Abstract

In this age of the Internet, with scalable compute and storage resources being easily accessible, users are surrounded with information. The existing information retrieval tools provide users with extensive data, which at times can become redundant and unnecessary. However, with extensive usage of this available data, a secondary level of wrappers like Natural Language Processing (NLP) frameworks that interact with data sets have become necessary. These wrappers must extract a concise summary from the primary data set gathered. The main reason 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 usage of machine learning (ML) algorithms for generic text summarization and compares it with other summary models available. It proposes text summarization using ML algorithms with usage of open-source NLP frameworks like Mahout and Lucene can be scaled efficiently. The performance of these Hadoop based algorithms is compared with other models commonly used for summarization. The goal is also to extend the evaluation to multi-lingual documents along with retaining quality and scalability of summary model.
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 [10] or small set of articles on specific topics [8]. 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 tries to resolve issues in MDS by using NLP techniques and open-source machine learning frameworks like Mahout [15] and Lucene [7].

2 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.

2.1 What is Text summarization?

A good summary should be concise 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 sum-
mary.
Sentence and sentiment analysis is in the developing stage, and hence, abstractive summary gener-
ation 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 sum-
mary. In this project, we will work towards obtaining well-defined extractive generic summaries.

2.2 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 differ-
ent 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 sen-
tence 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 will address these 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 infor-
mation. 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.

2.3 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, accessibility to those documents is limited. End users would like to get some infor-
mation 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 Hindi documents.

3 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* [10]. 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.
3.1 Statistical approach

In the statistical approach, features in sentences are selected based on concepts like co-occurrences of words and classification. Summarist [9] 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 [14] 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.

3.2 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 cosine 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, [16] which serves to index and search web pages. Summarization systems like TextRank [12], LexRank [17], Hypersum [19] use this approach for text. These systems obtain good summaries but get complicated when introduced to MDS.

3.3 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 trees, 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 [13]. 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.

3.3.1 LSA in summarization

Latent Semantic analysis is the most prominent algebraic 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 [18] 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).

\[ A = U \Sigma V^T \]  

A: Input matrix \((m\times n)\)  
U: Orthogonal left singular matrix \((m\times k)\)  
\(\Sigma\): Scalar values matrix \((k\times k)\)  
\(V^T\): Orthogonal right singular matrix \((k\times n)\)

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 \((m \times n)\) 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 $\Sigma$ 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 [6], Gong discusses usage of sentences with highest value for each feature. Steinberger et al. [18] used top k sentences with sum of feature values. Clustering algorithms can be used to select sentences similar to centroids.

### 4 Hypothesis

The hypothesis of this thesis 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 terms of scalability.

#### 4.1 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. Comparative study of hadoopified versions of LSA and LDA based summary models is done along with generic summary models TexRank, lexrank.

#### 4.2 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. LSA model (vs) LDA model evaluation in terms of scalability is done.
5 Solution

5.1 Design

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.

5.1.1 Datasets

In this project, experiments will be conducted on both English and Hindi data sets. Relevant data which will have redundant information and can be useful to assess the model has to be chosen. For this reason instead of using public datasets currently available, data extraction will be done from scratch. Nutch API will be used to extract and filter data from web. As there are no data sets available in hindi, we will use documents written in hindi as our data set.

5.1.2 Computation resources

For better performance and speed, experiments will be run on Amazon EC2 AMIs. All the experiments will be conducted on Linux OS, and host will have minimum memory of 6GB. Performance and speed of algorithms on multiple EC2 instances will be studied.

5.2 Implementation

The project will be implemented using Java programming language and evaluated using Perl. Lucene and Mahout are Java APIs which will be used for several information extraction algorithms (as described in 5.2.1 and 5.2.2). Other open-source java-based APIs like Tika and Open-NLP will be used. For evaluation purposes, ROUGE tool-kit will be used.

5.2.1 Lucene

Lucene API supports indexing of text in different formats like .html, .txt, several other document formats. PDF documents also can be indexed with the support of an external API named Tika [11]. Lucene API will be used for all the components in the input matrix stage. 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 can be 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 sentences. 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.
In indexing step, lucene will be 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.

5.2.2 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). In the SVD stage, lancoz algorithm will be used on the term vectors to obtain $V^T$. In the sentence selection, k-means algorithm is implemented using mahout API to obtain k clusters. Sentences from the k clusters are picked based on the similarity of each sentence to its centroid.

5.2.3 Parallelization strategy

Right and left singular eigen vectors can be obtained from matrix A by matrix multiplication $AA^T$, $A^T A$ 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.

5.3 Evaluation

Summary evaluation is an important step in understanding how the system has performed. In this thesis, we will be comparing the results from our system against summaries from tools like lexrank and textrank. The summary obtained will be analyzed using Perl based ROUGE tool-kit [20] and Tesla-n tool kit written in Python. ROUGE n-gram scores and ROUGE LCS scores for different NLP approaches will be compared.

6 Roadmap

6.1 Deliverables

- A thesis report consisting of design, implementation details, experiments, results and evaluation of summaries
- The source code
- Javadoc
- A manual for using the text summarization system
- A presentation on Thesis defense

6.2 Schedule

- Completion of implementation - Second week of October,
- Completion of experiments on local corpora - Third week of November,
- Evaluation and comparisons of summaries - Fourth week of January,
- Formal project proposal submission - Third week of March,
- Thesis defence - Second week of April.
References


