EXECUTIVE SUMMARY

An Abstract is a shortened version of a large document and provides useful and vital information about the document. With the overwhelming outcome of research papers on the web, extracting relevant paper of our choice demands us to go through the entire paper to check for relevance which is time consuming and involves manpower [1]. The beauty of an abstract lies in the fact that a short paragraph highlights the entire gist of the paper, thus reducing the time involved in digesting the paper.

The aim of the project is to develop Automatic Text summarizer which would generate summaries/ abstract using a novel technique in the field of Automatic Text summarization and compare the performance of the summarizer with some of the traditional techniques. The technique we are employing is a Network approach wherein a text document is converted into large networks and we analyse these large networks using an efficient community detection algorithm, *InfoMap*, a novel technique introduced for the first time in the field of Text summarization. Community detection algorithms aid in detecting communities in a graph and the nodes under each community share some common trait [4].

The research started way back in the late 1950's [2] and with the massive increase in the amount of information, motivated many researches to come up with an automated text summarizer which would extract vital information from huge documents. The project is predominantly research oriented (Type I) and involves lot of research in the field of automatic text summarization, Graph Theory and Community Detection Algorithm along with a significant amount of programming too. And the project aims to better off the efficiency of some of the existing text summarizers. The novelty of the project is the introduction of community detection algorithm, *InfoMap*. In case of communities in text network, it is assumed that sentences under each community try to *convey some vital information* about the document which is an added value. And selection of appropriate communities from the network aid in Abstract composition.

Contributions and Achievements:

- Created an Automated Text summarizer which aims at providing precise and concise information i.e. the summary of a huge text document employing an existing efficient community detection algorithm known as *InfoMap*.
- The approach is a novel attempt made in the field of text summarization which integrates the concepts of Text summarization, Graph Theory and Community detection algorithm.
- The report showcase the research carried out and the comparison and performance evaluation of four related Text summarizers versus our Text summarizer, see chapter 6.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The research started way back in the late 1950's [1] [2] and with the massive increase in the amount of information, motivated many researchers to come up with an automated text summarizer which would extract vital information from huge documents. More information became available with the advent of the Internet technology, and the information is now reachable by everyone and everywhere. As the available amount of information is increasing day by day this gave rise to search engines which aid user to access relevant information of their choice from the sea of information i.e. the Internet [1].

Here is an attempt made in developing an automated text summarizer which aid people in knowing the core content of a document without wanting the reader to go through the entire gist of the text document. The automated text summarizers employ various different techniques/ methods in knowing the keywords and extract only the relevant sentences which would convey the actual meaning of the whole document.

Summaries generated by the humans are subjective and tend to differ each time they are generated, and generation of summaries manually is a time consuming process and demands lot of man power. Thus an efficient one would be an automated one which does the job for us better and faster [1].

1.1 Motivation

The advent of the Internet Technology gave access to massive information over the Internet thus giving rise to Automated Text summarizers to brief the content of the document. Till date we have many automated summarizers based on different techniques and we now intent to develop one with high efficiency. The technique we are employing is a Network approach wherein a text document is converted into large networks and we analyse these large networks using an efficient community detection algorithm, a novel technique introduced for the first time in the field of Text summarization. Community detection algorithms aid in detecting communities/ clusters in a graph. The Text documents when converted to networks form a scale free network [8], and to identify highly connected hubs in these networks we need an efficient algorithm. Community detection algorithms detect communities, where the nodes under each community share some common trait. These communities aid in extraction of the keywords which later help in sentence extraction and to generate summary efficiently, thus motivating us to use this algorithm.

1.2 Aims and Objectives

The aim of our project is to develop automated text summarizer which convert large text documents to networks, analyse these networks, especially using *community detection algorithms*, with intent to extract the summary out of these networks and comparison of these results with the conventional methods.

The following are the main objectives:

- To convert large text documents to networks where the words represent a node and the adjacent words are linked via an edge [10].

- Employ community detection algorithms to detect communities/ clusters from complex networks and also explore various other approaches in the field of automated text summarization.

- Performance evaluation of the summarizers (summarizers using community detection algorithm versus other conventional techniques).

1.3 Thesis Outline

Chapter 1: Provides a brief introduction to Text summarization with the aims and objectives of the project along with the motivation for carrying out the project.

Chapter 2: Provides an insight on some of the traditional techniques in the field of Automated Text Summarization.

Chapter 3: Speaks about Community detection algorithms.

Chapter 4: Provides a detailed literature on various traditional text summarizers employed by us to evaluate the performance of our summarizer with them.

Chapter 5: Detailed information on the principle working of our summarizer.

Chapter 6: Provides information on the evaluation techniques employed to evaluate the performance of our summarizer.

Chapter 7: Concludes the project with a set of tasks accomplished for successful completion of the project.

Chapter 8: Portion which comment on further improving the summarizer.

CHAPTER 2

HISTORY

2.1 Introduction

The section covers some of the key works conducted in the field of Automated Text summarization. The research kick started with the works of Luhn in the year 1958 [1], with intent to shrink a large document to a smaller version.

In general there are 2 ways to accomplish this goal.

- Extraction
- Abstraction

Extraction: The process merely involves extraction of exact sentences from the main document which includes extraction of the vital sentences, phrases [1].

Abstraction: The process is much complex than extraction as it involves condensing the text and demands for Natural language generation technology. The process is time consuming when in comparison with the extraction process [1].

Most traditional techniques employ Extraction which is the mere extraction of the actual sentence from the document. The process of abstraction came to existence with the introduction of many pre-processing steps involving processes like stemming and so on. We also look into the Machine Learning concepts in the field of text summarization [1].

2.2 Traditional Techniques

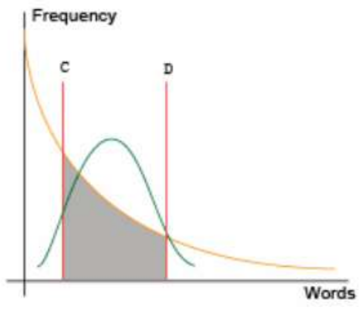
Automatic Abstract Generator was first introduced by Luhn known as “Father of Information Retrieval” in “*The Automatic Creation of Literature Abstract*" in the late 1950’s [1] [2]. Below is the technique employed by H.P. Luhn.

2.2.1 Luhn’s Technique

The key notion was to extract some of the sentences which are of importance from the document which would contribute as abstract when combined together. The assessment of the importance of the sentence is based on some of the assumptions as listed below.

1. The significance of the sentence is determined by the number of high frequency words in the sentence. It is expected that keywords are high frequency words and they appear in many places of the document. The frequency of all the words are computed from the whole document and the significance of the sentence are calculated based on the occurrence of these words in each sentence of the article. Luhn exhibits relationship between the word significance and their frequency. A graph [1] is plotted with

the words across the X-axis and the frequency of the word across the Y-axis, now Luhn determines two cut-offs C and D wherein the words to the left of C and the words to the right of D are excluded and the region between them are considered to be the significant ones. The 2 thresholds, limits the number of words wherein the words out of the threshold point C are considered to constitute words like 'in', 'into', 'by' etc. i.e. the most common words in a language which are the prepositions which are of no importance. And the words to the right of the threshold point D are the least occurring words in the article and are assumed to be insignificant.



____ Resolving Power of Significant Words
____ Words frequency

Figure 1: Frequency versus Word

The summarizer developed by Luhn was not so efficient and did contain many not so important sentences so a new technique gave into existence to better off the existing one by Edmundson.

2.2.2 H.P Edmundson Technique [3]

The method by Luhn did consider only the frequency of the words to determine the significance or the importance of the sentence in the document. Edmundson tries to build on Luhn' method by introducing 2 more evaluation features.

1. Key Word Extraction

The extraction of the keyword is same as that conducted by Luhn, and on assumption that frequency of a word is related to its significance. Now a vocabulary 'keyword vocabulary' is created with all the significant words ranked based on their frequencies arranged in descending order and the sentences are then weighted based on the ranking of the significant words in the sentence [1] [3].

2. Title Extraction

It is assumed that the titles and headings in a document summarize the content under it. Similar to the 'keyword vocabulary' a vocabulary is created for the titles and the headings and based on the frequencies of these words they are weighted. The weights assigned to these words are more when in comparison with the keywords in the 'keyword vocabulary'. Similar to the former method the weight of the sentence is computed by summation of the weights of the words in each sentence.

3. Cue Words

A dictionary is created with the pragmatic words in advance. The dictionary words are classified into 3 categories. 1. Null words 2. Bonus words 3. Stigma words. The words under each category are weighted differently and the weight of the sentence is computed by the summation of the weights of the cue words in the sentence.

The dictionary words are taken from hundreds of scientific papers. The words under these papers are weighted based on 3 categories [3] i.e. the Frequency, Selection ratio and the Dispersion. Frequency is defined as the total occurrence of the word in all the documents. Dispersion is defined as the number of paper containing the word and finally the Selection ratio is defined as ratio of the frequency of the word in the sentence to the frequency in the entire document. For classification of the words 3 experimental thresholds are picked, threshold Y1 for the dispersion and 2 more thresholds Y2 plus Y3 for the selection ratio. A word is considered as a Null word if the dispersion is higher than the threshold Y1 and the selection ratio lies in between Y2 and Y3, and the word is coined as Bonus if the selection ratio exceeds the higher limit Y3, finally the Stigma words are the words whose selection ratio lies on the below the lower limit Y2.

4. Sentence Location

The placement of the sentence determines the significance of the sentence. The assumption is that the significance is high if the sentence is placed at the start or end of the paragraph. The location weight is a summation of 2 weights, 1. Heading weight and 2. Ordinal weight. Similar to the ones above the Heading vocabulary is created which stores the title and the Heading words from hundreds of articles and the words are weighted based on the selection ratio. The heading weight of the sentence is defined as the sum of the heading word weights in the sentence. And the ordinal weight is weighted according to the position of the words in the document. Accumulation of the Heading weight and the Ordinal weight constitute the Location weight of a sentence.

5. Aggregation of the 4 methods

The linear summation of all the 3 weights computed from the above 3 methods for a given sentence forms the *Final Weight*, W.

$$W = p_1K + p_2T + p_3L + p4C$$

Where p_1, p_2, p_3, p_4 are +ve coefficients assigned to the weights K, T and L and the values are determined manually based on the generator performance. The sentences are then organized in descending order on their weights and the top N sentences are taken which form the abstract, where N was taken as 25% of the total sentences in the document.

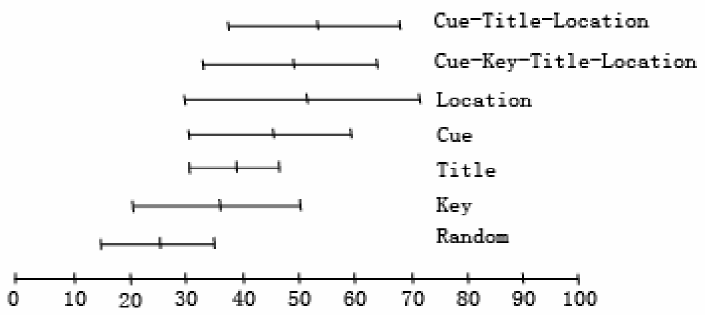


Figure 2 : Comparison of methods

Figure 2 [1] [3], illustrates the statistical results obtained while comparing abstracts generated by the automated summarizers versus the manually generated one. The figure shows the mean % centages of co selected sentences on a set of testing documents. The graph reveals the performance of the Key word method does do well, while the combination Cue – Title – Location yield better performance out of all the methods. Overall the performance was reasonably good but leaving some drawbacks like the determination of the coefficients p_1, p_2, p_3, p_4.

2.2.3 Complex Network Approach

The approach involves the concepts of Network theory to extract sentences for the abstract. The approach involves conversion of large text document into networks and tries to exploit the network metrics in the generation of the summary. Based on vast network metrics, several summarizers were developed; in bulk known as *CN-Summ* [8], and each summarizer employ unique network metric such as node degree, k-cores, shortest path and d-rings.

2.2.3.1 Basics of Graph Theory

Graph: Collection of dots connected or not connected via an edge. A graph is a pictorial representation to illustrate relationships between entities/ dots [9].

Basic Definitions [9]

Edge: 2 nodes are connected via a line known as an Edge.

Vertex: An edge in the graph has 2 endpoints known as the vertex.

Size of a graph: Sum of edges in the graph.

Order of a graph: Sum of nodes in the graph.

Degree of a node: Sum of edges connected to the node.

Complete graph: Nodes in the graph are connected to every other node via an edge.

Diameter of the graph: Shortest distance between 2 nodes which are placed far apart in a given graph.

Scale free network: A Graph in which the ratio of highly connected nodes known as the Hubs to the less densely connected nodes is a constant.

2.2.3.2 History

Before digging down deep into the summarizers let us have a glimpse at some of the traditional techniques employed using network approach.

1. An Approach by Salton et al [8]

Salton proposed an approach where the paragraphs represent a node and the nodes are interconnected via an edge built on a similarly measure computed based on the number of words common in between them. The edges are weighted based on the number of common words in between them and with the aid of the routing algorithm the most prominent paragraphs are extracted to compose the abstract. The approach was evaluated and was found that the algorithm choose around 45% of the paragraphs that were selected by the summarizers generated by humans. The approach had a negative side when it comes to the compression rate, as the paragraphs appear as it is and cannot be split while composing abstract.

2. Mani and Bloedorn approach [8]

The approach represents each word as a node and the nodes are attached by cohesion relations like recurrence, closeness, co-reference and synonymy. The approach expects a topic as input, based on the input, the nodes are selected and with the aid of a spreading activation algorithm the nodes are assigned some weights; later the ones with the highest weights indicate the sentences to be selected for participation in the generation of the summary.

3. Mihalcea's Work [8]

In this approach the sentences are ranked by recommendation algorithms which classify web pages. Below are the recommendation algorithms which rank the sentences.

1. Google's PageRank algorithm
2. HITS

As discussed above the technique remains same as that employed by *Salton et al* wherein the sentences form the nodes and the nodes are connected if they share some common words between them. And the edge weights are rated based on the count of the words matching in both the sentences. It was evaluated that the algorithm HITS performed better than the Google's PageRank algorithm.

2.2.3.3 CN-Summ: A complex Network Approach [8]

This novel technique is similar to the one employed by *Salton et al*, in this approach the node represents a sentence and an edge represents some common trait between them. The trait in this scenario is the common words. Please note only the lemmatized nouns are considered, henceforth limiting the number of edges between the nodes. We treat that if 2 sentences are connected then they provide complementary information about the topic, however the 2 may contain the same information.

The proposed technique comprises of 4 steps, *Step 1*. The pre-processing step: Lemmatization and sentence boundary check. *Step 2*. The resultant text is transformed to network and *adjacency matrix* and *weight matrix* are created. *Adjacency matrix* is an N * N matrix where N is the number of nodes where the value at the position $a_{ij} = a_{ji} = 1$ if their exists an edge between the nodes i and j, in the absence of an edge the value is 0. Similarly the weight matrix illustrates the weights assigned to the edges. These are known by matrix A and W for simplicity. In *Step 3*, we employ one of the network metrics to assign numerical weights and rank the network nodes. *Step 4*. The nodes are sorted in descending order of their ranks and the top n nodes are selected for the generation of the abstract. The value n denotes the compression rate. The technique comprises of 7 network measurements and leads to 14 summarization strategies.

2.2.3.3.1 Degree strategies: CN-Strength and CN-Degree [8]

The strategy tries to capture the most informative nodes based on the network metric 'degree'. The degree of the node denotes the number of edges connected to the node. We obtain the degree of a node from the Adjacency matrix A as.

$$\boldsymbol{ki = \sum_{j=1}^{N} a_{ij}}$$

k_i is the degree of the node i and a_{ij} corresponds to the position in the adjacency matrix A. If we consider the weight matrix in place of the adjacency matrix the formula deduces to

$$\boldsymbol{Si = \sum_{j=1}^{N} w_{ij}}$$

Where s_i denotes the strength of the node i, and w_{ij} corresponds to the position in the weight matrix W. The metrics degree and the strength are the simplest and most commonly used ones to analyse any complex network. The nodes with high degree or high strength are referred to as the *hubs*. The sentences with top n highest k_i are extracted to compose an abstract which form the 1st summarization technique - known as *CN-Degree*. Likewise the sentences corresponding to the top n highest s_i is extracted which form the 2nd summarization technique - known as *CN-Strength*.

2.2.3.3.2 Shortest path strategies: CN-SP, CN-SP^{wi} and CN-SP^{wc} [8]

The approach focusses on the *shortest path* between the nodes. A path is defined as the collection of non-recurring edges from one node to another. And the path length is defined as the number of edges in a path. Unlike the network metric 'degree' discussed earlier the shortest path approach takes into consideration all the nodes in the graph. The shortest distance is computed for all the nodes in the graph and the average distance is computed with respect to each node. With this information in hand the sentence inter-relationship can be determined. We define d_{ij} as the shortest distance or the path between the nodes i and j that can be computed from the adjacency matrix A. We consider the mean shortest path from a node to the rest of the nodes in the graph. The shortest path sp_i is calculated as follows.

$$\boldsymbol{Spi = \frac{\sum_{i \neq j} d_{ij}}{N-1}}$$

N is the total number of nodes in the graph and d_{ij} is N if there do not exist an edge between the nodes i and j and this type of disconnection between the nodes are encountered in *disconnected graphs*. For the weighted edges i.e. weighted graphs, we sum the weights of the edges with constitute the edges of a path. Care should be taken as the shortest path algorithms neglect high weighted edges and consider the lower ones for computing *shortest path*, so some kind of modifications are necessary to rectify this.

One way of rectification is by creating weights which are inversely proportional i.e. for the edges with high weights are to be mapped to another set of values which are low and vice versa. We now create another matrix W^{wc} which is the modified version of W. The elements of the matrix $w_{ij}^{wc} = 0$ if $w_{ij} = 0$ and $w_{ij}^{wc} = w_{max} - w_{ij} + 1$ if $w_{ij} > 0$, now the modified matrix is employed for determing the *shortest path* and the *mean shortest path* sp_i.

Another way of rectification is by determining matrix W^{wi} which is the modified version of W where the elements of the matrix is determined by taking the inverse of the weights of matrix W i.e. $w_{ij}^{wi} = 1/ w_{ij}$ if $w_{ij} > 0$ and $w_{ij}^{wi} = 0$ if $w_{ij} = 0$. We now determine the *shortest path* and the *mean shortest path* sp_i using matrix W^{wi}. With these matrices, 3 unique summarizers are developed i.e. CN-SP, CN-SP^{wc} and CN-SP^{wi}.

2.2.3.3.3 Locality index strategy: CN-LI [8]

The strategy looks for connectivity pattern in the neighbourhood node. We say the neighbors of a node i, is strongly connected to node i, if they share few links with the remaining nodes of the network and are said to have a localized connectivity pattern with node i. Nodes with higher values of locality index are termed as the most prominent nodes and these sentences contribute in the composition of the extract.

The locality index of a given node is computed as follows;

$$Li = \frac{N_i^{int}}{N_i^{int} + N_i^{ext}}$$

Primarily the locality index relates the number of internal connections between the neighbors of the node i, with the number of connections that exists with the remaining nodes in the network.

N_i^{int} = (Number of connections between k_i neighbors of node i) + (k edges from node i to k neighbours)

N_i^{ext} = Number of connection established by k_i neighbours with the remaining nodes in the network.

2.2.3.3.4 d-Rings strategy: CN-Rings^l, CN-Rings^k and CN-Rings^{lk} [8]

d-Ring is a subgraph of a graph G obtained from the morphological operation dilation. Dilation of a graph yields $\delta(g)$ a subgraph that comprises of the nodes of g and the nodes that are connected to g. d-dilation is the process of recursive execution of dilation d times.

$$\delta_d(g) = \underbrace{\delta(\delta(\ldots..(g)\ldots..))}_{\text{d times}}$$

Nodes in $\delta_d(g)$ includes nodes that are present in $\delta_{d-1}(g)$ plus some nodes that are connected to the nodes present in $\delta_{d-1}(g)$. D-Rings are known as *hierarchical level d of subgraph g*. Let us consider g as a graph with a single node, now the first level neighbour of node i comprises of k_i neighbour nodes which form the 1st ring similarly the 2nd ring form the next level consecutive nodes and so on. The number of rings decides the compression rate of the extract. The concept of concentric circles is important to determine the central idea and extract the sentences which complement the main idea. Nodes of higher degree k_i which are important are referred to as the *hubs*. With these 3 summarizers were developed.

1. *CN-Rings^l*, the strategy using the sentence location technique in sentence extraction, the sentences which appear at the first are selected when it becomes hard to include all the sentences from the outermost layer of the hub.

2. *CN-Rings^k*, based on the degree of a node the sentences with highest degree are selected, when it becomes hard to include all the sentences from the outermost layer of the hub.

3. *CN-Rings^{lk}*, it is a combination of above 2, where the nodes above certain threshold values are selected plus the sentences that appear first in the source article, when the outermost layer i.e. the d-layer doesn't fit in the abstract content.

2.2.3.3.5 k-cores strategy [8]

The strategy involves generation of a subgraph $core_k(G)$ of a graph G which comprises of a set of nodes whose sentences are closely related to each other. The nodes which are selected from the graph G are the nodes with degree greater than some threshold value and based on these cores the prime set of nodes are selected and are used in composition of the abstract.

2.2.3.3.6 w-Cuts strategy [8]

The approach is similar to the one explain before (k-Cores) the difference is that instead of degree of a node the edge weights are considered in the subgraph formation. The ones with higher edge weights are selected for abstract formation.

2.2.3.3.7 Voting strategy [8]

This approach is a combination of strategies that are listed above; the approach involves collating some of the above techniques and brings out the best combination in them.

2.3 Background Summary

The chapter covered few traditional techniques in the field of automatic summarization, which started from determining the most prominent words in the document and the significance of the sentence based on the occurrence of the words in each sentence of the article. Another approach to significance was by determining the significance factor of a sentence i.e. based on the relative positions of the significant words in a sentence, and the most significant sentences contribute in the composition of the abstract. The approach of Luhn was extended by H.P Edmundson who introduced 3 features, 1. Key word extraction i.e. based on word occurrence, 2. Title extraction which is based on the assumption that title summarizes the content under it, 3. Sentence placement which is based on assumption that the sentence following the title or the heading and the one at the end of the paragraph contain vital information which contribute in abstract formation. At last all the above 3 features were aggregated to form one. Later Kupiec came up with some more features like the sentence length, key phrase technique and Case consideration. The approach by Kupiec was followed up with the introduction of the concept of machine learning, which employs a Naive Bayes classifier to determine the probability of a sentence for its inclusion in Abstract formation. Finally the Complex Network approach which involved the concept of network theory in determining the vital words in the document based on network metrics.

CHAPTER 3

COMMUNITY DETECTION ALGORITHM

3.1 Introduction

Communities in a network are referred to as the group of nodes that are closely interconnected, while scarcely connected between the nodes of other groups [4] [5].

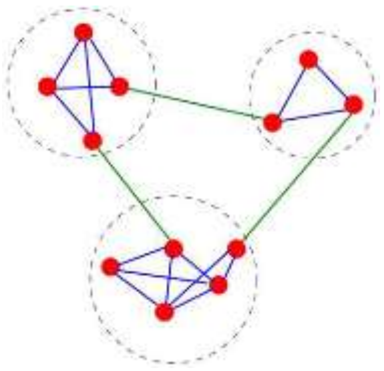


Figure 3 [4] [5]: Communities in a network

The study of complex networks was virtually impossible and now with the concept of network communities, the complex network can be broken down into several communities i.e. *Divide and conquer approach* and each community can be analysed individually which makes analyses a lot easier. In case of communities in text network, it is assumed that sentences under each community try to convey some vital information about the document. And selection of appropriate communities from the network aid in extract composition [5].

Causes of communities in Networks [4]

- Homophily: Nodes sharing similar attributes are more likely to be linked. This is also referred to as the *assortative mixing*.
- Nodes are more likely to be connected if both of them have a neighbour common between them.

3.2 Types of Community detection algorithm [5]

Detecting communities in any network is a challenging task and there is numerous detection algorithms developed and employed despite several challenges like variable community sizes, unknown number of communities in a network.

3.2.1 Graph Partitioning

In graph partitioning [4] we have many algorithms in which Minimum-cut [6] method is one of the most popular one. The approach divides the network into predefined number of sub networks which are usually of the same size/ density. In this method it is mandatory to predefine the number of sub networks we need. The partition is made in such a way that the number of edges between the groups is kept minimal and the count of edges between communities is referred to as the *cut size*.

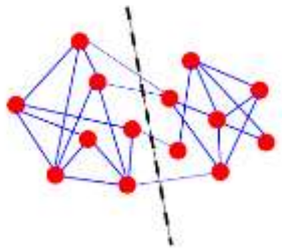


Figure 4: Graph Partitioning

Figure 4 [4] , illustrates a network with 14 nodes and employing Minimum cut method we divide the network into 2 clusters more or less of appropriate size bearing in mind minimal number of edges between the groups. The algorithm proved to do well only in applications which are intended to use this technique but failed to produce the same results when applied on general networks. The drawback of this technique is that the technique detects only a predefined number of sub networks and finds communities regardless of their existence in a network.

3.2.2 Hierarchical clustering [4]

In general with slight information about the network it is very uncommon to predict the number of clusters or the communities in the network beforehand and in these scenarios graph partitioning technique is proved to be of no use. And in general some graphs may have hierarchical structure i.e. may have nodes paired at different levels, forming clusters within a bigger cluster. For instance we encounter these kinds of network structures in social networks.

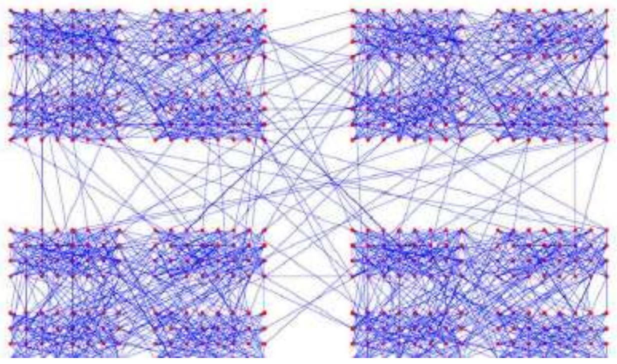


Figure 5 [4]: Communities nested inside bigger ones

The technique introduces a *similarity measure* based on which a network is divided into communities. The *similarity measure* computes some sort of similarity between pairs of nodes based on which clustering of nodes is done. We have several similarity measures for measuring similarity and below are some.

1. Hamming distance between the rows of adjacency matrix.
2. Cosine similarity.
3. Jaccard index and so on.

The pairwise similarities between the nodes of the network are computed by employing one of the above similarity measures and once the hierarchical structure is constructed employing one of the two classification method communities can be detected.

There are 2 ways of classification.

1. Agglomerative algorithms [4]: It is a *bottom up* approach i.e. ideally with the hierarchical structure of the network in hand we start merging the nodes with high similarity measure from bottom iteratively.

2. Divisive algorithms [4]: It is a *top down* approach and is the opposite of Agglomerative algorithms where we split the network iteratively by getting rid of the edges with low similarity measure ultimately leaving behind the communities.

Usually in Hierarchical clustering a threshold is set to limit the size of cluster we desire for and below is an example which illustrates the hierarchical structure of a network and the red dashed lines define the threshold for the community.

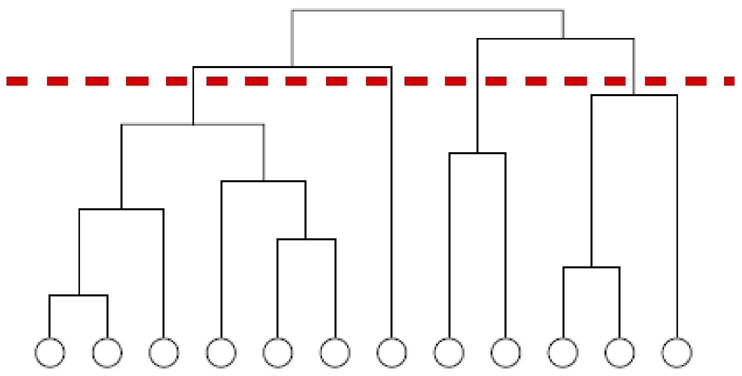


Figure 6 [4]: Hierarchical Representation of Network classified employing Hierarchical clustering

3.2.3 Girvan-Newman algorithm [4]

The algorithm was proposed by Girvan and Newman and is of historical importance as the technique marked an era in the field of community detection. The algorithm searches for edges which are of importance. The task of determining the edge importance is through the network metric measure called *betweenness*. The edge *betweenness* of an edge is defined as the number of shortest paths between all the vertexes in the network that run via this edge. The edges which connect communities have higher values of betweenness as majority of nodes from one group are connected to other groups via this edge. The technique involves determining edges that link communities and get rid of them so that ultimately only the communities are left behind. The betweenness value for each edge in a network is computed and the edges with high importance are determined and eliminated will we obtain desired number of communities.

Algorithm [5]

Step 1: Compute betweenness for each edge in the network.
Step 2: Edge with highest betweenness value is removed.
Step 3: Compute betweenness for the new network again.
Step 4: Jump to Step 2.

The results obtained from Girvan-Newman algorithm are reasonably good and the technique is not recommended for huge networks, as the complexity of the algorithm is $O(m^2n)$ considering *m* as the number of edges and *n* to be the total node count in the network.

3.2.4 Modularity maximization [4]

Modularity maximization is the most popular technique and is based on the functionality, *Modularity* [12]. The *Modularity* function Q was introduced by Girvan and Newman to define the terminating criteria for the algorithm Girvan and Newman. The measure *Modularity* defines the quality of partition in a network into communities. The algorithm firstly computes the modularity for all possible partitions i.e. communities, in a network and the partition with highest modularity is chosen as communities. The modularity Q is given by

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

Where,

$\frac{1}{2m} \sum_{ij} \left(A_{ij} \delta(c_i, c_j) \right)$ Defines the number of edges that are connected internally (intracommunity).

$\frac{k_i k_j}{2m}$ Is expected value of A_{ij} from the configuration model.

Employing modularity functionality Greedy technique was presented by Girvan and Newman. The Greedy technique is an agglomerative hierarchical clustering technique which iteratively collates nodes to form communities such that Q increases after merging. The technique kick starts with n clusters, cluster of 1 node and no edges exits initially. Now merging is done by connecting the nodes via an edge and we compute the increase in modularity. Likewise the modularity is computed for different possible combinations of nodes and the increase in modularity is computed. The combination which yields higher modularity is considered and the nodes are paired. This procedure is repeated till desired number of communities and size of communities are obtained.

Algorithm [5]

Step 1: Start with *n* communities and each community comprise of 1 node.
Step 2: Connect each pair of communities and compute increase in modularity ΔQ.
Step 3: Communities with maximum ΔQ are merged.
Step 4: Jump to Step 2 till expected number of communities are obtained.

Since the algorithm is designed to try out all possible combinations the time taken is exponential and henceforth is not recommended for networks with massive size/ density.

CliqueMod algorithm [4]: The problem with Greedy algorithm is that the initial steps are random and communities were singletons i.e. communities of single node. In order to overcome this problem instead of building the communities from scratch i.e. from 1 node we now identify the cliques in the network and then start building on them. And from experiments it is proved that the results obtained by employing CliqueMod algorithm are better than Greedy algorithm.

Algorithm [5]

Step 1: Determine cliques in the network which form the initial communities.
Step 2: Connect each pair with the existing communities and compute increase in modularity ΔQ.
Step 3: Communities with maximum ΔQ are merged.
Step 4: Jump to Step 2 till expected number of communities are obtained.

3.2.5 Clique based methods [5]

A Clique is a complete graph. A complete graph is a graph in which every node in the network is connected to every other node via an edge. Many detection algorithms are based on identifying these cliques in the network. The approach is as follows; firstly we extract cliques that are not a part of any clique and whose sizes are above some pre-defined threshold, then the approach employs several other techniques in overlapping these cliques appropriately which later form communities in networks.

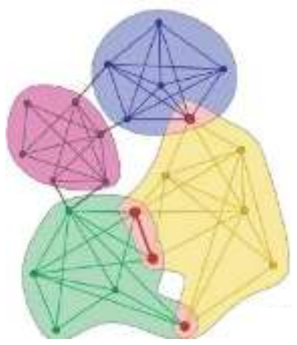


Figure 7 [5]: 4 clique communities in a network

3.2.6 Surprise maximization [7]

The approach is a recent one and engages a measure *Surprise*, S in detecting communities in networks. Like the Modularity maximization the measure S evaluates the quality of partition using "*cumulative hypergeometric distribution*". The difference between S and Q is that in Q the quality was measured based on the density of the edges while S takes into consideration both the density of the edges plus the density of nodes in a given network. And performance wise it was proved that S outperforms modularity measure Q.

3.2.7 InfoMap [14]

InfoMap is the most efficient community detection algorithm which was developed by *Rosvall* and *Bergstorm*. In this approach the network is partitioned via 2 stage nomenclature which is based on the concepts of Huffman coding.

Stage 1: Differentiate network communities

Partition is done based on the inter-community links, lower the inter-community links the more probable to form communities.

Stage 2: Differentiate nodes in community.

With the aid of a random walker we now define the quality of communities formed.

Further the technique is optimized through simulated annealing.

CHAPTER 4

TRADITIONAL AUTOMATED TEXT SUMMARIZERS

4.1 Introduction

The chapter illustrates a detail literature on some of the traditional Automated Text summarizers developed with a purpose to compare the performance of our summarizer with them.

CHAPTER OUTLINE

- *List of Text summarizers*: A brief introduction on the list of traditional summarizers developed.
- *Design Framework*: Steps involved in designing a summarizer.
- *Lexical Processing*: Text is subjected to a pre-processing step before it is fed to the summarizers.
- *Traditional Automated Text Summarizers*: A detailed literature on 4 Text summarizers developed employing various network metrics.
- *Conclusion.*

4.2 List of Text Summarizers

We have developed 4 traditional text summarizer which employ different network metrics to determine key sentences from a document and the metrics employed are as follows:

- Degree
- Betweenness
- PageRank
- Clustering Coefficient

4.3 Design Framework

The design of the summarizer is as follows:

Step 1: Text document is subjected to *Lexical Processing* stage.

Step 2: Convert *outcome of the previous stage* into a *complex network representation*.

Step 3: Analyse *complex networks* using different network metrics to *identify keywords* in the document.

Step 4: Compute *sentence significance* which is the total count of keywords in the sentence.

Step 5: *Sort* the sentences in descending order of their significance.

Step 6: *Collate* top N sentences which are rated high on significance which form the document Abstract, where N depends on compression ratio.

4.3.1 Lexical Processing

The source text document is initially subjected to a set of pre-processing techniques before converted to network representation. To parse through the document content we developed a Parser system which aid in text parsing and necessary lexical processing.

The Pre-Processing techniques [1] are as follows:

- Remove Environments
- Extract Main Body
- Delete Equations
- Eliminate Non-Alpha numerals
- Tokenization
- Stop Words
- Porter Stemming

A detailed explanation of each of the pre-processing technique is discussed in the next Chapter. The pre-processing techniques remains same for all the traditional as well as our summarizer but the only thing that changes is the way we check for keywords from the document.

4.3.2 Traditional Automated Text Summarizers

We have selected 4 text summarizers which employ both basic as well as complex network metrics in Abstract generation.

4.3.2.1 Automated Text Summarizer - Degree

The prime concept is to extract important sentences from the document which would form an abstract when combined together. Below is the way the sentence is checked for importance and Abstract generation.

1. Degree as metric is the simplest text summarizer and determination of keywords in the document is based on the occurrence of the word in the entire document. The given source document after subjecting to lexical processing is left with a set of essential significant words by filtering out the not so important content in the document.

2. The residual text known as "essential significant words" are now represented in the form of a network. Wherein each node form a word and the neighbouring words are connected through an edge. The sentence significance is determined based on the high frequency words in the sentence and these words are expected to occur in several places in a document. The frequencies of all the words in the document are computed i.e. node degree and the sentence significance is calculated based on the occurrence of these keywords in each sentence.

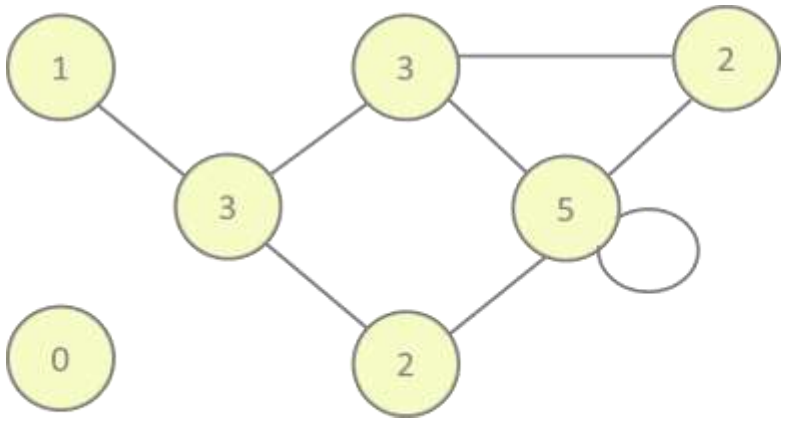


Figure 8: Simple Network: Nodes with degree as labels

Above is a simplest graph with 7 nodes and 8 edges connecting the nodes and each node in the graph is labelled based on the number of links connected to the node i.e. node degree.

Below is an example [10] which illustrates the way we convert text into a graph and determine keywords based on the concepts of network metric - Degree.

Original

"What's that? asked Sally.
Pay my bill for last week, due this morning.
Sally got up quickly, and flitting down the
table, put her arm round her friend's
shoulder and whispered in her ear."

Without stopwords

"asked Sally
pay bill last week morning
Sally got quickly flitting
table put arm friend shoulder
whispered ear"

After Stemming

"ask Sally
pay bill last week
morning Sally get
quickly flit
table put arm
friend shoulder whisper
ear"

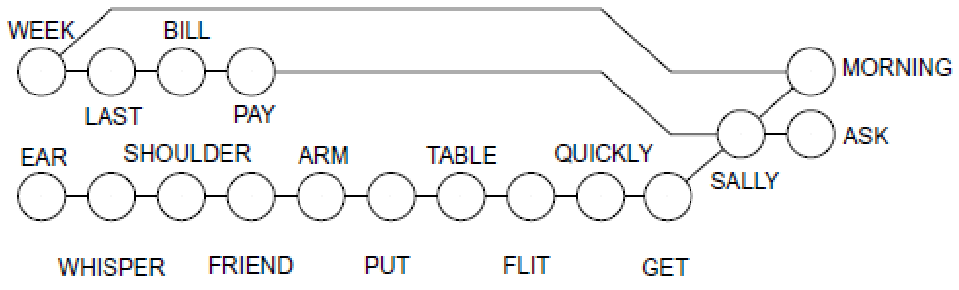


Figure 9 [10]: Network Representations of Text

Words	Network metric – Degree
Sally	4
Flit	2
Put	2
Pay	2
Last	2
Bill	2
Whisper	2
Table	2
Shoulder	2
Morning	2
Arm	2
Quickly	2
Got	2
Friend	2
Week	2
Ear	1
Ask	1

Table 1: Words versus Network metric-Degree

Table above depicts Words versus Network metric-Degree sorted in descending order of their degree and we name the top 50% of the words as *keywords* and compute sentence significance based on the occurrence of these words in the sentence and the collation of the Top N sentences form the document Abstract, where N depends on the compression ratio.

4.3.2.2 Automated Text Summarizer - Betweenness

1. Betweenness is another simple network metric which gives a measure of the number of shortest paths in a network pass through the given node while traversing from every node to every other node in the network. Nodes with high degree of betweenness act as bridges/ an intermediary and aid in connecting nodes in a network.

2. We determine keywords based on the assumption that words with higher values of node betweenness are vital. The given text document after subjecting to lexical processing (discussed in previous traditional summarizer) is then converted to network representation to determine network metric-Betweenness of each of the node in the network.

3. Betweenness of each of the nodes in the network are computed and are sorted in descending order of their betweenness and the top 50% of the words are named *keywords*.

4. Sentence significance is computed based on the existence of keywords in the sentence and the top N significant sentences are collated which forms the document abstract.

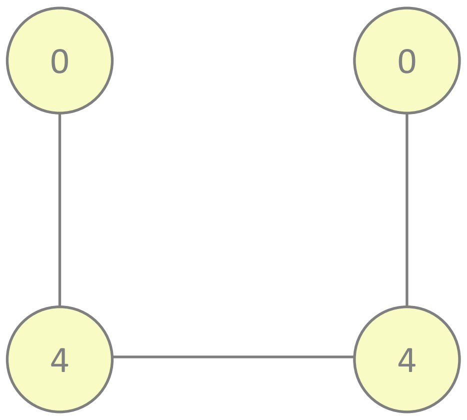
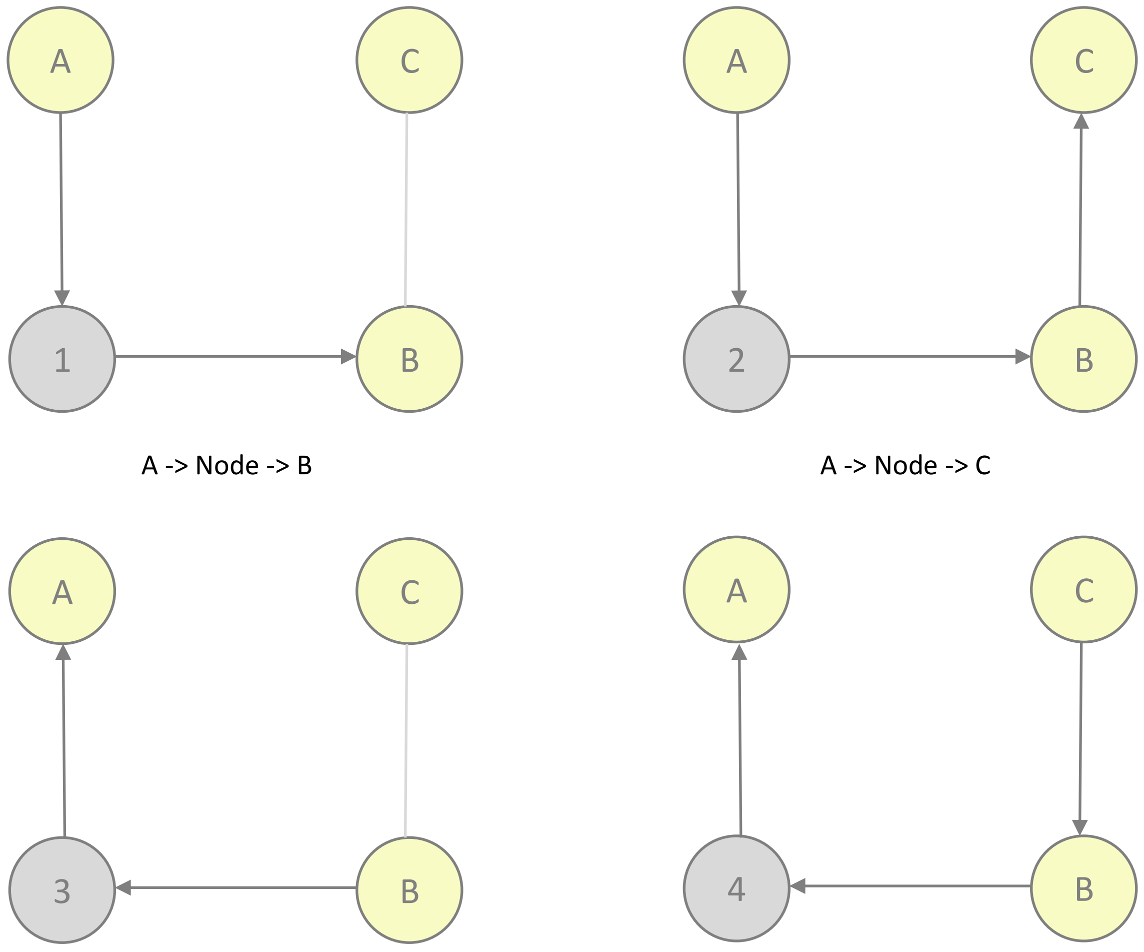


Figure 10: Simple Network: Nodes with betweenness as labels

Betweenness computation of the marked node in the network is as follows:



A -> Node -> B

A -> Node -> C

B -> Node -> A

C -> Node -> A

Words	Network metric – Betweenness
Sally	146
Flit	144
Quickly	144
Table	140
Got	140
put	132
arm	120
round	104
friend	84
Pay	40
Morning	40
whisper	32
Bill	16
Week	16
Last	4
Ask	0
ear	0

Table 2: Words versus Network metric-Betweenness

Table above illustrates Words versus Network metric-Betweenness sorted in descending order for the network shown in figure 10. The top 50% of the words are considered as the *keywords*, and the most significant sentences are extracted which form the document abstract.

4.3.2.3 Automated Text Summarizer - PageRank

The sentence significance is computed as follows:

1. Google's famous link analysis algorithm '*PageRank*' [11] is employed in *keyword* determination. Higher the PageRank value of a node the more probable the word being a keyword. PageRank algorithm is used in Google's search engine which assigns numerical weights to each web page on the internet on the basis of the number of hyperlinks linked with them. Similarly in our approach we weight each node in the network based on the number of links connected to it.

2. The Text network is subjected to PageRank algorithm and we assume the top 50% of the words with high PageRank values to be the document keywords and the document abstract would be the aggregation of all the significant sentences weighted based on the number of keywords present in the sentence.

Simplified Algorithm: *PageRank* [11]

1. Iterative Algorithm: The weighting of each node is computed repeatedly as and when a new node is connected to the network.

2. Initially all the nodes in the network are weighted equal and later based on the number of links connected to each of them the weights would vary.

3. Let us assume 4 nodes, P, Q, R and S in the network, and we kick start with an estimated equal PageRank value of 0.25. Please note the PageRank of each node depends on the PageRank of the node connected to it and the computation is given as below;

$$PR(P) = \frac{PR(Q)}{L(Q)} + \frac{PR(R)}{L(R)} + \frac{PR(S)}{L(S)}$$

Where PR - denotes PageRank value of the node and L - denotes the degree of the node in case of Undirected graph.

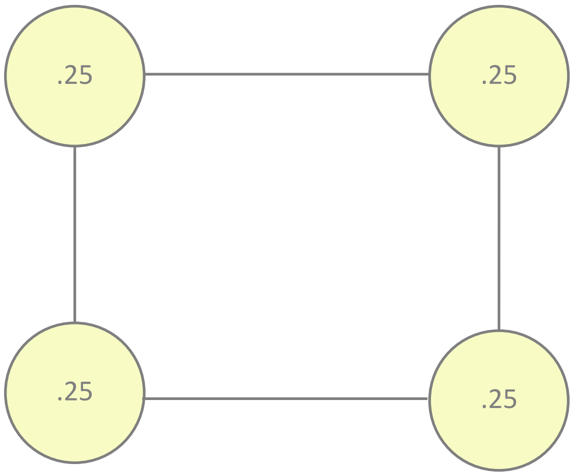


Figure 11: PageRank: Undirected graph

4. In case of directed graphs the way we compute PageRank varies a bit where L - denotes the number of outbound links.

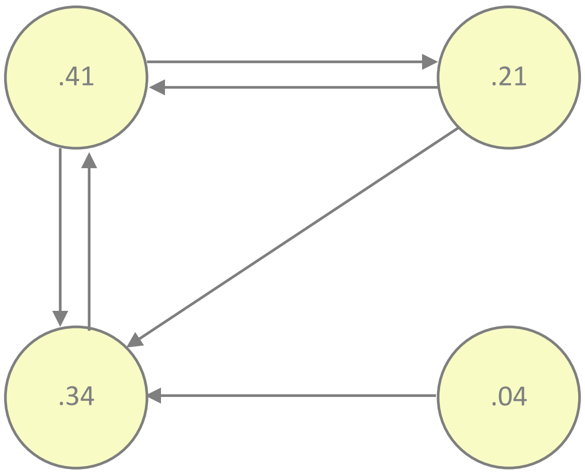


Figure 12: PageRank: Directed Graph

Words	Network metric - PageRank
Sally	0.10047
Whisper	0.06434
Shoulder	0.06043
Friend	0.05823
Round	0.05698
Arm	0.05624
Put	0.05574
Table	0.05530
Flit	0.05478
Quickly	0.05397
Last	0.05377
Week	0.05345
Bill	0.05345
Got	0.05262
Morning	0.05240
Pay	0.05240
Ear	0.03567
Ask	0.02968

Table 3: Words versus PageRank

Table above shows Words versus Network metric-PageRank sorted in descending order for the network shown in figure 10.

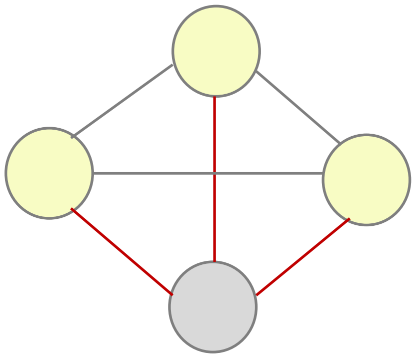
4.3.2.4 Automated Text Summarizer - Clustering Coefficient

Keywords are determined with the aid of another network metric, *Clustering coefficient* and the steps are as follows.

1. Clustering coefficient [9] is a measure of how close the nodes in a network are to form a cluster/ complete graph.

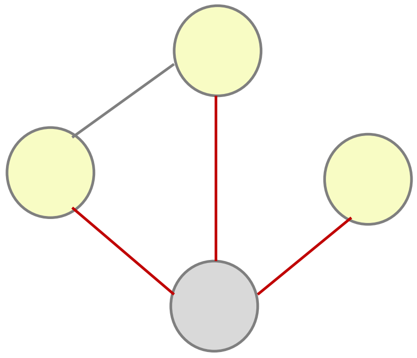
$$Clustering\ Coefficient\ = \frac{Total\ edge\ count\ between\ its\ neighbours}{Number\ of\ edges\ required\ to\ form\ a\ complete\ graph}$$

Clustering coefficient in undirected graph

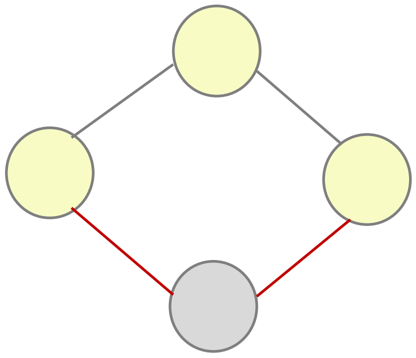


$$Clustering\ Coefficient = \frac{(2 * |edge_{jk}|)}{(k_i * (k_i - 1))}$$

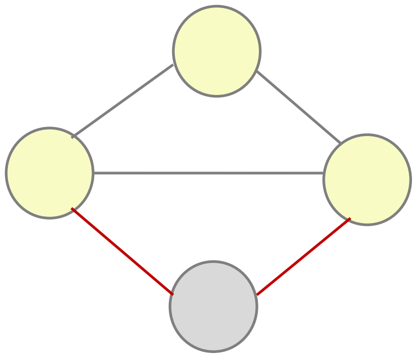
$$= \frac{(2*3)}{(3*2)} = 1$$



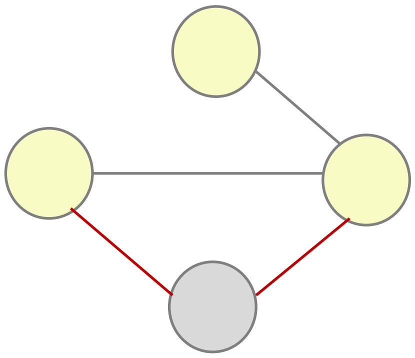
$$Clustering\ Coefficient = \frac{(2*1)}{(3*2)} = \frac{1}{3}$$



$$Clustering\ Coefficient = \frac{(2*0)}{(2*1)} = \frac{0}{2} = 0$$



$$Clustering\ Coefficient = \frac{(2*1)}{(2*1)} = \frac{2}{2} = 1$$



$$Clustering\ Coefficient = \frac{(2*1)}{(2*1)} = \frac{2}{2} = 1$$

2. For every node in the network clustering coefficient is computed and the nodes with higher measure as termed as *keywords* which aid in extraction of vital sentences from the document.

4.4 Conclusion

A comprehensive and principle working of each of the traditional text summarizers was discussed and the chapter predominantly focussed on various network metrics via which keywords were determined which aid in extraction of vital sentences from the core document.

CHAPTER 5

AUTOMATED TEXT SUMMARIZER - COMMUNITY DETECTION ALGORITHM

5.1 Introduction

Automated Text Summarization has always been a field of interest and research. In our approach we developed a novel text summarizer which collates the concepts of *Text summarization* which involved detailed research on all the traditional techniques in the field of text summarization, *Graph Theory* which aided in better analysis of text and *Community Detection Algorithm*, to detect communities in networks.

CHAPTER OUTLINE

- *Design Framework*: Steps involved in designing novel summarizer.
- *Lexical Processing*: Text is subjected to Pre-processing step.
- *Network representation of Text*: Text converted to Graph and analysed using network metrics.
- *Our Automated Text Summarizer*: Text summarizer employing community detection algorithm.
- *Conclusion.*

5.2 Design Framework

Our approach also remains same as many existing traditional ones i.e. Extraction based approach. In this approach, the abstract generated is mere extraction of vital sentences from the core document. As in many other traditional summarizers the framework remains same as explained in the previous chapter and the steps are as follows:

Step 1: Text document is subjected to *Lexical Processing* stage.

Step 2: Convert *outcome of the previous stage* into a *complex network representation*.

Step 3: Analyse *complex networks* employing community detection algorithm to *identify keywords* in the document.

Step 4: Compute *sentence significance* which is the total count of keywords in the sentence.

Step 5: *Sort* the sentences in descending order of their significance.

Step 6: *Collate* top N sentences rated high on significance which form the document Abstract, where N depends on compression ratio.

5.2.1 Lexical Processing

To parse through the document content we have developed a Parser system in java which aid in getting rid of not so important content from the core document. The Parser system developed is capable of parsing through only .txt files and we focus only on summarizing content of .txt files as this is the simplest form of text storage. This is the initial step in any automated text summarizer and detailed information on the Pre-processing techniques is as follows:

Lexical Processing Steps

Manual Pre-Processing

- Remove Environments
- Extract Main Body
- Delete Equations

Automated Pre-Processing

- Tokenization
- Eliminate Non-Alpha numerals
- Stop Words
- Porter Stemming

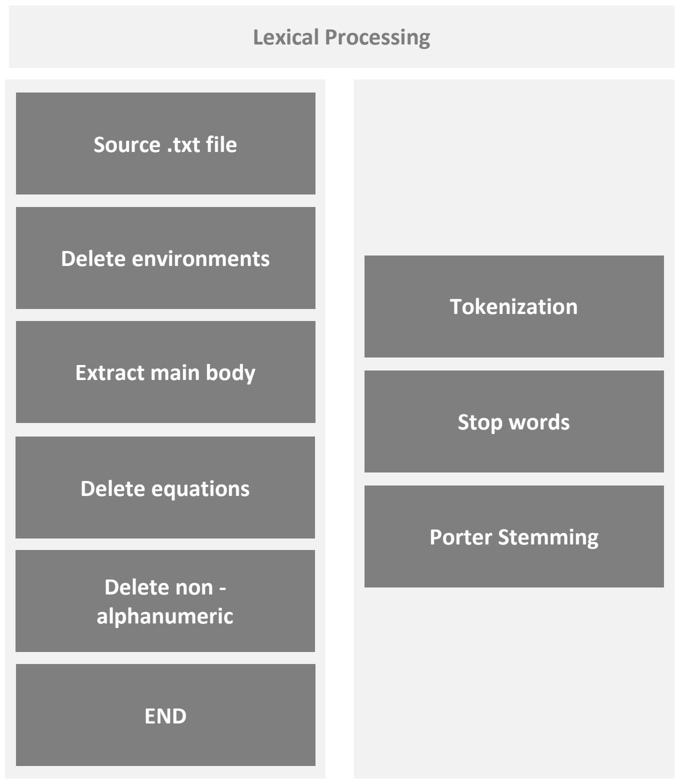


Figure 13 [1]: Lexical Processing

1. Source .txt file

To parse through the contents of the source document a Parser system is developed which extracts only the necessary significant words by stripping other unnecessary content. The parser system is developed in java language and is capable of processing only normal text file/ flat file i.e. .txt file. We have selected .txt file format as it is the simplest form of text container that doesn't support images, pictures or any sort of graphics in them.

2. Remove Environments

A given source text document may contain many not so important contents that are not necessary in abstract composition. The not so important contents would be the tables, images, graphs etc. And these are deleted manually [1].

3. Extract Main Body

We extract core content of the document by removing certain portions of the document such as the Abstract, Conclusion and Bibliography which doesn't contribute in Abstract generation. From the main body extracted sentences are taken and further subjected to a set of lexical processing [1].

4. Delete Equations

Many of the scientific papers include lot of equations which doesn't contribute in Abstract formation and can get rid of it in the initial stage itself. Since our focus is primarily on text we can eliminate all the equations encountered in the document.

5. Tokenization

The parser developed by us aid in tokenization of the source document content. The key function of the Parser system is to fetch document sentences and extract significant content from the document. Firstly tokens are extracted from the sentences and are checked for importance, only the words which are found useful are retained with the rest filtered out.

6. Stop Words

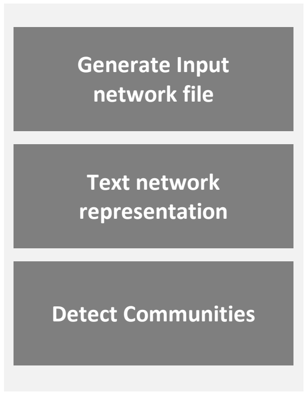
A vocabulary of stop words are maintained which constitute the not so important words which do not attribute to fall under the category of *keyword* list. These words could be the prepositions, conjunction in English language and we also get rid of the words whose size is less than 3 characters which are considered not important.

7. Porter Stemmer

In normal English Language, words can appear in more than one form with their meaning remaining same. So in order to deduce these words to a single word we go for a Stemming algorithm which aid in vocabulary reduction. One drawback of stemming would be we cannot guarantee words to be deduced correctly, for example, word - coming can be deduced to 'com' rather than 'come' but however this doesn't disturb the outcome of the experiment as the transformation of all the words with same meaning deduce to a single unique word [1]. *NOTE: Existing porter stemmer code* [15] *is used*.

5.2.2 Network Representation of Text

The pre-processed output from the lexical processing stage is represented in the form of a network to analyse and determine *keywords*.



Generate Input Network File

After the lexical processing stage the outcome is now ready to be converted into network wherein we now denote each word as a node and the adjacent words are connected via an edge. In the process to represent text as network we generate a network file which provides edge details for graph generation.

The format is as follows:

Step 1: We associate a unique number sequence for each word.

For instance: We associate unique number sequence for each unique word for below text as

**"Ask Sally
pay bill last week
morning Sally get
quickly flit
table put arm
friend shoulder whisper
ear"**

Number sequence	Word
1	Ask
2	Sally
3	Pay
4	Bill
5	Last
6	Week
7	Morning
8	Get

9	Quickly
10	Flit
11	Table
12	Put
13	Arm
14	Friend
15	Shoulder
16	Whisper
17	Ear

Table 4: Network File - I

Step 2: Now the adjacent words are represented as edges in the network and in the network file it is denoted as below:

"Ask Sally
pay bill last week
morning Sally get
quickly flit
table put arm
friend shoulder whisper
ear"

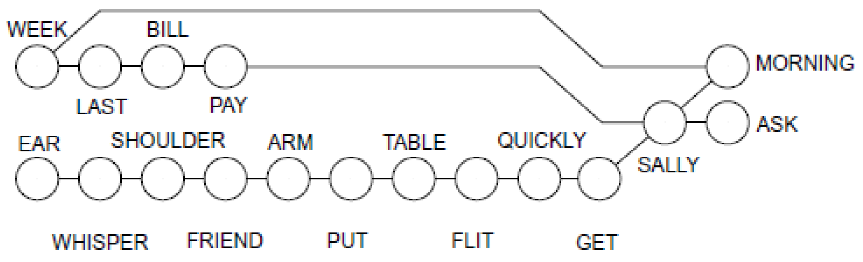
Edge	Adjacent Words
1 2	Ask Sally
2 3	Sally pay
3 4	Pay bill
4 5	Bill last
5 6	Last week
6 7	Week morning
7 2	Morning Sally
2 8	Sally get
8 9	get quickly
9 10	quickly flit
10 11	flit Table
11 12	Table put
12 13	put Arm
13 14	Arm friend
14 15	friend shoulder
15 16	shoulder whisper
16 17	whisper ear

Table 5: Network File - II

In the above table, the column *edge* is enough to represent the given text as network and we refer to this content in a file as the *network file*.

Text Network Representation

The network file generated when converted to network will appear as below. Similarly when a huge document is converted to network a complex network is produced and the network is analysed further via community detection algorithm.



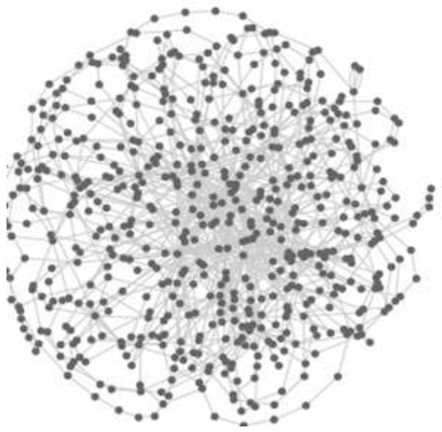


Figure 14 [10]: Huge Complex Text Network

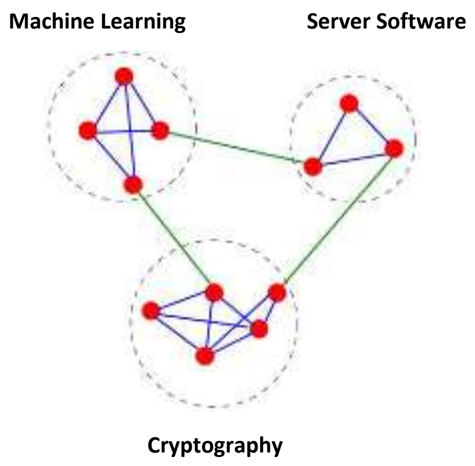
Detect Communities

We detect communities in complex text network by using an existing efficient Community Detection Algorithm, *InfoMap*. The approach we are employing is a Divide and Conquer approach wherein we analyse complex network by dividing the network into smaller networks/communities and a unique thing about the communities in the text is that text under each community tries to convey unique information about the document and hence integration of all the communities sum up to the document information.

Note: We have used an efficient existing InfoMap code, and the way we use is as a black box usage wherein we feed text network to InfoMap algorithm and the outcome of that would be the list of communities.

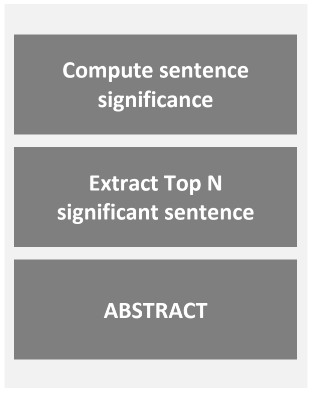
In our approach, the document abstract is formed by collating the most significant sentences from each community and thus assuring the abstract covers all the major topics in the document.

For instance: If a document speaks about different topics such as Machine Learning, Cryptography and Server software, then 3 communities are formed.



5.2.3 Abstract Generation

An abstract is precise and concise information about a document and hence care should be taken while extracting sentences for the abstract from the document.


Compute sentence significance
Extract Top N significant sentence
ABSTRACT

Sentence Significance and Extraction

The *significant words* obtained after the lexical processing stage are converted into network and are subjected to Community detection algorithm to detect communities of words. We assume the words under each community to be the *keywords* and these keywords are related to some unique information about the document as discussed in the figure above. We now compute the Sentence significance based on the occurrence of these keywords in the sentence and the top significant sentences are accountable to form the Abstract.

Abstract Generation

Vital sentences from each community are selected based on compression ratio and the collation of all these sentences from different communities form the document Abstract.

5.3 Conclusion

The chapter covered detailed working of our summarizer which included a range of pre-processing techniques and the way we check for keywords and extract vital sentences from the main document. The version of summarizer we have developed is a basic version and in future the summarizer can be made complex and advanced with many features embedded in them.

CHAPTER 6

EXPERIMENTS AND RESULTS

6.1 Introduction

The chapter illustrates a detailed literature on the performance evaluation of our summarizer versus some of the traditional ones.

CHAPTER OUTLINE

- *Summarization Evaluation:* A brief introduction on the general properties which aid in summary evaluation.
- *Types of Summary Evaluation:* Intrinsic and Extrinsic summary evaluation.
- *Intrinsic Summary Evaluation:* Similarity measure, Precision & Recall and F-Measure.
- *Extrinsic Summary Evaluation:* Panel of markers who evaluate the quality of the summary generated.
- *Conclusion*

6.2 Summarization Evaluation [12]

Evaluating *automated text summarizers* and the *summary* generated out of them is a tedious process. In general terms there are at least 2 general properties which aid in summarizer and summary evaluation.

- Compression Ratio CR: Measure of how short the summary is over the original text document.

$$CR = \frac{\text{Summary size}}{\text{Actual text size}}$$

- Retention Ratio RR: Measure of information retained in the summary generated.

$$RR = \frac{\text{Summary information}}{\text{Actual text information}}$$

6.2.1 Types of Summary Evaluation [12]

The evaluation is broadly classified into 2 methods,

1. Intrinsic evaluation
2. Extrinsic evaluation

6.2.2 Intrinsic Summary Evaluation [12]

The evaluation technique predominantly involves comparison of the summarizer with a set of standards, taken as a reference summary, which is later compared with the auto generated one for similarity. The Intrinsic technique employed in our performance evaluation is *Precision & Recall* and *F - Measure*.

Evaluation Approach: Our approach would be to compare the original abstract with the auto generated abstract generated by the summarizers. Firstly we compute the similarity of the sentences between the abstracts and later the quality of the summary is evaluated through *Precision and Recall* method.

6.2.2.1 Similarity Measure (Cosine similarity) [1]

The similarity of 2 vectors is measured using cosine of the angle between them.

$$\text{Cosine similarity } \cos\theta = \begin{cases} 0 & \text{Vectors are perpendicular, indicating } \textit{independence} \\ 1 & \text{Vectors are parallel, indicates the vectors are } \textit{identical} \\ -1 & \text{Vectors are opposite to each other, indicates } \textit{contradictory} \end{cases}$$

In case of Text similarity, the measure of similarity lies between 0 - 1 as the term frequencies i.e. TF - IDF vectors (refer next page) cannot be –ve.

Euclidean Dot Product

We determine the cosine of the angle between the vectors by employing *Euclidean dot product* formula. Given 2 vectors X and Y, and the angle between the vectors be θ.

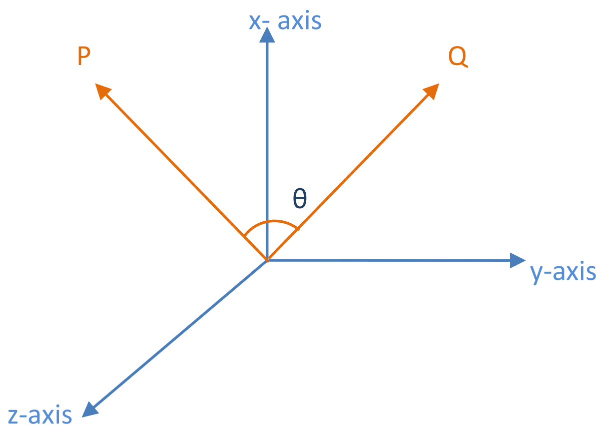


Figure 15 [1]: Vector representation

$$P.Q = |P|\ |Q| \cos\theta$$

$$\text{Similarity} = \cos\theta = \frac{P\,.\,Q}{|P|\ |Q|}$$

$$\text{Cos } \theta = \frac{\sum_{j=1}^{n} P_j * Q}{\sqrt{\sum_{j=1}^{n}(P_j)^2}\sqrt{\sum_{j=1}^{n}(Q_j)^2}}$$

6.2.2.2 Text Similarity [14]

The measure of similarity between texts is referred to as the Text Similarity. And in our experiment the similarity is measured by computing the cosine of the angle between the vectors wherein each vector denotes a "Sentence". The technique involves computing <Word: frequency> pair in the sentence and calculates the TF-IDF scores for each sentence/ vector. Below example illustrates the way we check for text similarity.

For example:

Let us assume 2 sentences a and b, let X, Y, Z, M and N represent words in the sentence.

Sentence a:

X Y Z X X Y Z M M N X Y M X Y Z Y X

Sentence b:

X Y Z X X Y Z M X Y Z Y X

TF - IDF Vector a:

X: 6	Y: 5	Z: 3	M: 3	N: 1

TF - IDF Vector b:

X: 5	Y: 4	Z: 3	M: 1	N: 0

Similarity Measure between sentences a and b:

$$\frac{[(6*5\,)+(5*4)+(3*3)+(3*1)+(1*0)]}{\sqrt{6^2+5^2+3^2+3^2+1^2}\ \sqrt{5^2+4^2+3^2+1^2+0^2}}$$

Similarity Measure = 0.97

Note:

- A Similarity measure of value 1 implies the sentences are similar.
- 0 implies the sentences are completely dissimilar.
- 0 - 1 implies partial similarity.

6.2.2.3 Abstract Similarity [1]

From the previous cosine similarity measure we conclude higher the measure; the more similar are the sentences. We now compute the similarity between the abstracts by selecting each sentence from the auto generated abstract and find a matching sentence in the original abstract through cosine similarity.

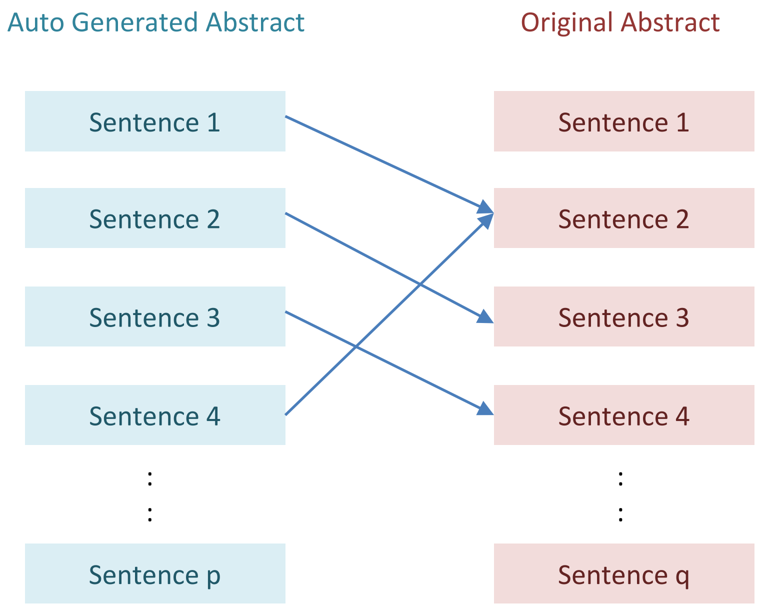


Figure 16 [1]: 1 - N Sentence Mapping

Figure 9, illustrates 1- N mapping between the sentences of the 2 Abstracts. The sentence together with its matching sentence forms a "Sentence pair" and it is the average cosine values of sentence pairs determine the similarity between the Abstracts.

6.2.2.4 Precision & Recall and F-Measure [1]

The most commonly used technique in the context of Information retrieval is *Precision and Recall*. And below are the equations employed to evaluate the quality of the summary generated by the summarizers.

$$\text{Precision} = \frac{|\ \{\text{Original Abstract sentence}\} \cap \{\text{Auto-Generated Abstract sentence}\}\ |}{\text{Number of sentences in Auto-Generated abstract}}$$

$$\text{Recall} = \frac{|\ \{\text{Original Abstract sentence}\} \cap \{\text{Auto-Generated Abstract sentence}\}\ |}{\text{Number of sentences in Original abstract}}$$

We combine the 2 measures to provide a single measure known as *F1 Score* or *F-Measure*.

$$\text{F-Measure} = \frac{2 * \text{Recall} * \text{Precision}}{(\text{Recall} + \text{Precision})}$$

Computing *Precision and Recall* involves determining 'Sentence pairs' and we weight each 'Sentence pair' based on the similarity measure between the sentences.

The weighting mechanism is as below:

$$\text{Weight}(p_i, C(p_i)) = \begin{cases} 0 & \text{cosine similarity}(p_i, C(p_i)) < 0.4 \\ 0.5 & 0.4 \leq \text{cosine similarity}(p_i, C(p_i)) < 0.7 \\ 1 & 0.7 \leq \text{cosine similarity}(p_i, C(p_i)) \end{cases}$$

For our experiment we modify the way we compute Precision and Recall as

$$\text{Precision} = \frac{\sum_{i=1}^{M} Weight\ (p_{i,}C(p_i\ ))}{M}$$

$$\text{Recall} = \frac{\sum_{i=1}^{N} Weight\ (q_{i,}C(q_i\ ))}{N}$$

Wherein the sentence p_i and its counterpart $C(p_i)$ form a sentence pair and the denominators M and N are the number of sentences in the Auto-Generated and Original abstract respectively [1].

6.2.2.5 Abstract Evaluation

The section comment on the performance evaluation of four traditional Text summarizers versus our Text summarizer with the statistical results obtained by employing the evaluation techniques discussed above. The dataset used for testing the summarizers comprised of 25 papers selected in the field of 'Computer Science' (Source: http://www.arxiv.org/) where in the size of the documents were in the range of 3000 - 4000 words.

Please note that the experiments were conducted with a Compression Ratio CR of 10% i.e.

$$\text{CR} = \frac{\text{Summary size}}{\text{Actual text size}} = 10\%$$

SUMMARIZER	Degree	Betweenness	PageRank	Clustering Coefficient	Community Detection Algorithm
Precision	0.412	0.540	0.461	0.500	0.529
Recall	0.488	0.581	0.524	0.520	0.561
F-Measure	0.447	0.560	0.481	0.510	0.545

Table 6: Statistical comparison of the summarizers

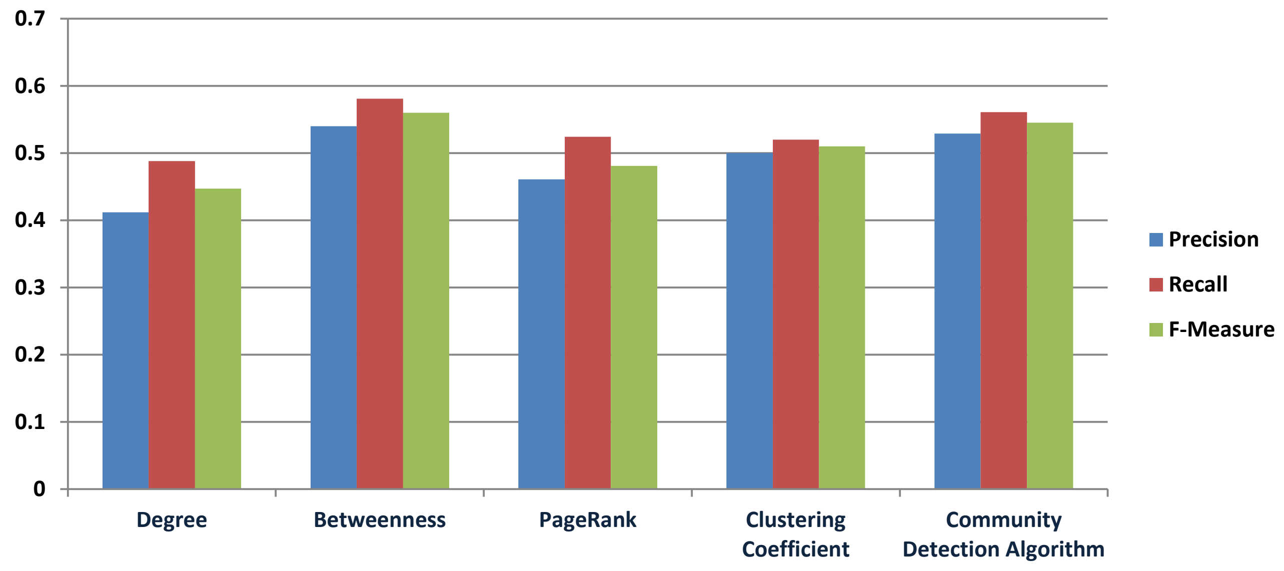


Figure 17: Pictographic representation of the statistical results obtained via Precision and Recall

From the statistical results obtained, the summarizer built employing 'Community Detection Algorithm' performed well and stood 2nd in performance, when the summarizers were put to test under Precision and Recall method.

6.2.3 Extrinsic Summary Evaluation [12]

The Summaries generated from the summarizers are subjective and hence hard to evaluate.

Shortcomings in Intrinsic Summary Evaluation

- The summarization technique employed involves mere extraction of vital sentences from the main document and doesn't matches with the actual summary of the document as the actual summary is not mere selection of sentences from the main document but a simplified and precise version.

- The evaluation technique employed above deals with computing similarity between the abstracts through word matching, which doesn't comment on the *meaning* of the abstracts. For instance the sentences

  Sentence 1: University of Bristol is a reputed university.
  Sentence 2: University of Bristol is *not* a reputed university.

  The similarity measure of the above sentences is 0.93541, but the meaning of the sentences contradicts each other.

Henceforth in order to evaluate the summary thoroughly we go for another approach known by Extrinsic summary evaluation which checks for correctness, appropriateness and acceptability of the computed summary through some task, for instance through a *Question Game* or a *panel of markers* who evaluate the quality of the summary generated.

6.2.3.1 Question Game

The purpose of the game is to test for correctness and the ability of the summary to convey core information of the main document. The game is as follows. Step 1: The testers go through the main document, identify and mark the core points. Now the testers frame questions on the identified core points. Step 2: Now the assessors are asked to answer the questions thrice.

- Without viewing any text (baseline 1).
- After viewing the Auto-Generated summary.
- After viewing the whole document (baseline 2).

An informative summary should be the one which is able to answer most of the questions correctly and should be nearer to baseline 2 than baseline 1.

For our summarizer evaluation, *Panel of Markers* approach was selected and the approach is as follows.

6.2.3.2 Panel of Markers Approach

Set of people who go through the Auto generated summary plus the original document and comment on the quality of the summary.

6.2.3.2.1 Experimental Framework

The experimental setup comprised of a panel of 25 markers and a set of 50 scientific papers along with their summaries generated via our text summarizer. The test primarily focussed on *summary coherence* and *summary informativeness*.

Summary Coherence: The summaries generated by the process of extraction involve mere extraction of the vital sentences from different parts of the core document and hence the summary lacks continuity or coherence when these vital sentences are collated.

Summary Informativeness: Is a measure of information retained in the condensed version.

6.2.3.2.2 Rating and Feedback

A panel of 25 markers were selected and each marker was given 2 papers along with their respective Auto-Generated summaries for evaluation and comment on the quality of the summary generated. The process comprised of the following steps.

Step 1: Markers go through the papers and summaries thoroughly.
Step 2: Check for Summary coherence and Summary Informativeness.
Step 3: Feedback session: The markers were engaged in a questionnaire and the questions were answered on a scale of 0 - 10, ranging from low to high.

6.2.3.2.3 Experimental Results

In the Feedback session the markers answered questions on *summary coherence* and *summary informativeness* on a scale of 0 - 10 and the results captured are as below.

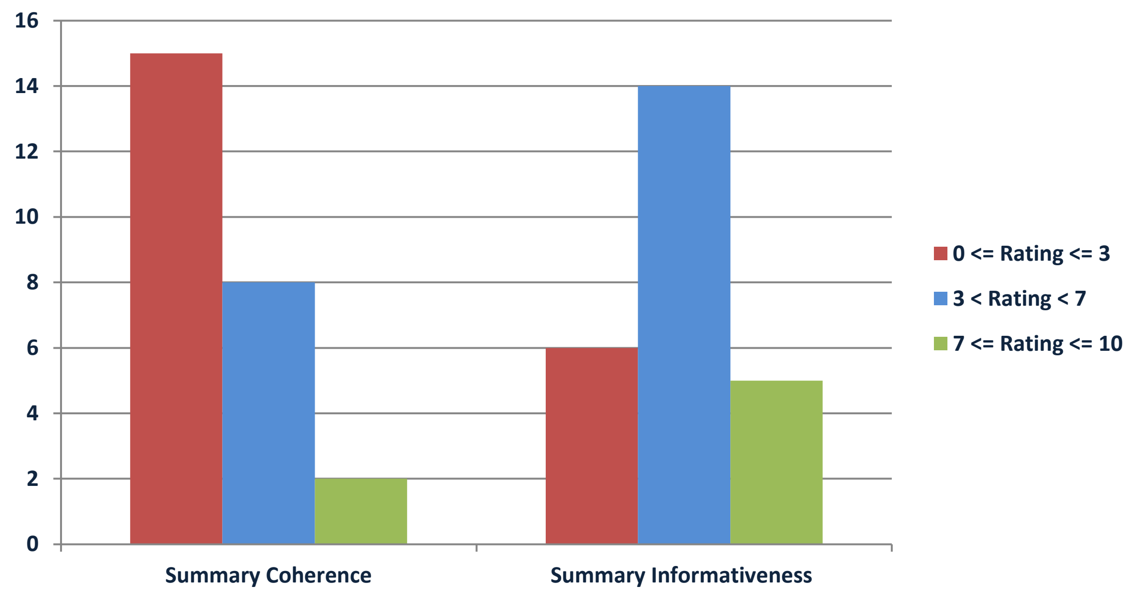


Figure 18: Statistical results obtained via Panel of Markers approach

Based on the experimental results we conclude summaries generated by the process of 'Extraction' lack *coherence* but fairly manage to withhold core content of the document.

6.3 Summary

The chapter looked at 2 aspects of summary evaluation

- In the first approach we compared the performance of our text summarizer with 4 related text summarizers and it was found the summarizer employing community detection algorithm performed well when put to test under *Precision and Recall* technique.

- In the second approach i.e. the *Panel of Markers* approach proved to be an efficient way of evaluation as the summaries generated are subjective and demands human intervention for efficient evaluation.

The above discussed 2 approaches were very much necessary for evaluating the performance of any summarizer and the summary generated out of them. Based on the results captured from the above discussed 2 evaluation techniques it was found the Text summarizer employing community detection algorithm proved to provide better results.

CHAPTER 7

CONCLUSIONS

Automated Text summarizers simplified the way we check for the core content from a huge document and aided in a more efficient and faster access of information. In our project we developed Automated Text summarizer which provides precise and concise information from .txt files.

The core tasks completed is as follows:

1. Successfully implemented 4 Traditional Automated Text Summarizers employing various network metrics. A detailed algorithm on each of the Text summarizer is discussed in Chapter 5.

2. Successfully integrated the concepts of Text Summarization, Graph Theory and Community Detection Algorithm and developed a novel, efficient Automated Text Summarizer.

    - Developed a Parser system to parse through document content and aid in extraction of necessary information by subjecting the Text document to a set of pre-processing techniques. The Parser system is developed in Java language and is presently able to process only .txt files. A comprehensive information and principle working of Parser system is discussed in Chapter 5.

    - Converted text documents to networks.

    - By employing an efficient community detection algorithm *InfoMap,* communities were detected from complex text networks, as sentences under each community carry vital information about the document it is the collation of significant sentences from each of the community forms the document abstract. A detailed working of the algorithm is covered in Chapter 5.

3. The performance evaluation of the Text summarizer is discussed in Chapter 6 and the chapter looked at 2 aspects of summary evaluation.

    - Approach 1# : Intrinsic Summary Evaluation

      We compared the performance of our text summarizer with 4 related text summarizers and it was found the summarizer employing community detection algorithm performed well when put to test under *Precision and Recall* technique.

    - Approach 2# : Extrinsic Summary Evaluation

      Based on the results obtained from the *Panel of Markers* technique, the summaries generated employing our algorithm managed to retain core content of the document and further improvisation can yield high performance, discussed in Chapter 7.

To conclude, the project covered an in depth research in the field of text summarization and an effort made from our side to contribute in the field of Text summarization and from the results obtained we feel the Automated Text summarizer developed by us will be an asset in the field of Automated Text Summarization.

CHAPTER 8

FUTURE ENHANCEMENTS

From the works of Luhn to till date lot of research has been carried out to better off the performance of traditional text summarizers. The Automated Text Summarizer developed by us is a basic version and there's always room for improvement by supplementing with add-on.

Below is the list of suggestions made in improving the efficiency of our summarizer.

1. Independent of Document format: The Parser system developed is capable of parsing only .txt files and demands flexibility in document format. In future we would like to develop an Intelligent Parser system which would be capable of parsing through any given document i.e. .txt, pdf, doc, latex or any other text formats.

2. Advanced Automated Text Summarizer: The summarizer developed is a basic version which employs sole community detection algorithm technique in keyword extraction and selection of significant sentences from the source document. In future we would like to add-on more efficient feature to the developed basic version for a much more efficient and advanced summarizer.

   Feature Add-ons are as follows:

   - Title Extraction
   - Sentence Location
   - Key Phrase
   - Sentence Length

   The features which yield better performance can be selected and detailed information on each of the feature is discussed in Chapter 2.

3. Abstraction based approach: As many other traditional text summarizers our approach is still Extraction based which involves mere extraction of the vital sentences from different parts of the core document and hence the summary lacks continuity or coherence when these vital sentences are collated. A major challenge in text summarization is in achieving a more human readable summary and hence demands for a more complex semantic natural language processing algorithms in future.

BIBLIOGRAPHY

[1] Yang Lei, “*Automatic Abstract Generator for scientific paper”*, University of Bristol, Sept 2006.

[2] Hans Peter Luhn, “*The Automatic Creation of literature Abstract”*, IBM Journal, April 1958.

[3] H. P. Edmundson, “*New Method in Automatic Extraction”*, Journal of the association for computing Machinery, April 1969.

[4] Santo Fortunato, “*Community detection in graphs”*, Complex Networks and Systems Lagrange Laboratory, ISI Foundation, Viale S. Severo65, 10133, Torino, I, Italy.

[5] Steve Gregory, “*Community structure in networks”*, University of Bristol, Department of Computer science.

[6] M. E. J. Newman, “*Detecting community structure in networks*”, 2004.

[7] Rodrigo Aldecoa and Ignacio Marín, “*Deciphering network community structure by Surprise*”, 2011.

[8] Lucas Antiqueira a, Osvaldo N. Oliveira Jr. a, Luciano da Fontoura Costa a, Maria das Graças Volpe Nunes b, “*A complex network approach to text summarization*”, 2009.

[9] Tom Scutt, Kirsten Cater and Dave Cliff, “*Graph Theory*”, University of Bristol, Department of Computer Science, Algorithmic and Economic Aspects of the Internet Computer Science COMSM2006, 2006.

[10] Diego Raphael Amancio, Eduardo G Altmann, Osvaldo N Oliveira Jr and Luciano da Fontoura Costa, “*Comparing intermittency and network measurements of words and their dependence on authorship*”, New Journal of Physics, December 2011.

[11] Tom Scutt, “*Cloud Algorithms: Big Data, Information & Preferences*”, University of Bristol, Department of Computer science, Algorithmic and Economic Aspects of the Internet Computer Science COMSM2006, 2006.

[12] Martin Hassel, "*Evaluation of Automatic Text Summarization*", KTH Numerical Analysis and Computer Science, Sweden 2004

[13] Document similarity using lucene, *Website*: http://stackoverflow.com/questions/10649898/better-way-of-calculating-document-similarity-using-lucene

[14] Günce Keziban Orman, Vincent Labatut, Hocine Cherifi, “*On Accuracy of Community Structure Discovery Algorithms*”, Galatasaray University.

[15] John Keyes (keyeg@ils.unc.edu), Porter Stemmer Code
http://www.google.co.uk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCMQFjAA&url=http%3A%2F%2Fwww.dbs.ethz.ch%2Feducation%2Fmmir%2FSS_01%2Fuebungen%2FPorterStemmer.java&ei=gDVkUKnFI8KGhQe2ooAg&usg=AFQjCNG78Rn7PZ-SiyPhH1TlEJljp4zN1Q

APPENDIX : A

SAMPLE RESULTS

Original Abstract

This paper provides theoretical foundations for the group signature primitive. We introduce strong, formal definitions for the core requirements of anonymity and traceability. We then show that these imply the large set of sometimes ambiguous existing informal requirements in the literature, thereby unifying and simplifying the requirements for this primitive. Finally we prove the existence of a construct meeting our definitions based only on the assumption that trapdoor permutations exist.

Auto Generated Abstract

"Group signatures are special type of signatures which allows individuals in a group to sign a message anonymously on behalf of the group. The paper defines a group signature scheme constituting 4 polynomial time algorithms and the scheme is proved secure considering the above 2 security properties. A group is a collection of members and a group manager and are associated with a pair of keys; a group public key, gpk (verification key) and all the members of the group is associated with their secret key through which they generate digital signature of any given message. Full Anonymity: We extend the security primitive of anonymity to Full anonymity by formalization under which the Adversary choose a message and sent to 2 group members to produce a digital signature and is returned with a signature of the message which is signed anonymously by one of the randomly chosen group member. Full Traceability: Is a strong form of collusion resistance meaning even when a group of members collude together by pooling their secret keys are unable to generate a valid signature on behalf of another group member, and holds true even if the colluding group is aware of the group manager's secret key, gmsk. Collusion resistance: Group of members collude together by pooling their secret keys are unable to generate a valid signature on behalf of another group member, and is true even if the colluding group is aware of the group manager's secret key, gmsk. The group signature scheme, GS = (GKg, GSig, GVf, GOpen) basically has 4 algorithms. Group verification algorithm GVf, checks whether the signature on m is a valid signature or not and outputs 1 if the signature on m is valid else returns 0."

NETWORK WORKBENCH

Introduction

Network Workbench is a complex network tool which supports:

- Pre-processing
- Modelling
- Analysis
- And Visualization of networks

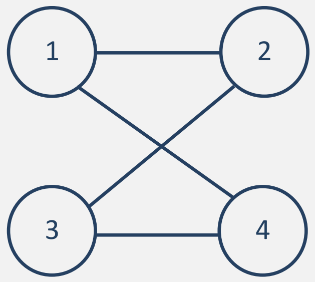
In our project we have made use of this tool in determining network metrics like the node Degree, Betweenness, PageRank and Clustering coefficient which aid in keyword determination in traditional Automated Text summarizers by feeding a network file known as Network workbench file or .nwb file

.nwb file

File format which provides comprehensive information on the network structure and the file syntax is as follows:

```
*Nodes
 Id*int     label*string
1           "word 1"
2           "word 2"
3           "word 3"
4           "word 4"

*UndirectedEdges
Source*int          target*int
1                   2
2                   3
3                   4
1                   4
```



The Upper portion of the file provides information about the number of nodes and the unique number sequence associated with the word. And the lower portion of the file gives information on the edge details, i.e. which node is connected to which other node in the network. Beside is the network generated when the .nwb file is visualized employing *Network Workbench*.

APPENDIX : B

SOURCE CODE

Automated Text Summarizer: Community Detection Algorithm

Below core portion of Summarizer code illustrates the way sentences are checked for significance and Abstract is generated.

```
// Outer Loop

while(outer < N)                                                        // N - Number of communities
{
    System.out.println("Community number ----------" + outer);          // To display community number
    eachCommunityKey = communityKeywords[outer];                        // Store each community content in an array
    if(eachCommunityKey.length == 0)
           break;

    // Inner Loop is executed for each community

    HashMap<Integer, Integer> vital_Sentences = new HashMap<Integer, Integer>();

    for (int keyCounter = 0 ; keyCounter < eachCommunityKey.length  ; keyCounter++)
    {
            if(eachCommunityKey[keyCounter] == null)
            {
                   System.out.println("Community size------------------- " + (keyCounter));
                   break;
            }
            System.out.println("loop2 -" + keyCounter);

            // Sentence significance is computed based on the occurrence of Keywords in the sentence

            for(int sentCounter = 0 ; sentCounter < sentence_Indexer.size(); sentCounter++)
            {
                   System.out.println("Sentence number :"+ sentCounter);
                   String sentence = sentence_Indexer.get(sentCounter);
                   StringTokenizer tokens = new StringTokenizer(sentence, " ");

                   while(tokens.hasMoreTokens())
                   {
                         String token = tokens.nextToken();

                         if(token.startsWith(eachCommunityKey[keyCounter])||token.equalsIgnoreCase(eachCommunityKey[key
                         Counter]))
                         {
                                 System.out.println ("Token   : " + token + "     community Keyword   :" +
                                 eachCommunityKey[keyCounter]);

                                 if(vital_Sentences.containsKey(sentCounter))
```

```
                                        {
                                                int value = vital_Sentences.get(sentCounter);
                                                value++;
                                                vital_Sentences.put(sentCounter, value);
                                                System.out.println(sentCounter + "-------------" + value);
                                        }
                                        else
                                        {
                                                vital_Sentences.put(sentCounter, 1);
                                                System.out.println(sentCounter + "-------------" + 1);
                                        }
                                }
                                }
                                }
                }

                // Sentences are sorted in descending order based on significance

                sorted_Sentences = net.sortHashMapByValuesD(vital_Sentences);

                // Compression Ratio is ratio of Abstract size to Actual document size = 0.1

                int compressionRatio = (int) ( 0.9 * sorted_Sentences.size());
                System.out.println();
                System.out.println("Sentence count  :" + sorted_Sentences.size() + "    Compression ratio   :" + compressionRatio);
                List<Integer> keyset = new ArrayList<Integer>(sorted_Sentences.keySet());
                Collections.reverse(keyset);
                for(int key : keyset)
                {
                        compressionRatio++;
                        if((compressionRatio + 3) > sorted_Sentences.size())
                                break;
                        System.out.println("Key : " + key + "   Occurence : " + sorted_Sentences.get(key));
                        sortKey.add(key);
                }
                outer++;
        }

        // The most significant sentences are arranged in order of their existence in the original document

        Collections.sort(sortKey);
        Set<Integer> sentences = new HashSet<Integer>();
        System.out.println("---------------------------------"); System.out.println();
        for(int key : sortKey)
        {
                sentences.add(key);
        }
        System.out.println();
        sortedFinalKeyList = new ArrayList<Integer>(sentences);
        Collections.sort(sortedFinalKeyList);
        for(int key : sortedFinalKeyList)
        {
                System.out.println("key :" + key );
        }
        System.out.println();
```

```
  // Display Abstract

   System.out.println("------------------ SUMMARY ----------------------"); System.out.println();

   for(int key : sortedFinalKeyList)
   {
            System.out.print(sentence_Indexer.get(key)+". ");
   }
   System.out.println(); System.out.println();
   System.out.println("---------------- END OF SUMMARY --------------------");

   }

    // Catch Block

    catch(Exception ex)
    {
             System.err.println("Error message "+ ex.getMessage());
    }
  }
}
```