesis? Experiments, what did you think were worth measuring, etc.
- Based on your measurements and qualititative analyses, how well did your approach work out?
- Use graphs, tables, and other diagrams to illustrate your analyses.
- Based on your analyses, how well does your implementation or approach match your hypothesis?
- What do you deduce from this effort? How would you change or tweak your hypothesis?

Chapter 5

Conclusions

The conclusions chapter usually includes the following sections.

5.1 Current Status

5.2 Future Work

5.3 Lessons Learned

Since I need to illustrate several items in the bibliography, I'll do a cite for these references [**?**, **?**, **?**, **?**, **?**].

Bibliography

[1] Rachit Arora and Balaraman Ravindran. Latent dirichlet allocation and singular value decomposition based multi-document summarization. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 713–718, Washington, DC, USA, 2008. IEEE Computer Society.

[2] John M. Conroy and Dianne P. O'leary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 406–407, New York, NY, USA, 2001. ACM.

[3] Hal Daumé, III and Daniel Marcu. Bayesian query-focused summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 305–312, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[4] H. P. Edmundson. New methods in automatic extracting. *J. ACM*, 16(2):264–285, April 1969.

[5] David A. Ferrucci. Ibm's watson/deepqa. *SIGARCH Comput. Archit. News*, 39(3):–, June 2011.

[6] Sanda Harabagiu and Finley Lacatusu. Topic themes for multi-document summarization. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 202–209, New York, NY, USA, 2005. ACM.

[7] Erik Hatcher and Otis Gospodnetic. *Lucene in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA, 2004.

[8] Rohit Khare and Doug Cutting. Nutch: A flexible and scalable open-source web search engine. Technical report, 2004.

[9] H. P. Luhn. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165, April 1958.

[10] Chris Mattmann and Jukka Zitting. *Tika in Action*. Manning Publications Co., Greenwich, CT, USA, 2011.

[11] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in Action*. Manning Publications Co., Greenwich, CT, USA, 2011.

Appendix A

UML Diagrams

This is an optional appendix and can be eliminated if you don't have anything to share here.

Appendix B

Code Listing

This is an optional appendix and can be eliminated if you don’t have anything to share here.

Appendix C

User Manual

This is an optional appendix and can be eliminated if you don't have anything to share here.