A Project Report

On

# Automatic Text Summarization

Submitted in partial fulfilment of the requirement of University of Delhi for the Degree of

**Master of Science**

*in*

**COMPUTER SCIENCE**

*Submitted by*

**Abhishek Gupta(Roll no. 1723902)**
**Yogesh Kumar(Roll no. 1723947)**

*Supervisor*

**Professor Vasudha Bhatnagar**

**DEPARTMENT OF COMPUTER SCIENCE**
**UNIVERSITY OF DELHI**

**Academic Year 2018-19**

# ACKNOWLEDGEMENT

We are thankful to our supervisor, **Prof.Vasudha Bhatnagar**, Head of Department, Department of Computer Science, University of Delhi, for her invaluable contributions and generous suggestions towards the completion of this project. She has always strived to provide us with the best of her capability. We feel privileged and honored to have worked under her. She has guided us at each step with her immense pool of knowledge and expertise of the topic. We are also immensely grateful to the lab staff of Department of Computer Science, University of Delhi, for providing us with lab facilities for carrying out the experiments. We truly believe that this project would not have been possible without the support of all of them.

**Project Group Members:**

Abhishek Gupta(Roll no. 1723902): ________________

Yogesh Kumar(Roll no. 1723947) : ________________

# DECLARATION

We hereby declare that the project work entitled **AUTOMATIC TEXT SUMMARIZATION** being submitted in the partial fulfillment of the requirement for the award of the degree of M.Sc. (Computer Science), is a record of original and bonafide work carried out by the undersigned in the Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi, New Delhi, India. The work presented in this Project has not been submitted to any other Institute or University for the award of any degree or diploma.

**Project Group Members:**

Abhishek Gupta(Roll no. 1723902): ____________

Yogesh Kumar(Roll no. 1723947) : ____________

# CERTIFICATE

The project entitled **AUTOMATIC TEXT SUMMARIZATION** is being submitted in the partial fulfillment of the requirement for the award of the degree of M.Sc.(Computer Science), is a record of original and bonafide work carried out by **Abhishek Gupta(Roll no. 1723902)** and **Yogesh Kumar(Roll no. 1723947)** under the supervision of Prof.Vasudha Bhatnagar in the Department of Computer Science, Faculty of Mathematical Sciences, University of Delhi, New Delhi, India. The work presented in this Project has not been submitted to any other Institute or University for the award of any degree or diploma.

**Professor Vasudha Bhatnagar**
Supervisor
Department of Computer Science
Faculty of Mathematical Sciences
University of Delhi

**Professor Vasudha Bhatnagar**
Head,
Department of Computer Science
Faculty of Mathematical Sciences
University of Delhi

# ABSTRACT

There is an enormous amount of textual material available in digital form, comprising of web pages, news articles, blogs etc which is growing exponentially. Manual summarization of a text is a tedious and laborious process. This creates need for automated process to handle the problem. Automatic summarization is a process of condensing text document using a computer algorithm which retains its salient details. Generated summaries help users to read a large document in a concise manner and check whether a document is relevant for him/her after reading the summary.

In this project, we study and implement entailment based text summarization method with minor variations in the original algorithm. We further study the Latent Semantic Analysis (LSA) based algorithm for automatic text summarization. We implement most recent LSA based Topic method followed by the studying the effect of stop words at preprocessing step. We also study graph-based TextRank, CoRank algorithms and implement these with different similarity measures. Performance of all studied algorithms is evaluated on DUC 2002 data-set. We use standard ROUGE tool for performance evaluation of studied algorithms.

# Contents

# List of Tables

# Chapter 1

# Introduction

The advent of Internet has yielded a massive increase in the amount of information available online in the form of text documents, blogs, scientific papers, news article, etc. Due to the excessive amount of information, it has become infeasible to manually extract the important content from the text available online. Thus, it has become a necessity to use automatic methods which can efficiently retrieve useful information.

Automatic text summarization is a process of condensing text while preserving the important content of the text. Automatic summarization techniques are either extractive or abstractive. In abstractive technique, summary is created by paraphrasing/rephrasing the important information contained in the text document. On the other hand extractive techniques work by selecting important sentences from the original text to create a meaningful summary. Extractive techniques are more popular than abstractive techniques because abstractive techniques require the understanding and generation of text, which is challenging.

Our objective is to study, implement and experiment with different extractive text summarization algorithms and compare their results. In the minor project, we studied and implemented entailment based text summarization algorithm proposed in the research paper, Text Summarization through Entailment-based Minimum Vertex **(Gupta, Kaur, Mirkin, Singh, & Goyal, 2014)**. In this paper, authors formulate the text summarization problem as a graph-based optimization problem, a weighted minimum vertex cover which uses textual entailment scores to construct the graph. In this work, we further explored graph properties (indegree, outdegree) on entailment graph to find different approaches for generating summaries.

Then we focus on Latent Semantic Analysis (LSA) based text summarization algorithms. LSA is an algebraic-statistical method that reveals hidden semantic structures of text documents. We studied an LSA based research paper, Text Summarization of Turkish Texts using Latent Semantic Analysis **(Ozsoy, Cicekli, & Alpaslan, 2010)**. The method described in this paper first construct $term \times sentence$ matrix of input text document. After applying LSA on this matrix, it considers extracted concepts as subtopics of the text document then find main topics by using these subtopics to select summary sentences from the main topics.

We further studied graph based text summarization methods. We studied Word-Sentence Co-Ranking for Automatic Extractive Text Summarization **(Fang, Mu, Deng, & Wu, 2017)**. This paper described a method named CoRank. CoRank is an improvement over an existing TextRank **(Mihalcea & Tarau, 2004)** algorithm. TextRank considers only the sentences as a unit to rank the sentences. CoRank overcomes this limitation of TextRank by incorporating the word-sentence relationship to rank the sentences.

The studied research papers are described in chapter 2. We implement all the described methods and compute their results on DUC 2002 data-sets. Implementation, software-hardware used, data-set, evaluation methods, and results are detailed in chapter 3.

# Chapter 2

# Related Work

In this chapter, we describe the algorithms that we study as part of our major project.

## 2.1 Text Summarization using Entailment-Based Minimum Vertex Cover

### 2.1.1 Text Entailment

Textual Entailment (TE) is an asymmetric relation between two text fragments specifying whether one fragment can be inferred from the other. In the TE framework, the entailing and entailed texts are termed $text(t)$ and $hypothesis(h)$, respectively. Textual entailment can be illustrated with examples of three different relations:

1. An example of a positive TE (text entails hypothesis) is:
    - text: If you help the needy, God will reward you.
    - hypothesis: Giving money to a poor man has good consequences.
2. An example of a negative TE (text contradicts hypothesis) is:
    - text: If you help the needy, God will reward you.
    - hypothesis: Giving money to a poor man has no consequences.
3. An example of a non-TE (text does not entail nor contradict) is:
    - text: If you help the needy, God will reward you.

- hypothesis: Giving money to a poor man will make you a better person.

Textual Entailment was exploited for text summarization in order to find the highly connected sentences in the document.

Authors (Gupta et al., 2014) formulate the text summarization task as a Weighted Minimum Vertex Cover (wMVC) problem.

The proposed method consists of the following main steps:

1. Intra-sentence Textual Entailment Score Computation.
2. Entailment-Based Connectivity Scoring.
3. Entailment Connectivity Graph Construction.
4. Find minimum weighted vertex cover(summary) of graph.

We have described each step in the following section.

### 2.1.2 Intra-sentence Textual Entailment Score Computation

Consider a document $D$ to be summarized consists of $N$ sentences $(S_1, S_2, ...., S_N)$. Then, TE score computed between each pair of sentences in $D$ using a textual entailment tool. TE scores for all the pairs of sentences are stored in an $N \times N$ sentence entailment matrix $(SE_{N \times N})$. Each entry $SE[i, j]$ in the matrix represents the extent by which sentence $S_i$ entails sentence $S_j$ that is how much sentence $S_i$ in document $D$ infer sentence $S_j$.

### 2.1.3 Entailment-Based Connectivity Scoring

Entailment between sentences indicates connectivity, so it is an indicator of sentence salience. More specially, the salience of a sentence is determined by the degree by which it entails other sentences in the document. By using the sentence entailment matrix authors compute the connectivity score for each sentence by summing the entailment scores of the sentence with respect to the rest of the sentences in the document and denote this sum as $ConnScore$. Applying it to each sentence in the document to obtain the $ConnScore_{1 \times N}$ vector where $N$ is the number of sentences in the document.

### 2.1.4 Entailment Connectivity Graph Construction.

The input document to be summarized is represented as an undirected weighted graph $G = (V, E, w)$, where a node(vertex) $v \epsilon V$ corresponds to a sentence in the document; an edge $(u, v) \epsilon E$ exists if either $u$ entails $v$ or $v$ entails $u$ with a value at least as high as an empirically-set threshold. A weight $w$ is then assigned to each sentence based on $ConnScore$ vector, i.e. $ConnScore[i]$ is the weight($w$) of the $i^{th}$ node in the graph.

### 2.1.5 Find minimum weighted vertex cover(summary) of graph

**Weighted Minimum Vertex Cover**

Weighted Minimum Vertex Cover (wMVC)is a combinatorial optimization problem listed within the classical NP-complete problems(Garey and Johnson, 1979; Cormen et al., 2001). A Vertex Cover of a connected undirected weighted graph $G$ is a subset of vertices $V$ of $G$ such that for every edge in $G$, at least one of its endpoints is in $V$. A Weighted Minimum Vertex Cover (wMVC) of $G$ is a Vertex Cover that has the smallest total weight among all possible Vertex Covers. A graph can have multiple Vertex Covers but the value of wMVC is unique.

**wMVC for text summarization**

After applying wMVC on graph G, it returns a cover C which is a subset of the sentences with a minimum total weight, corresponding to the best connected sentences in the document. The cover represents the summary of the input document.

## 2.2 LSA based Text Summarization

Latent Semantic Analysis is a technique in natural language processing which uncover hidden semantic structures of terms and sentences in a text document. LSA technique is capable of capturing and modeling interrelationships among terms so that it can semantically cluster terms and sentences(Gong & Liu, 2001). LSA is a unsupervised technique based on singular value decomposition which is domain and language independent. LSA based summarization methods follow steps discussed below:

### 2.2.1 Input matrix creation

An input document needs to be represented in a way that enables a computer to understand and perform calculations on it. This representation is usually a matrix representation where columns are sentences and rows are words/phrases. The cells are used to represent the importance of words in sentences.

The process starts with the creation of a terms by sentences matrix $A = [A_1\ A_2\ :::\ A_n]$ with each column vector $A_i$ represents the weighted term-frequency vector of sentence $i$ in the document under consideration. If there are a total of $m$ terms and $n$ sentences in the document, then we will have an $m \times n$ matrix $A$ for the input document. Since every word does not normally appear in each sentence, the matrix $A$ is usually spars. There are different approaches to filling out the cell values of matrix $A$.

### 2.2.2 Singular Value Decomposition:

Singular value decomposition is a mathematical method which models the relationships among terms and sentences. It decomposes the input matrix into three other matrices as follows:

$$A = U\Sigma V^T \tag{2.1}$$

where $A$ is the input matrix with dimensions $m \times n$, $U$ is an $m \times n$ matrix which represents the description of the original rows of the input matrix as a vector of extracted concepts, $\sigma$ is an $n \times n$ diagonal matrix containing scaling values sorted in descending order, and $V$ is an $m \times n$ matrix which represents the description of the original columns of input matrix as a vector of the extracted concepts.

### 2.2.3 Sentence selection:

After decomposing input term-sentence matrix using SVD various sentence selection methods can be used for selecting final summary sentences.

We have studied the Topic method described in the research paper, Text Summarization of Turkish Texts using Latent Semantic Analysis (Ozsoy et al., 2010). This method is described in next section.

### Topic method

In this approach, input matrix creation and SVD calculation steps are used as described above and the $(concept \times sentences)$, $V^T$ matrix is used for selecting summary sentences. The main idea in the topic method is to find out the main concepts and sub-concepts. The resulting concepts extracted from SVD calculations are known to be the topics of the input document. However, these topics can be sub-topics of other extracted topics as well. In this approach, after deciding the main topics that may be a group of subtopics, the sentences are collected from the main topics as a part of the summary.

Between the SVD calculation step and the sentence selection step, there exists a pre-processing step. The aim of the pre-processing step is to remove the overall effect of sentences that are related to the concept somehow, but not the core sentence for that concept. For each concept, which is represented by the rows of the $V^T$ matrix, the average sentence score is calculated. Then the cell values which are less than or equal to the average score are set to zero as shown below:

$$V'^T[i,j] = \begin{cases} if\ (V^T[i,j] < mean(i))\ then\ V'^T[i,j] = 0 \\ else\ V'^T[i,j] = V[i,j] \end{cases} \tag{2.2}$$

Here $mean(i)$ is the mean value of row $i$ in matrix $V^T$. This step removes sentences that are not highly related to the concept, leaving only the most important sentences related to that concept.

After the step of pre-processing comes the step of finding out the main topics. For this step, a $(concept \times concept)$ matrix is created by finding out the concepts that have common sentences. The common sentences are the ones that have cell values other than zero in both concepts that are considered. Then the new cell values of the $(concept \times concept)$ matrix are set to the total of common sentence scores. After the creation of the $(concept \times concept)$ matrix, the strength of each concept is calculated. For each concept, the strength value is computed by getting the cumulative cell values for each row of the $c(oncept \times concept)$ matrix. The concept with the highest strength value is chosen as the main topic of the input document. A higher strength value indicates that the concept is much more related to the other concepts, and it is one of the main topics of the input text.

After these steps, the sentences are collected from the pre-processed $V'^T$ matrix, a single sentence is collected from each main concept until pre-defined numbers of sentences are collected.

## 2.3 Graph based Text Summarization

A graph-based ranking algorithm is a way of deciding the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. We implement two well known graph based text summarization methods TextRank and CoRank. Both of these methods based on the concept of PageRank (Page, Brin, Motwani, & Winograd, 1999) algorithm. The PageRank is a graph-based algorithm that outputs a probability distribution used to represent the likelihood that a person randomly clicking on links of a web page will arrive at any particular page.PageRank can be calculated for collections of documents of any size.

Formally, let $G = (V, E)$ be a directed graph with the set of vertices $V$ and set of edges $E$, where $E$ is a subset of $V \times V$. For a given vertex $V_i$, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex $V_i$ points to (successors). The score of a vertex is defined as follows (Brin and Page, 1998):

$$S(V_i) = (1 - d) + d \times \sum_{j \epsilon (V_i)} \frac{1}{|Out(V_j)|} S(V_j) \tag{2.3}$$

where $d$ is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. In the context of Web surfing, this graph-based ranking algorithm implements the "random surfer model", where a user clicks on links at random with a probability $d$, and jumps to a completely new page with probability $1 - d$. The factor $d$ is usually set to 0.85 (Brin and Page, 1998). Starting from arbitrary values assigned to each node in the graph, the computation iterates until convergence below a given threshold is achieved.

### 2.3.1 The TextRank Model

To enable the application of graph-based ranking algorithms to natural language texts, a text document considered as an array of sentences in which each sentence represents a node in the graph and the interconnections(edges) between the sentences represented by the different similarity measures. Hence it is a weighted undirected graph. Here strength between the vertices (sentences) $V_i$ and $V_j$ is represented as a weight $a_{ij}$.

Rada Mihalcea et al. (Mihalcea & Tarau, 2004) introduce a new formula for graph-based ranking that takes into account edge weights when computing the score associated with a vertex in the graph.

$$P_i = (1-d) + d \sum_{w_{ij} \neq 0} \frac{a_{ij}}{\sum_k a_{jk}} P_j \tag{2.4}$$

After the fixed number of iterations, $P_i$ is the final vector containing the ranking of each node(sentence).

The algorithm consists of the following steps:

1. Represent the text document as an array of sentences.
2. Identify relations that connect sentences(vertices), and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3. Iterate the algorithm until convergence.
4. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection the sentences.

### 2.3.2 The CoRank Model

TextRank algorithm does not consider the importance of words, i.e., assuming the weights of every word are equal. Obviously, the importance of words in a document varies significantly. For example, a list of keywords of a scientific paper can best reflect the topic, whereas stop words are not indicative of topics. So Changjian Fang et al. (Fang et al., 2017) believed that the mutual influence between sentences and words does exist and will boost the sentence scoring. To be specific, the mutual influence here means that: (i) the words appearing in high-weight sentences deserve high weights themselves; conversely (ii) a sentence containing more high-weight words should be marked a higher score.

Under the bag-of-words model, a document $T$ is usually represented by a $n \times m$ matrix $C = [C_{ij}]$, $1 \leq i \leq n, 1 \leq j \leq m$, where $c_{ij}$ is the count of $j$th word in $i$th sentence. To facilitate the sentence ranking, an undirected weighted graph is often built among sentences, and thus a PageRank-like algorithm, such as TextRank, can be run on this graph to score sentences. Mathematically, the adjacency matrix of the sentence graph is denoted as

$A = [a_{ij}], 1 \leq i, j \leq n$, where $a_{ij}$ is the similarity between $i$th and $j$th sentence. In this research paper (Fang et al., 2017) author define the weight $a_{ij}$ by using jaccard similarity.

$$a_{ij} = Jaccard(S_i, S_j) = \frac{|W_i \cap W_j|}{|W_i \cup W_j|} \tag{2.5}$$

where $W_i$ is the set of words contained in $i$th sentence. In the CoRank model, the authors aim to incorporate the word-sentence relationship into the sentence graph for enhancing the sentence scoring. In other words, CoRank tries to make full use of the mutual feedback between words and sentences. For that authors define, $Q_j$, $1 \leq j \leq m$ is the weight of the $j$th word. Thus $Q_j$ can be updated by using the weights of sentences that contain the $j$th word.

$$Q_j = \frac{\sum_{i=1}^{n} c_{ij} P_i}{\sum_{i=1}^{n} c_{ij}} \tag{2.6}$$

where $P_i$ is the vector from Eq.(2.2) of TextRank. With the weight of every word, the calculation of sentence weight determined by words denoted as $P_i^*$.

$$P_i^* = \frac{\sum_{i=1}^{n} c_{ij} Q_j}{\sum_{i=1}^{n} c_{ij}} \tag{2.7}$$

Then, authors use a linear combination of $P_i$ by Eq.(2.2) and $P_i^*$ by Eq.(2.5) to obtain a new weight vector of each sentence for the next-round iteration.

$$P_i = \alpha P_i + (1 - \alpha) P_i^* \tag{2.8}$$

where $\alpha \, \epsilon \, [o, 1]$ is the coefficient that balances between the sentence graph and word-sentence relationship.

Compared with TextRank, the proposed CoRank integrates the sentence-weight led by the words, as shown in Eq.(2.5) and Eq.(2.6). This essentially implies that TextRank treats every word equivalently, i.e., all words have the same weight. However, in the CoRank model, the authors believe the importance of different words is distinct, just like the keywords highlighted in each paper. The steps of CoRank algorithms remain the same as of TextRank, only the final iterative equation is changed i.e equation 2.4 to 2.8.

# Chapter 3

# Implementation and Results

## 3.1 Hardware and Software Used

We use Ubuntu 18.04 OS on Oracle VM VirtualBox and Windows 10 with AMD A8-6410 processor and 8 GB of RAM for all our experiments. We use R and Python programming language for implementation of studied algorithms. For calculating textual entailment scores, we use BIUTEE (Stern & Dagan, 2012) tool based on EOP[1]. BIUTEE is a transformation-based Textual Entailment system to compute textual entailment scores between pairs of sentences. BIUTEE was trained with 800 text hypothesis pairs.

## 3.2 Datasets

We use DUC 2002[2] dataset for evaluating the performance of studied algorithm and their validation. DUC 2002 dataset consists of 567 documents (however there are 533 unique documents) and each document has 1 to 3 reference summaries. Reference summaries are abstractive and approximately 100 words in length.

## 3.3 Evaluation Method

ROUGE (Lin, 2004) is an evaluation toolkit for evaluating the performance of system generated summaries against a set of reference summaries.

[1]https://github.com/hltfbk/EOP-1.2.3
[2]https://duc.nist.gov/data.html

ROUGE generates recall, precision and F measure for performance evaluation of algorithmic (system) generated summaries with reference summaries. We use ROUGE 1.5.5 version for evaluation of algorithmic summaries.

ROUGE is based on n-gram co-occurrence, the longest common subsequence and the weighted longest common subsequence between the ideal summary and the extracted summary. These are defined as follows.

- ROUGE-n: This metric is based on a comparison of n-grams. A series of n-grams (mostly two or three but rarely four) is elicited from the reference summaries and the system summary (algorithmic summary). Let $p$ be "the number of common n-grams between system and reference summary", and $q$ be "the number of n-grams extracted from the reference summary only". The score is computed as:

  $$ROUGE - n = \frac{p}{q}$$

- ROUGE-L: This measure employs the concept of longest common subsequence (LCS) between the two sequences of text. The intuition is that the longer the LCS between two summary sentences, the more similar they are. Although this metric is more flexible than the previous one, it has a drawback that all n-grams must be consecutive.

- ROUGE-S: It is any pair of word in a sentence in order, allowing for arbitrary gaps. This can also be called skip-gram coocurrence. For example, skip-bigram measures the overlap of word pairs that can have a maximum of two gaps in between words. As an example, for the phrase "cat in the hat" the skip-bigrams would be "cat in, cat the, cat hat, in the, in hat, the hat".

## 3.4 Performance evaluation of Entailment based methods

### 3.4.1 wMVC Result:

We implemented entailment based wMVC (Gupta et al., 2014) in our minor project. Result of evaluation of wMVC on DUC2002 are presented in Table 3.1. We generate 100 words summaries for all DUC2002 documents. We used mean as threshold to construct the graph as described in section 2.1.4.

| **ROUGE** | **Precision** | **Recall** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4832 | 0.3728 | 0.4150 |
| ROUGE-2 | 0.2028 | 0.1566 | 0.1744 |
| ROUGE-L | 0.4518 | 0.3480 | 0.3878 |

Table 3.1: Result of wMVC on DUC 2002 data-set

Authors used random 60 documents and we used all documents of DUC2002 data-set. On comparing our results with authors we are getting better results for all metrics.

### 3.4.2 Variations of Original Algorithm

#### Graph Construction

Graph $G = (V, E, w)$ is a weighted graph with sentences as nodes ($V$). Edges ($E$) are drawn on the basis of relation among sentences. The edges are constructed as, if ($SE[i, j] > SE[j, i]$) then a directed edge from $i$ to $j$ is constructed otherwise $j$ to $i$ and a weight ($w$) is assigned as entailment score between $i$ and $j$ or $j$ and $i$ according to the directed edge constructed.

#### Interpretation of Outdegree of a Node in Entailment Graph

Outdegree of a node (here sentences) is the number of outgoing edges from the node. We consider weighted outdegree of a node on entailment graph calculated as sum of entailment scores of all outgoing edges. Weighted outdegree of a graph means how much a sentence entails the other sentences in the document. Higher is the weighted outdegree of a sentence more it entails the other sentences in the document. Higher is the weighted outdegree of a node (sentence) more eligible it to be included in the summary.

#### Steps for Document Summarization based on Outdegree notion in Entailment Graph are as follows

1. Construct entailment graph of document.
2. Calculated weighted outdegree of each sentence.
3. Arrange the sentences in decreasing order of Weighted outdegree.
4. Select top scoring sentences until the summary length is reached.

**Interpretation of Indegree of a Node in Entailment Graph**

Indegree of a node (here sentences) is the number of incoming edges from the node. We consider weighted indegree of a node on entailment graph calculated as sum of entailment scores of all incoming edges. Weighted indegree of a graph means how much this sentence entailed by other sentences in the document. Higher is the weighted indegree of a sentence more it entailed by other sentences in the document. Lower is the weighted indegree of a node (sentence) more eligible it to be included in the summary.

**Steps for Document Summarization based on Indegree notion in Entailment Graph are as follows**

1. Construct entailment graph of document.
2. Calculated weighted indegree of each sentence.
3. Arrange the sentences in increasing order of Weighted indegree.
4. Select top scoring sentences until the summary length is reached.

**Result:**

Table 3.2 and 3.3 gives result of evaluation based on outdegree and indegree of a sentence respectively.

| **ROUGE** | **Precision** | **Recall** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4285 | 0.3800 | 0.4017 |
| ROUGE-2 | 0.1498 | 0.1330 | 0.1405 |
| ROUGE-L | 0.3972 | 0.3513 | 0.3719 |

Table 3.2: Performance of DUC2002 summarization based on Weighted Outdegree of a node in entailment graph

| **ROUGE** | **Precision** | **Recall** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4435 | 0.3911 | 0.4145 |
| ROUGE-2 | 0.1539 | 0.1358 | 0.1439 |
| ROUGE-L | 0.3946 | 0.3490 | 0.3694 |

Table 3.3: Performance of DUC2002 summarization based on Weighted Indegree of a node in entailment graph

**Observations**

- For weighted outdegree and Indegree, we get almost the same result. This is because the intuition behind for ordering(ranking) the sentence based on weighted indegree or weighted outdegree is the same i.e. weighted outdegree method work the same as weighted indegree but conversily. Here we get the best precision, recall, f-score for ROUGE 1 for both of the methods.

- Both indegree and outdegree based summaries have large sentences. The large sentences contain more information than smaller sentences so larger sentences entail many other sentences which result to have higher weighted outdegree. That is why they included in the summary.

- Large sentences cannot be entailed easily as these sentences contain more information which can not easily entail. So larger sentences have a fewer number of sentences which are entailing them resulting lesser weighted indegree. That is why they included in the summary.

On comparing indegree/outdegree to wMVC, we find that wMVC performs better. But all the three methods rely on the textual entailment score computed using EOP[3] tool. For checking the quality of the selected sentences, we manually checked some sentences and found that the results were not good as we did not get the satisfactory scores between highly entailing sentence or weakly entailing sentences. Even for the same sentence as text and hypothesis, we are not getting perfect score i.e. 1. So we did not explore the graph properties for improving the algorithm and moved to another technique for text summarization.

## 3.5 Performance evaluation of LSA based Method

We implement the topic method (Ozsoy et al., 2010) because it is the most recent and had best performance among all the discussed LSA based methods. We further study the effect of stop-words on this method. First, we implement this method without removing the stop-words and then implement with removing stop-words at preprocessing step. The results of thses two variations on original Topic method are presented in Table 3.4 and Table 3.5.

[3]https://hltfbk.github.io/Excitement-Open-Platform/

**With stop word:**

| **ROUGE** | **Recall** | **Precision** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4687 | 0.4199 | 0.4418 |
| ROUGE-2 | 0.1855 | 0.1668 | 0.1753 |
| ROUGE-L | 0.3789 | 0.3397 | 0.3573 |

Table 3.4: Performance of Topic method (with stop-word) on DUC2002 dataset

**After removing stop words:**

| **ROUGE** | **Precision** | **Recall** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4643 | 0.4149 | 0.4373 |
| ROUGE-2 | 0.1813 | 0.1620 | 0.1708 |
| ROUGE-L | 0.3788 | 0.3397 | 0.3573 |

Table 3.5: Performance of Topic method (without stop-word) on DUC2002 dataset

**Observations:** From Table 3.4 and 3.5 we observe that all three ROUGE metrics have high average score when stop word are not removed at preprocessing step. So stop word are useful in LSA based text summarization.

## 3.6 Performance evaluation of graph based TextRank and CoRank:

We implement CoRank method as described in (Fang et al., 2017). The final CoRank iterative equation is given by equation 2.8. If we choose $\alpha = 1$ then this equation became $P_i = \alpha P_i$ which is the equation of TextRank algorithm. The authors of CoRank have conducted experiment with different values of $\alpha$ to incorporate word-sentence relationship. They got best result at $\alpha = 0.6$. So we implement TextRank and CoRank i.e $\alpha = 0$ for TextRank and $\alpha = 0.6$ for CoRank.

We study the variation in the algorithm by replacing jaccard similarity measure with cosine similarity. Result are described in Table 3.6 to 3.9.

**With jaccard similarity:**

| **ROUGE** | **Recall** | **Precision** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4658 | 0.3855 | 0.4207 |
| ROUGE-2 | 0.1887 | 0.1548 | 0.1696 |
| ROUGE-L | 0.4230 | 0.3501 | 0.3820 |

Table 3.6: Summary length is 100 words, DUC2002 dataset, TextRank

| **ROUGE** | **Recall** | **Precision** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4800 | 0.3921 | 0.4306 |
| ROUGE-2 | 0.2000 | 0.1629 | 0.1791 |
| ROUGE-L | 0.4322 | 0.3531 | 0.3877 |

Table 3.7: Summary length is 100 words, DUC2002 dataset, CoRank

**With cosine similarity:**

| **ROUGE** | **Recall** | **Precision** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4657 | 0.3854 | 0.4207 |
| ROUGE-2 | 0.1887 | 0.1548 | 0.1669 |
| ROUGE-L | 0.4322 | 0.3531 | 0.3871 |

Table 3.8: Summary length is 100 words, DUC2002 dataset, TextRank

| **ROUGE** | **Recall** | **Precision** | **F-Score** |
|---|---|---|---|
| ROUGE-1 | 0.4818 | 0.3936 | 0.4322 |
| ROUGE-2 | 0.2016 | 0.1640 | 0.1804 |
| ROUGE-L | 0.4338 | 0.3544 | 0.3891 |

Table 3.9: Summary length is 100 words, DUC2002 dataset, CoRank

**Observations:**

- In TextRank and CoRank for both of the cases, we get the highest precision, recall, and f-score in rouge 1.

- We get a slightly better result in case of cosine similarity for all ROUGE measures in case of CoRank. But the difference is very small i.e. we don't get significant improvement for cosine similarity over jaccard similarity. And for TextRank we get almost same result for both of the cases.

- CoRank outperforms TextRank in both of the cases. Hence by incorporating the word-sentence relationship in TextRank algorithm improved the result. This proved that the importance of different words is distinct and must be used to rank the sentences.

# Chapter 4

# Conclusions

Gupta Et al. presents a novel method for single document extractive summarization. They formulate the summarization task as an optimization problem and employ the weighted minimum vertex cover algorithm on a graph based on textual entailment relations between sentences. We implement this method in Python and R. This method has outperformed previous methods that employed TE for summarization. We experiment with graph properties to find some new variation to this method. For this we construct the graph as directed. Graph nodes are corresponds to the sentences in the documents and weighted directed edges are asymmetric relationships (Textual Entailment) between the sentences. In this graph we compute weighted outdegree of each node as the sum of weights of all outgoing edges. Higher is the weighted outdegree of a node (sentence) more eligible it to be included in the summary. Conversely for indegree, lesser is the weighted outdegree of a node (sentence) more eligible it to be included in the summary .

We used ROUGE 1.5.5 tool and DUC2002 dataset for evaluating the all algorithmic summaries in our study. For weighted outdegree and Indegree, we get almost the same result. This is because the intuition behind for ordering (ranking) the sentence based on weighted indegree or weighted outdegree is the same. Both indegree and outdegree based summaries have large sentences. The large sentences contain more information than smaller sentences so larger sentences entail many other sentences which result to have higher weighted outdegree. Large sentences cannot be entailed easily as these sentences contain more information which cannot easily entail. So larger sentences have a fewer number of sentences which are entailing them resulting lesser weighted indegree. That is why they included in the summary. On comparing results of indegree/outdegree to wMVC, we find that wMVC performs better. But all the three methods rely on the textual entailment score computed using EOP tool. EOP tool is not giving the satisfactory scores. So we did not explore the graph properties for improving the algorithm and moved to another technique for text summarization.

Latent Semantic Analysis is a technique in natural language processing which uncover hidden semantic structures of terms and sentences in a text document. LSA technique is capable of capturing and modeling interrelationships among terms so that it can semantically cluster terms and sentences. LSA is an unsupervised technique based on singular value decomposition which is domain and language independent. We implement the topic method because it is the most recent and had best performance among all the LSA based methods. The main idea in the topic method is to find out the main concepts and sub-concepts. The resulting concepts extracted from SVD calculations are known to be the topics of the input document. However, these topics can be sub-topics of other extracted topics as well. In this approach, after deciding the main topics that may be a group of subtopics, the sentences are collected from the main topics as a part of the summary. This method includes a pre-processing step; its aim is to remove the overall effect of sentences that are related to the concept somehow, but not the core sentence for that concept. For finding the main topics, a ($concept \times concept$) matrix is created by finding out the concepts that have common sentences. The common sentences are the ones that have cell values other than zero in both concepts that are considered. We study the effect of stop-words on this method. First, we implement this method without removing the stop-words and then implement with removing stop-words at preprocessing step. After removing the stop word we get slight decrement in results. So stop word are useful in LSA based text summarization.

CoRank is a novel word-sentence co-ranking model for automatic extractive text summarization. CoRank is an improvement to an existing algorithm TextRank. TextRank only consider the sentences as a unit to rank the sentences. Authors of CoRank overcome this limitation by defining relationship between the words and sentences. CoRank combines the word-sentence relationship with the graph based ranking model. With the assumption that different words should have biased weights, the procedure of CoRank can be viewed as a mutual reinforcement between words and sentences. That is, sentence scoring derives the word scoring that in turn enhances the sentence scoring. Also, CoRank can be represented as a concise matrix notation, based on which the convergence can be theoretically guaranteed. This equation linearly adds TextRank and word-sentence relationship. We get better result for all the metrics in case of CoRank than the TextRank. By incorporating the word-sentence relationship to TextRank improves the results. We further study the variation in the CoRank algorithm by replacing jaccard similarity measure with cosine similarity. After that we get very slight improvement. So after adding word-sentence relationship to TextRank and using cosine similarity we get best result for all of the discussed methods.

# References

Fang, C., Mu, D., Deng, Z., & Wu, Z. (2017). Word-sentence co-ranking for automatic extractive text summarization. *Expert Systems with Applications*, *72*, 189–195.

Gong, Y., & Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 19–25). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/383952.383955` doi: 10.1145/383952.383955

Gupta, A., Kaur, M., Mirkin, S., Singh, A., & Goyal, A. (2014). Text summarization through entailment-based minimum vertex cover. In *Proceedings of the third joint conference on lexical and computational semantics (* sem 2014)* (pp. 75–80).

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Mihalcea, R., & Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing.*

Ozsoy, M. G., Cicekli, I., & Alpaslan, F. N. (2010). Text summarization of turkish texts using latent semantic analysis. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 869–876).

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The pagerank citation ranking: Bringing order to the web.* (Tech. Rep.). Stanford InfoLab.

Stern, A., & Dagan, I. (2012). Biutee: A modular open-source system for recognizing textual entailment. In *Association for computational linguistics* (p. 73).