VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELGAUM – 590014

A Project Report on

"Abstractive Multi-Document Text Summarization"

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Engineering
in
Information Science & Engineering

Submitted by:

AKSHATA BHAT	**1PI11IS008**
K R ANUSHREE	**1PI11IS127**

Under the guidance of

Internal Guide
Dr. S. Natarajan
Professor
Department of IS & E,
PESIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

PES INSTITUTE OF TECHNOLOGY

100 Feet Ring Road, BSK 3rd Stage, Bengaluru – 560085

January 2015 – May 2015

PES INSTITUTE OF TECHNOLOGY

100 Feet Ring Road, B S K 3rd Stage,
Bengaluru-560085

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled **"Abstractive Multi-Document Text Summarization"** carried out by **Akshata Bhat**, bearing USN **1PI11IS008**, **K R Anushree**, bearing USN **1PI11IS127**, are bonafide students of **PES INSTITUTE OF TECHNOLOGY**, Bangalore, an autonomous institute, under VTU, in partial fulfillment for the award of degree of **BACHELOR OF ENGINEERING IN INFORMATION SCIENCE & ENGINEERING** of **Visvesvaraya Technological University, Belgaum** during the year **2015**. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the above said degree.

Dr S Natarajan Internal Guide Professor Department OF ISE PESIT	**Dr. Shylaja S S** HOD Department OF ISE PESIT	**Dr. K. S. Sridhar** Principal & Director PESIT

External Viva

Name of the Examiners **Signature with Date**

1._____ _____

2._____ _____

Abstract

In today's world, there is lack of knowledge of events that happens around us. There is always an imperativeness of business news in one's life. A fence of knowledge has been formed due to the value of latest business news. A person should always keep proper track of the business news, for knowing the direction of the market as well as the market flow, for better business and better performance.

A lot of news websites are available to public on web, which generate hundreds of articles every day. Each news website presents its own perspective of the event under discussion and also put forth their opinions which could be a stark contrast to some other news website's presentation of the same event. Hence, it becomes very important to have a general overview of these opinions. This generates the need to have a concise summary of different articles which can convey the most important information. Manual summarization is a tedious, monotonic and time consuming task. Therefore, there is a need for an Automatic Text Summarization.

The project purports to generate an abstract summary of different articles which discuss similar events. We explore different algorithms to do the same, which have earlier not been explored for generating this type of summaries. The highlight of the project is to generate the **abstract text summaries for multiple news articles.**

Acknowledgement

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible, and whose guidance and encouragement helped us in completing the project successfully. We consider it a privilege to express gratitude and respect to all those who guided us throughout the course of the completion of the project.

We extend our sincere thanks to **Dr. S. Natarajan**, Professor, Department of Information Science and Engineering, PESIT, our project guide, for his constant guidance, encouragement, support and invaluable advice without which this project would not have become a reality.

We would like to thank our project coordinator, **Prof Raj Alankar**, Professor, Department of Information Science and Engineering, PESIT, for creating a supportive environment to carry out the project.

We express our gratitude to **Dr. Shylaja S S**, Head of the Department, Information Science, PESIT whose guidance and support has been invaluable.

We would like to express our heartfelt thanks to **Dr. M. R. Doreswamy**, Founder Secretary, PES Institutions, **Prof. D. Jawahar,** CEO, PES Institutions and **Dr. K. S Sridhar**, Principal and Director, PESIT for providing us with a congenial environment for carrying out the project.

Last, but not the least, we would like to thank our friends whose invaluable feedback helped us to improve the software by leaps and bounds, and our parents for their unending encouragement and support.

Table of Contents

List of Figures

List of Tables

Chapter - 1

Introduction

In this section, we provide a brief introduction to our project "Abstractive Multi-Document Text Summarization". We will give an overview of the field of Summarization, followed by a precise description of the problem we are trying to solve, the project specific goals, motivation for us to choose the problem, the scope of our project and the methodology we adopt to solve the problem. We conclude the section by providing brief overview of the algorithms that we will be implementing to address the problem.

1.1 Overview

A summary of a document is a (much) shorter text that conveys the most important information from the source document. There are a number of scenarios where automatic construction of such summaries is useful. For example, an information retrieval system could present an automatically built summary in its list of retrieval results, for the user to quickly decide which documents are interesting and worth opening for a closer look—this is what Google models to some degree with the snippets shown in its search results. Other examples include automatic construction of summaries of news articles or email messages to be sent to mobile devices as SMS; summarization of information for government officials, businessmen, researches, etc., and summarization of web pages to be shown on the screen of a mobile device, among many others.

Text summarization tasks can be classified into single-document and multi-document summarization. In single-document summarization, the summary of only one document is to be built, while in multi-document summarization the summary of a whole collection of documents (such as all today's news)

is built. In this project, we have experimented with multi-document summarization.

The summarization methods can be classified into abstractive and extractive summarization. An abstractive summary is an arbitrary text that describes the contexts of the source document. Abstractive summarization process consists of "understanding" the original text and "retelling" it in fewer words. An extractive summary, in contrast, is a selection of sentences (or phrases, paragraphs, etc.) from the original text, usually presented to the user in the same order—i.e., a copy of the source text with most sentences omitted. An extractive summarization method only decides, for each sentence, whether or not it will be included in the summary. An abstractive summary can be further obtained from the extracted summary using techniques of sentence compression, theme intersection algorithms and also bring the cohesiveness in generated summary by using sentence ordering algorithms. The project aims to build abstractive summaries from the extractive summaries generated.

1.2 Problem Definition

As stated in the Introduction, this project generates an Abstractive summary for multiple business news articles. With various techniques been developed for text summarization so far, the goal of the project is

- To analyze the performance of some of existing techniques for document summarization by executing them.
- To experiment with a new machine learning technique, which has not been used for summarization.
- To design a technique to perform Abstractive Multi-Document Summarization from the extractive summaries generated.
- Evaluate their performance on the basis of few pre-defined parameters.

With a large number of news articles on the same event or topic, with different perspective or context, going through each of them is a time consuming task. The goal is to generate a precise cohesive summary. The project uses machine learning, statistical and natural language processing techniques for generating summary. The summarization would be performed on the business news articles that are collected using RSS feeds and crawling. Dataset would be created by collecting articles from various news websites. Subset of articles, are those that are about the same topic, which would be obtained using a clustering technique. Once the subset of documents is obtained, various techniques would be used to generate the summary and hence the performance of techniques would be evaluated.

1.3 Goals

The "**Abstractive Multi-document Text Summarization**" project aims at generating concise abstract summaries of multiple documents that describe similar events. It extracts the most important sentences from each document and outputs a compressed short abstract summary from it. The project analyses the results of using "Deep-Learning", "Lexical Chain" and "Support Vector Clustering" techniques to generate summaries.

The technical goals that has to be achieved by this project are :-

1. **Build Data Corpus of Business News Articles** - The business news articles corpus to be built by crawling the links obtained from rss feeds of various news websites.
2. **Document Clustering based on Event Similarity** - The documents / articles obtained from various sources to be clustered based on the event they describe. This clustering should happen at an optimal level, that is not too generalized or too descriptive.

3. **Extract Important Sentences** -The most important sentences from different documents forming a single cluster to be extracted based on different features, which will contribute to the summary for the event described in that particular cluster. The process to be repeated for different event clusters. Different techniques to be used to extract these sentences - Deep Learning, Support Vector Clustering and Lexical Chains.
4. **Generate Abstractive Summary** - The abstractive summary to be generated from the extracted sentences using linguistic and compression methods.
5. **Analysis** - Analyze the summaries generated from different techniques and compare their performance based on pre-defined parameters.

1.4 Motivation

The digital data available to us on World Wide Web is growing at an exponential pace. In the current era of information overload, text summarization across several documents which cover same subject has become an important and timely tool for user to quickly understand the large volume of information. For humans, generating a summary is a straightforward process but it is time consuming to explore huge amount of multi-source data available for same subject. Therefore, there is need for automatically generating the summary and getting the general idea of long textual documents.

1.5 Project Scope

The deliverable "**Abstractive Multi-document Text Summarization**" of project is implementation of different algorithms for performing the abstractive text summarization on multiple documents/articles available for similar event. The performance of these algorithms will be compared based on parameters - grammar, focus, coverage, length, structure, non-redundancy.

There is a need for a concise representation of vast amount of information available to us in the form of digital data. Several business news websites are available to us today to refer to about a particular event that is happening around. However, the articles of each of the websites differ in the content to some extent and these contents can represent different perspective. Due to lack of time, an enthusiastic news follower may not be able to go through each of them. Hence the project aims to generate a concise summary of all these articles.

The other dimension of the project - is the research aspect of it. Different machine learning, statistical and natural language techniques will be chosen to perform the summarization. These techniques will not have been used previously for generating summaries. The performance of the techniques will be analyzed. This analysis can serve as a small contribution to the research in this field.

1.6 Methodology

The "**Abstractive Multi-document Text Summarization**" project follows a set of actions in order to achieve the completion of project. The methodologies planned for the execution of this project are:

1. Building the data corpus of Business News articles by using RSS feeds and web crawling. With the help of RSS feeds the links of the news articles to be obtained. Using the link, the web page containing the article should be parsed and hence the complete description to be obtained. The data collected to be stored in a json file.
2. A suitable language has to be selected to code the modules in. Choosing a suitable language is important. The chosen language must have modules that will help the developers in implementing the algorithms with minimal effort.
3. A set of good feature functions for the documents and sentences need to be chosen. These chosen feature function decide the performance of the algorithm to generate the summary. Good feature functions ensure lesser convergence time of algorithm and hence better results.
4. The machine learning and statistical techniques that would be used for the algorithms to be implemented needs to be chosen. Also a suitable document clustering technique needs to be identified and the level of clustering should be decided.
5. Once the algorithm implementation is done, an interface should be built so that a common user can interact with the application. The interface will be Web based as majority of the people use web for innumerable purposes.

1.7 Algorithms

In this section, we discuss different algorithms that will be used in the project. These include algorithm for document clustering and various techniques and algorithm to generate summaries.

1.7.1 Document Clustering

A definition of clustering in simple words could be "the process of organizing objects into groups whose members are similar in some way". A *cluster* is therefore a collection of objects which are coherent internally, but clearly dissimilar to the objects belonging to other clusters. Therefore, a **cluster of documents** consists of those documents which are similar in contents. Hence, they are a group of documents that speak or discuss similar events.

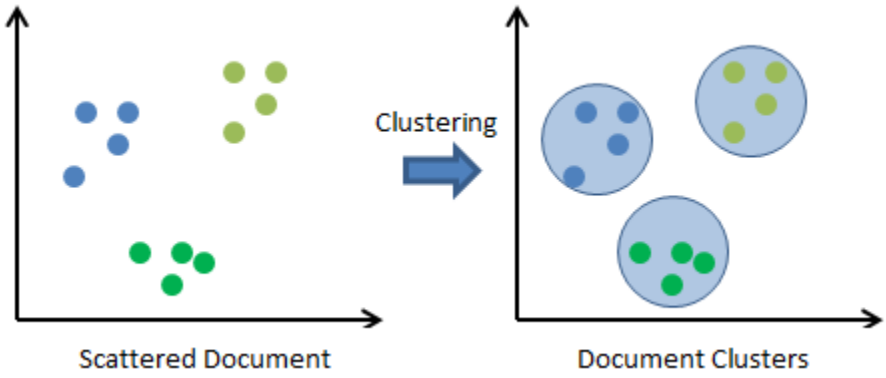


Fig 1.1 : Document Clustering

The similarity criterion is distance: two or more objects belong to the same cluster if they are "close" according to a given distance (in this case cosine similarity). This is called distance-based clustering.

The **k-means clustering** algorithm is known to be efficient in clustering large data sets. This clustering algorithm is one of the simplest and the best known unsupervised learning algorithms that solve the well-known clustering

problem. The K-Means algorithm aims to partition a set of objects, based on their attributes/features, into *k* clusters, where *k* is a predefined or user-defined constant. The main idea is to define *k* centroids, one for each cluster. The centroid of a cluster is formed in such a way that it is closely related.

We do the clustering at two levels - first level, apply the algorithm for all documents to obtain *m* clusters and second level, apply the algorithm on each of the clusters obtained earlier, to generate *n* clusters. At the end we will be having *m* times *n* clusters, for each of which we need to generate the summary.

The general steps in k-means clustering algorithm are given in the figure below.

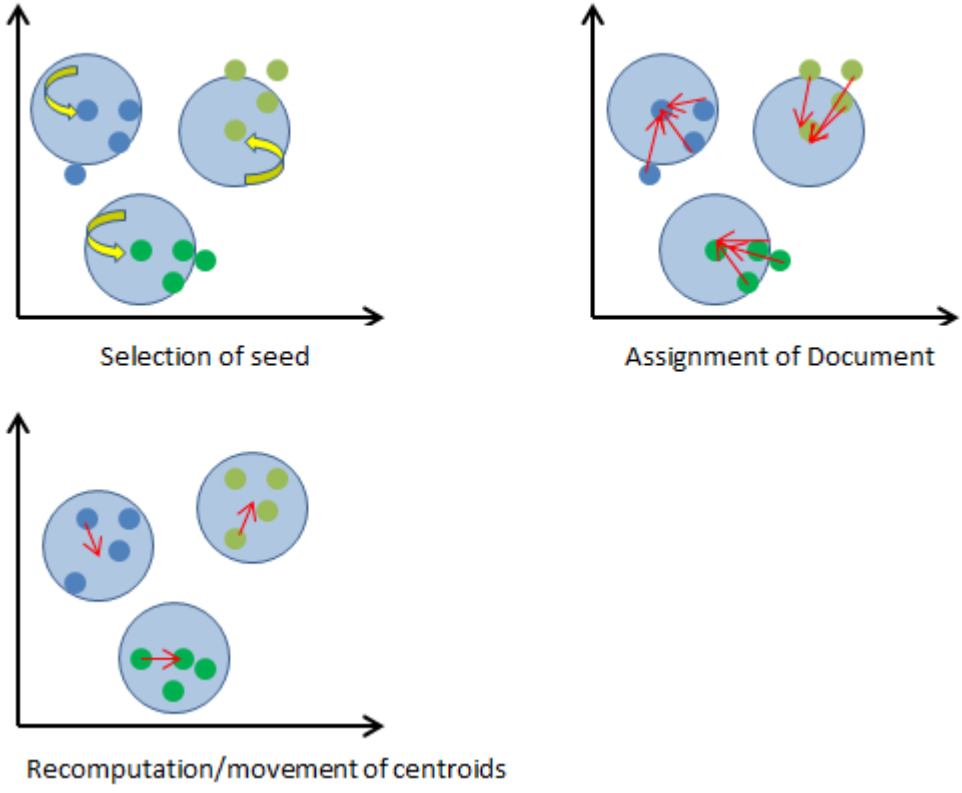


Fig 1.2 : K-means algorithm for Documents Clustering

1.7.2 Algorithms for Summary Generation

In this section, we discuss three different techniques/algorithms that will be implemented to obtain summaries. These are - "Deep Learning", "Support Vectors for Clustering" and "Lexical Chains".

1.7.2.1. Deep Learning

Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text. The algorithm which we have implemented is Restricted Boltzmann Machine.

Restricted Boltzmann Machine is a stochastic neural network. It is a network of neurons where each neuron has some random behavior when activated. It consists of one layer of visible units (neurons) and one layer of hidden units. Units in each layer have no connections between them and are connected to all other units in other layer. Connections between neurons are bidirectional and symmetric. This means that information flows in both directions during the training and during the usage of the network and those weights are the same in both directions.

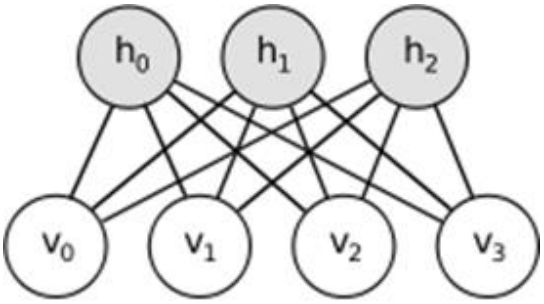


Fig 1.3 : Restricted Boltzmann Machine

Initially, the network is trained by using the data set and setting the neurons on visible layer to match data points in this data set. Once the network is trained it can be used on new unknown data to make classification of the data (this is known as unsupervised learning). For summarizing the text there is a need of structuring the text into certain model which can be given to RBM as input. First of all the density of the document is reduced by using various preprocessing techniques and then it is converted into sentence matrix.

	T	P	Tw	C
S1				
S2	f1	f2	f3	f4
.	..	....	..	..
.	..	....	..	..
Sn	....	..	..	..

Fig 1.4 : Sentence matrix

A sentence matrix S of order n*v is containing the features for every sentence of a matrix. This structured matrix is the input to our RBM as a visible layer.

The free energy formula of an RBM is defined as :-

$$\mathcal{F}(v) = -b'v - \sum_i \log(1 + e^{(c_i + W_i v)}).$$

The visible and hidden units are conditionally independent given one-another. Probabilistic version of neutral activation function :

$$P(h_i = 1|v) = sigm(c_i + W_i v)$$

$$P(v_j = 1|h) = sigm(b_j + W'_j h)$$

The Gibbs chain is initialized with the hidden sample generated during the positive phase, therefore implementing Contrastive Divergence (CD). CD is a recipe used for training the undirected graph. It relies on an approximation of the gradient of the log-likelihood based on a short Markov chain started. One step of CD is performed to get the cost updates. We hence obtain a good set of feature functions. In second phase, obtained feature vectors are fined tuned by adjusting the weights of the units of the RBM. Back propagation algorithm is used to fine tune it. Once the RBM is trained, sampling is done. Samples of P(x) is obtained by running a Markov chain to convergence, using Gibbs sampling as the transition operator. Gibbs sampling of the joint of N random variables $S = (S_1, ..., S_N)$ is done through a sequence of N sampling sub-steps of the form $S_i \sim p(S_i|S_{-i})$ where S_{-i} contains the $N-1$ other random variables in S excluding S_i. The visible units are sampled simultaneously given fixed values of the hidden units. Similarly, hidden units are sampled simultaneously given the visible units. A step in the Markov chain is thus taken as follows:

$$h^{(n+1)} \sim sigm(W'v^{(n)} + c)$$
$$v^{(n+1)} \sim sigm(Wh^{(n+1)} + b)$$

$h^{(n)}$ refers to the set of all hidden units at the nth step of the Markov chain.

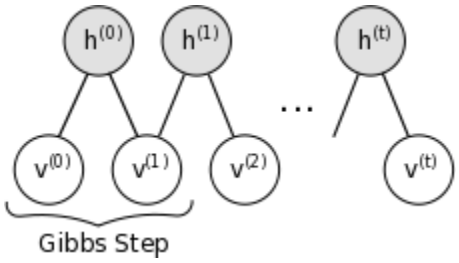


Fig 1.5 : One step of Gibbs Sampling

The optimal feature vector set is generated by obtaining the sample at the end of the chain. Threshold values are set for the feature functions. These

threshold values are obtained based on the experiments. The sentence with feature vector, that satisfies these threshold values, is selected.

1.7.2.2 Support Vector for Clustering

In the Support Vector Clustering (SVC) algorithm, data points are mapped from original data space to a high dimensional feature space using a Gaussian kernel. Using a nonlinear transformation Φ from χ to some high dimensional feature-space, we look for the smallest enclosing sphere of radius R, for the image of data. This is described by the constraints:

$$||\Phi(x_j) - a||^2 \leq R^2 \; \forall j$$

This sphere is mapped back to data space, where it forms a set of contours which enclose the data points. These contours are interpreted as cluster boundaries. Points enclosed by each separate contour are associated with the same cluster. The shape of the enclosing contours in data space is governed by two parameters: q, the scale parameter of the Gaussian kernel, and C, the soft margin constant.

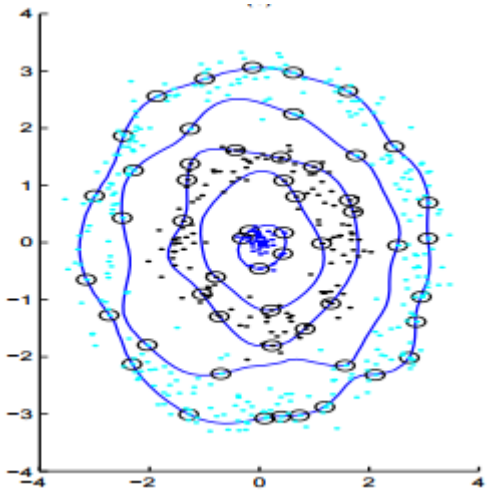


Fig 1.6 : Clustering of Data points after mapping them to feature space

As the width parameter of the Gaussian kernel is decreased, the number of disconnected contours in data space increases, leading to an increasing number of clusters. SVC is a nonparametric clustering algorithm that does not make any assumption on the number or shape of the clusters in the data.

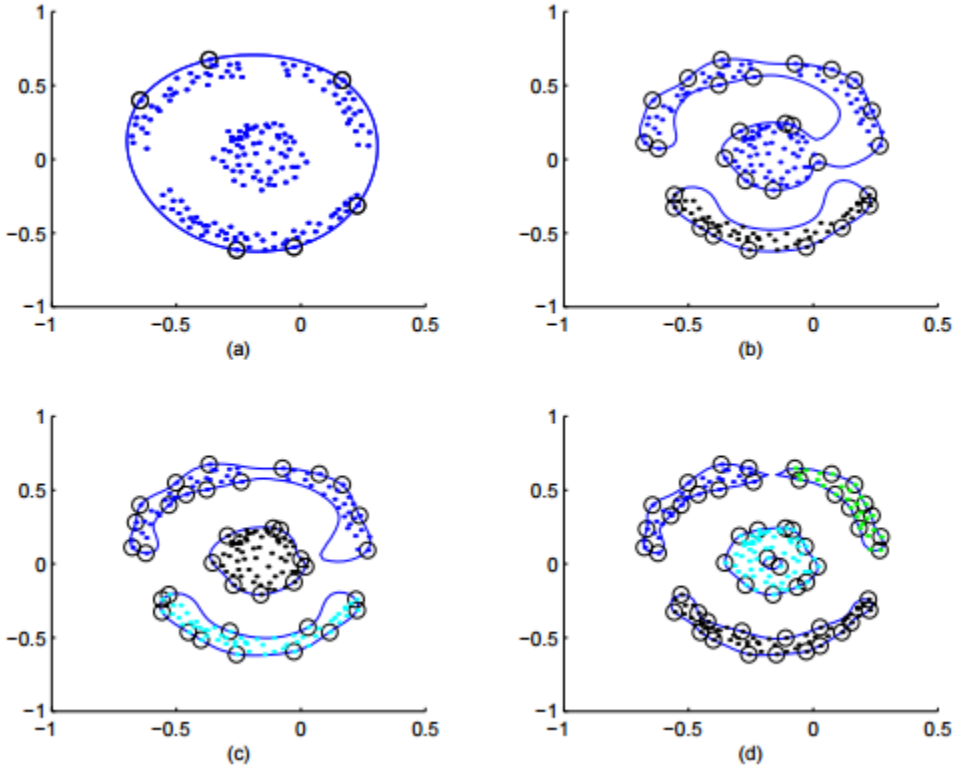


Fig 1.7 : Contours generated by SVC as q is increased.

Each sentence in our news article collections, to be summarized, is represented as feature vectors. The image of these feature vectors are transformed into higher dimension using Gaussian Kernel function. Also, a set of random data points which lie on same hyper-plane as that of the original data is generated. The original data-points form one class - Class A, and the randomly generated data-points form the other class - Class B. SVM is applied to separate Class A from Class B. In the process we obtain

representative vectors, vectors which represent the clusters obtained for the data. These representative vectors are nothing but the non-bounded support vectors whose Lagrange's multiplier is less than C (Gaussian Kernel). Then for each vector, we identify the cluster representative to which it is closest and add it to the respective cluster. This algorithm is also known as Find and Join Clusters (FJC). The distance between vectors is computed using the cosine similarity.

Once clusters of sentences are obtained, we need to score the sentences, so that the most important sentence from each cluster can be selected to be part of the summary. We need a sentence scoring algorithm, where sentences are scored based on different features and weights corresponding to each feature. We then sort sentences based on score and choose the highest scored sentence. The brief outline of algorithm is depicted in the figure below.

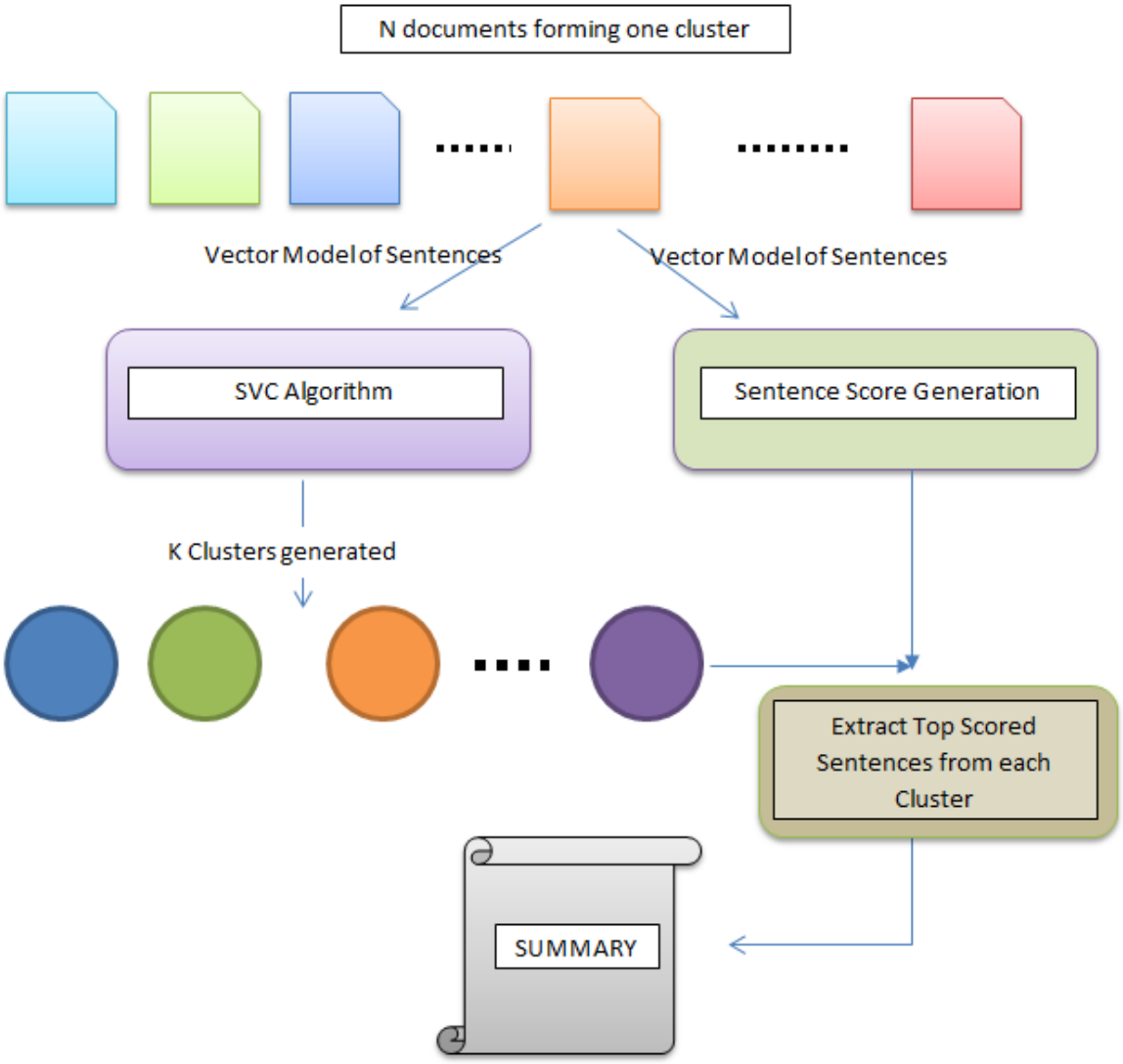


Fig 1.8 : Applying SVC for Multi-Document Summarization(extractive)

1.7.2.3 Lexical Chain

A Lexical chain is a sequence of related words in the text. It spans short (adjacent words or sentences) or long distances (entire text). It is independent of the grammatical structure of the text. It captures the cohesive structure of the text.

For summarization using Lexical Chains, the following steps are performed :

1. Sentences and words are tokenized. Part of speech tagging is performed.
2. Lexical Chains are constructed
3. Lexical chains are scored
4. Significant sentences are then extracted.

Generally, a procedure for constructing lexical chains follows three steps:

1. Select a set of candidate words. Nouns and Noun Phrases are considered as candidate words. Part of Speech Tagger module of NLTK library is used to tag the data. Once the data is tagged, candidate words are extracted.

2. For each candidate word, find an appropriate chain relying on a relatedness criterion among members of the chains;

3. If it is found, insert the word in the chain and update it accordingly.

WordNet lexical database is used by this algorithm for determining the relatedness of the words. Senses in WordNet databases are represented relationally by synonym sets ('synsets') - which are the sets of all the words sharing the common sense. Consider the following example : -

"***Mr****. Kenny is the* ***person*** *that invented an* ***anesthetic machine*** *which uses* ***micro-computers*** *to control the rate at which an anesthetic is pumped into the blood….* "

"Mr." is first created [lex "Mr.", sense {mister, Mr.}]. "Mr." belongs only to one 'synset', so it is disambiguated from the beginning. The next candidate word "person" has two senses "human being" and "grammatical category of pronouns and verb forms". Hence the chain is split into two related to this chain in the sense "a human being" by a medium-strong relation.

The two chain entries are -

[lex "Mr.", sense {mister, Mr.}]

[lex "person", sense {person, individual, someone, man, mortal, human, soul}].

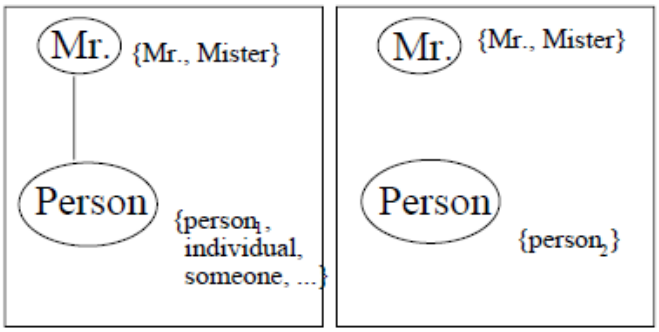
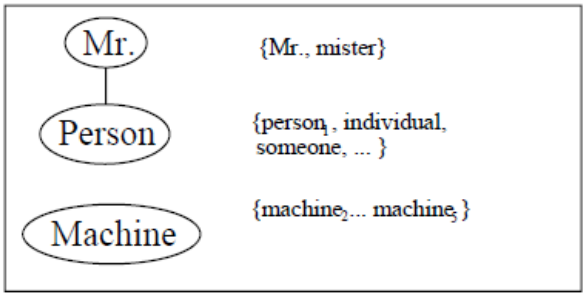


Fig 1.9 : LC Step 1, Interpretations 1 and 2

The next candidate word "Machine" has five senses. The sense "efficient person", hence the word is inserted into both the chains.



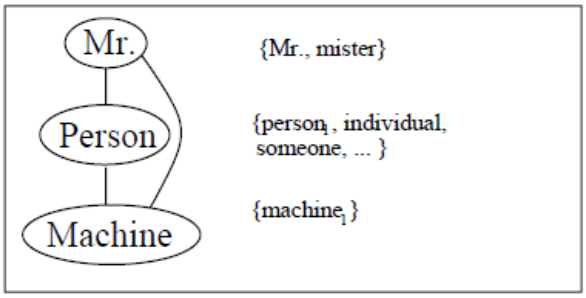


Fig 1.10 : LC Step 2, Interpretation 1 Fig 1.11:LC Step 3, Interpretation 2

Continuing this for all the candidate words, the lexical chains are generated. The lexical chains are scored using relation type, sentence distance and paragraph distance. The lexical chains are arranged in descending order of their score. For each chain in the summary representation, the sentence that contains the first appearance of a representative chain member in the text is chosen.

Chapter - 2

Literature Survey

This section deals with discussion of some of the existing techniques, a survey of work carried out by researchers in the domain of Text Summarization. This survey was carried out to know the existing techniques that are used for Text Summarization.

Text Summarization

With the explosion of information supplied by the growth of the World Wide Web, it is no longer suitable for a human observer to understand all the data coming from diverse sources. With this growth of information and available computing power, automatic classification and summarization of textual data gains increasingly high importance. The idea of **Text Summarization** is to summarize multiple text documents by extracting sentences that have high importance and forming an abstract summary from it.

Steps in Text Summarization Process

Text Summarization process consists of several steps. Broadly, they are data acquisition from various sources, document clustering to group the documents based on their similarity with respect to the content and event discussed in it, preprocessing, applying algorithm to generate summary, constructing the abstract summary and finally the evaluation of summary generated.

Daniel Jacob Gillick [1] describes practical issues associated with summarization. Preprocessing is the stage of preparing document suitable for further processing. Many segments of raw text are not suitable for extraction. Tables, quotes, bylines, and all kinds of formatting markup are unwelcome in a summary, and should be removed before analysis begins. Some of the techniques are cleaning up formatted text, sentence compression, and sentence segmentation. Sentence selection is the next

step. To choose which sentences to include in a summary, we need some way of measuring the value of each sentence, or more generally, selection unit. The techniques that can be used are - sentence scoring based on features like sentence length, topic, frequency of words, position of the sentence, document length etc., maximal marginal relevance, graphical algorithms, topical models.

The other steps of summarization described along with techniques are sentence ordering and evaluation. With respect to sentence ordering, he describes that the ordering of a set of sentences can dramatically affect the meaning and readability of the summary. While a proper treatment of sentence ordering ought to be more holistic, that is, considered jointly with selection, little work in this area has had much impact.

Measuring and understanding the differences between different summarization systems is crucial to advancing the field. A variety of automatic and semi-automatic tools are used. The somewhat standard, though problematic ROUGE metric measures word overlap between machine-generated summaries and a set of human-written abstracts. Evaluation can also be performed manually. Some parameters to evaluate the generated summaries are - sentence responsiveness, grammatical correctness, non-redundancy, referential clarity, focus, structure and cohesiveness.

Document Clustering

Document Clustering, or **Text Clustering**, is a subfield of data clustering where a collection of documents are categorized into different subsets with respect to document similarity. Such clustering occurs without supervised information, i.e., no prior knowledge of the number of resulting subsets or the size of each subset is required.

Vasileios Hatzivassiloglou, Luis Gravano, Ankineedu MagantiAn [2] investigate four hierarchical clustering methods - single-link, complete-link,

groupwise-average, and single-pass. Single-pass clustering makes irrevocable clustering assignments for a document as soon as the document is first inspected. Among the four techniques that they have considered, single-pass is then the best suited for the topic detection task, which requires systems to make clustering assignments "on-line" as soon as a new document is received. To investigate the limitations of such an on-line algorithm, they also experimentally compare the performance of single-pass with the other three clustering algorithms mentioned above.

The other component of a clustering strategy that they explore in this paper is the document features that guide the clustering. Typically, document clustering techniques use the words that appear in the documents to define the "distance" function that determines the final clustering. But additional, more linguistically informed sets of features can be used in an attempt to limit the input features to the most important ones, facilitating the task of the learning (i.e., clustering) algorithm. In this paper they also investigate two such sets of automatically identified features: matched noun phrase heads, where additional pre-modifiers are excluded, and proper names (single nouns and phrases), categorized as people, place, or organizations' names.

Kathleen R. McKeown and her team at Columbia University [3] describe the document clustering technique deployed in their product - Newsblaster. Newsblaster hierarchically classifies the news stories gathered by the crawler into three levels. At the top level, cosine similarity is used to compute the similarity between the news category and news article. At the next two lower levels, the system uses agglomerative clustering with a groupwise-average similarity function to group and identify similar events. It also incorporates a log-linear statistical model for automatically adjusting the relative weights of the different features.

Michael Steinbach George Karypis Vipin Kumar [4] present the results of an experimental study of some common document clustering techniques in

their paper. In particular, they compare the two main approaches to document clustering, agglomerative hierarchical clustering and K-means. Hierarchical clustering takes quadratic time whereas K-means takes linear time to perform clustering.

Extractive summarization

Extractive Summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form. The importance of sentences is decided based on statistical and linguistic features of sentences.

Vishal Gupta [5] discusses several methods of summarization used in past, namely Term Frequency-Inverse Document Frequency (TFIDF) method, Clustering based method, Graph theoretic approach, Machine Learning approach, LSA, An approach to concept-obtained text summarization, Text Summarization with Neural Networks, Automatic text summarization based on fuzzy logic and Text summarization using regression for estimating feature weights. The features that can be used for sentence selection and ordering are - Content word, title similarity, Noun phrase, title word, occurrence of non-essential terms such as "furthermore", "additionally", discourse based information, upper-case, biased words, pronouns, uppercase, sentence location, sentence length, sentence to centroid cohesion.

The paper also includes survey on NeATS. NeATS, a **Multi-Document Summarizer,** generates summaries in three stages: content selection, filtering, and presentation. The goal of content selection is to identify important concepts mentioned in a document collection. In a key step for locating important sentences, NeATS computes the likelihood ratio to identify key concepts in unigrams, bigrams, and trigrams, using the on-topic document collection as the relevant set and the off-topic document collection as the irrelevant set. With the individual key concepts available,

these concepts are clustered in order to identify major subtopics within the main topic. Clusters are formed through strict lexical connection. Each sentence in the document set is then ranked, using the key concept structures. NeATS uses three different filters: sentence position, stigma words, and maximum marginal relevance.

Dipanjan Das, Andre F.T. Martins [6], present a survey which intends to investigate some of the most relevant approaches both in the areas of single-document and multiple-document summarization, giving special emphasis to empirical methods and extractive techniques. Some promising approaches that concentrate on specific details of the summarization problem are also discussed. Special attention is devoted to automatic evaluation of summarization systems, as future research on summarization is strongly dependent on progress in this area. They describe in their paper various techniques used for generating summaries automatically, for each of the types - Single Document Summarization and Multi Document Summarization.

Under Single Document Summarization, they present a survey on execution and performance of - Machine Learning Methods, namely Naive Bayes, Rich Features and Decision Trees, Hidden Markov Models, Log-Linear Models, Neural Networks, - Deep Learning Language Methods. Under these they discuss two different discourse rhetorical structure - a binary tree representing relations between chunks of sentence and the one based on heuristics with the traditional features that have been used in the summarization literature. The discourse theory the Rhetorical Structure Theory (RST) that holds between two non-overlapping pieces of text spans: the nucleus and the satellite. Under Multi Document Summarization, they present a discussion on techniques namely Abstraction and Information Fusion, Topic Driven Summarization and MMR, Graph Spreading Activation and Centroid Based Summarization.

They describe briefly some unconventional approaches that, rather than aiming to build full summarization systems, investigate some details that underlie the summarization process, and that we conjecture to have a role to play in future research on this field. These methods are - Short Summaries, Sentence Compression and Sequential Document Representation.

Abstractive Summarization

Abstractive Summarization consists of understanding the source text by using linguistic method to interpret and examine the text. The abstractive summarization aims to produce a generalized summary, conveying in information in a concise way, and usually requires advanced language generation and compression techniques. There are several techniques to generate Abstractive Summary. They could be obtained from the extracted sentences (extractive summary) or generated using Natural Language Generation techniques.

Atif Khan, Naomie Salim [7] present a survey on abstractive text summarization methods in their paper. Abstractive summarization methods are classified into two categories i.e. structured based approach and semantic based approach. The main idea behind these methods has been discussed. Structured based techniques include - Tree Based, Template Based, Ontology Based, Rule Based. Semantic based techniques include - Multimodal Semantic, Information Item Based, Semantic Graph Based. Besides the main idea, the strengths and weaknesses of each method have also been highlighted. It is concluded from their literature studies that most of the abstractive summarization methods produces highly coherent, cohesive, information rich and less redundant summary.

Jackie CK Cheung [8], in his thesis presents a distinction between abstractive and extractive summarization. The abstractive summarizer is the Summarizer of Evaluative Arguments (SEA), adapted from GEA, a system for

generating evaluative text tailored to the user's preferences. While experimenting with the SEA summarizer, they noticed that the document structuring of SEA summaries, which is adapted from GEA and is based on guidelines from argumentation theory, sometimes sounded unnatural. They found that controversially rated UDF features (roughly balanced positive and negative evaluations) were treated as contrasts to those which were non-controversially rated (either mostly positive, or mostly negative evaluations). In SEA, contrast relations between features are realized by cue phrases, signaling contrast such as "however" and "although". These cue phrases appear to signal a contrast that is too strong for the relation between controversial and uncontroversial features. To solve this problem, they devise an alternative content structure for controversial corpora, in which all controversial features appear first, followed by all positively and negatively evaluated features.

Deep learning

Deep Learning has emerged as a new area of machine learning research. It tries to mimic the human brain, which is capable of processing and learning from the complex input data and solving different kinds of complicated tasks well. Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text.

PadmaPriya, G. and K. Duraiswamy [9] present an approach for multi document text summarization using **Restricted Boltzmann Machine**, a Deep Learning Technique. RBM is a graphical model for binary random variables. It consists of one layer of visible units (neurons) and one layer of hidden units. Units in each layer have no connections between them and are connected to all other units in other layer.

They first perform pre-processing on the documents using techniques like part of speech tagging, stemming and stop-word removal. For feature

computation they use four feature functions - title similarity, positional feature, term weight and concept feature. After obtaining a good set of feature vectors using RBM, they are further fine-tuned by using back propagation. Cross Entropy Error is used for adjusting the weight vectors. Two major problems faced while text summarization are ranking problem and selecting a subject of the top ranked sentences. They solve the ranking problem by finding the intersection between title and a particular sentence. They generate the sentence score for every sentence and these sentences are then arranged in the descending order based on the score. They obtain compression ratio as an input form the user. The no of top ranked sentences selected depends on this compression ratio.

The evaluation matrices they consider are recall, precision and f-measure. They carry out the experiment for three different document set from different knowledge domain. This algorithm is sensitive to the input data. The proposed algorithm has satisfactory results.

Support Vectors for Clustering

Support Vector Machines (SVMs) provide a powerful method for classification (supervised learning). Use of SVMs for clustering (unsupervised learning) is now being considered in a number of different ways. Currently known research focuses on the field of image processing and simple data clustering.

Ben-Hur, Horn, Siegelmann and Vapnik [10] present a novel clustering method using the approach of support vector machines. Data points are mapped by means of a Gaussian kernel to a high dimensional feature space, where they search for the minimal enclosing sphere. This sphere, when mapped back to data space, can separate into several components, each enclosing a separate cluster of points. They also present a simple algorithm for identifying these clusters. They further demonstrate the performance of

their algorithm on several datasets. They also elaborately put forth the mathematics behind the technique and also essential formulae.

They analyze the application of the algorithm by considering clustering with BSVs(Bounded Support Vectors) and without BSVs. They present examples suitably to demonstrate the same. Also they analyze the presence of overlapping and strongly overlapping clusters. They conclude the work, by stating that their algorithm has a distinct advantage over other, i.e. being based on a kernel method it avoids explicit calculations in the high-dimensional feature space, and hence is more efficient. Also a unique advantage of their algorithm is that it can generate cluster boundaries of arbitrary shape, whereas other algorithms that use a geometric representation are most often limited to hyper-ellipsoids.

Kees Jong, Elena Marchiori, Aad van der Vaart[11] introduce a heuristic method for non-parametric clustering that uses support vector classifiers for finding support vectors describing portions of clusters and uses a model selection criterion for joining these portions. Clustering is viewed as a two-class classification problem and a soft-margin support vector classifier is used for separating clusters from other points suitably sampled in the data space. The method is tested on five real life datasets, including microarray gene expression data and array-CGH data. They briefly outline the algorithm by enumerating different stages. The SVC technique is not used so far for sentence clustering or document clustering.

Lexical Chains

A Lexical Chain is a sequence of related words in the text. It spans short(adjacent words or sentences) or long distances(entire text). It is independent of the grammatical structure of the text. It captures the cohesive structure of the text. Lexical chains are heavily used is the resolution of an ambiguous term and identification of the concept that the

term represents. WordNet, part of speech tagging are mainly used in the identification of lexical chains. Since this results in the identification of the key section of the text, lexical chains can be used for Text Summarization. The summary generation using this technique relies on the model of topic progression in the text derived from lexical chains instead of its full semantic interpretation.

Regina Barzilay and Michael Elhadad[12] present a novel approach to perform text summarization using Lexical Chains. They present a new algorithm to compute lexical chains in a text, merging several knowledge resources like WordNet, parts of speech tagger and shallow parser for identification of nominal groups and a segmentation algorithm. They carry out four steps in order to obtain the summary - segmenting the original text, constructing lexical chains, identifying the strong lexical chains and extracting the significant sentences. They present empirical result on the identification of strong chains among the possible candidates produces their algorithm. They describe how to identify significant sentences using lexical chains. They also present the preliminary evaluation of the results obtained by their method. These results indicate the strong potential of lexical chains as a knowledge source for sentence extraction.

Key-phrase Extraction

Extracting keywords is one of the most important tasks when working with text. Readers benefit from keywords because they can judge more quickly whether the text is worth reading.

Alyona Medelyan[13] describes that a typical keyword extraction algorithm has three main components:

- Candidate selection: Here, we extract all possible words, phrases, terms or concepts (depending on the task) that can potentially be keywords.

- Properties calculation: For each candidate, we need to calculate properties that indicate that it may be a keyword. For example, a candidate appearing in the title of a book is a likely keyword.
- Scoring and selecting keywords: All candidates can be scored by either combining the properties into a formula, or using a machine learning technique to determine probability of a candidate being a keyword. A score or probability threshold, or a limit on the number of keywords is then used to select the final set of keywords.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin and Craig G. Nevill-Manning[l] describe the algorithm **KEA** (Keyword Extraction Algorithm) in their paper. Kea is an algorithm for automatically extracting keyphrases from text. Kea identifies candidate keyphrases using lexical methods, calculates feature values for each candidate, and uses a machine-learning algorithm to predict which candidates are good keyphrases. The machine learning scheme first builds a prediction model using training documents with known keyphrases, and then uses the model to find keyphrases in new documents. They use a large test data corpus to evaluate Kea's effectiveness in terms of how many author-assigned keyphrases are correctly identified.

G¨onen¸c Ercan [n] has tried to generate keyphrases by using lexical chains and machine learning algorithm - Naive Bayes Algorithm. In the thesis, he describes the feature functions that are used in the KEA algorithm. He has experimented with different feature functions that can be obtained from WordNet and lexical chains and suggested a few of them which yield best results. These are term frequency, first occurrence in text, last occurrence in text, semantic relation score, direct semantic relation score, lexical chain span, direct lexical chain span, hyponym and hypernym hierarchical levels, sentence count and direct sentence count.

Chapter - 3

Project Requirement Specification

3.1. Gantt Chart

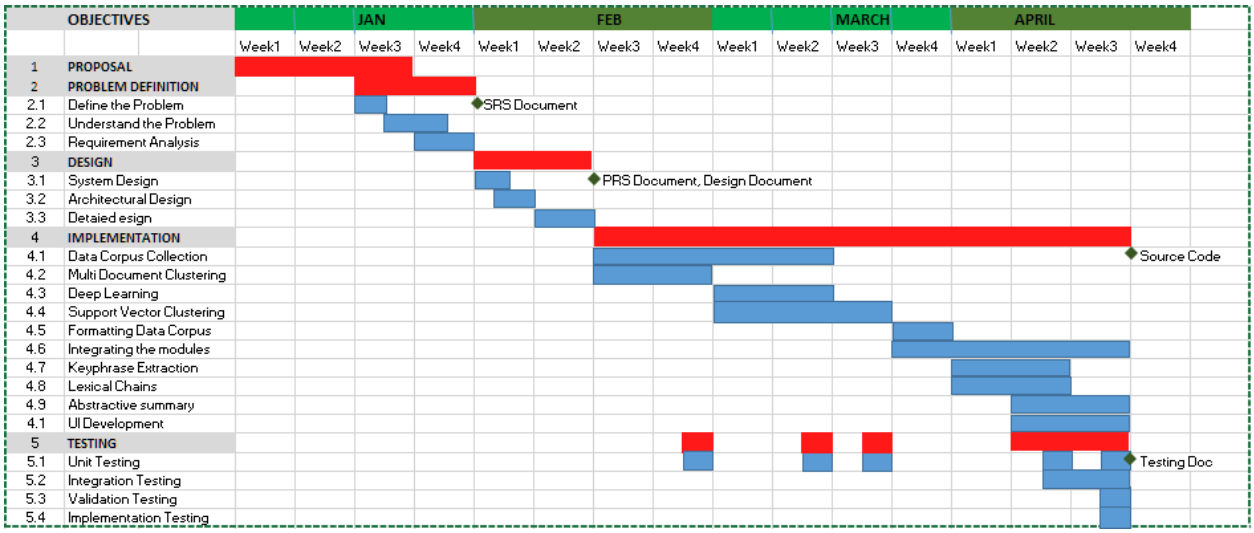


Fig 3.1 : Gantt Chart

3.2. Activity Diagram

Activity Diagram depicts each of the activity conducted in a project in stepwise activities and actions. The arrows shows the directions of how control moves in the project based on the activity that is being performed.

The figures given below are the activity diagrams for document clustering, Deep learning, Support Vectors for Clustering and Lexical Chains.

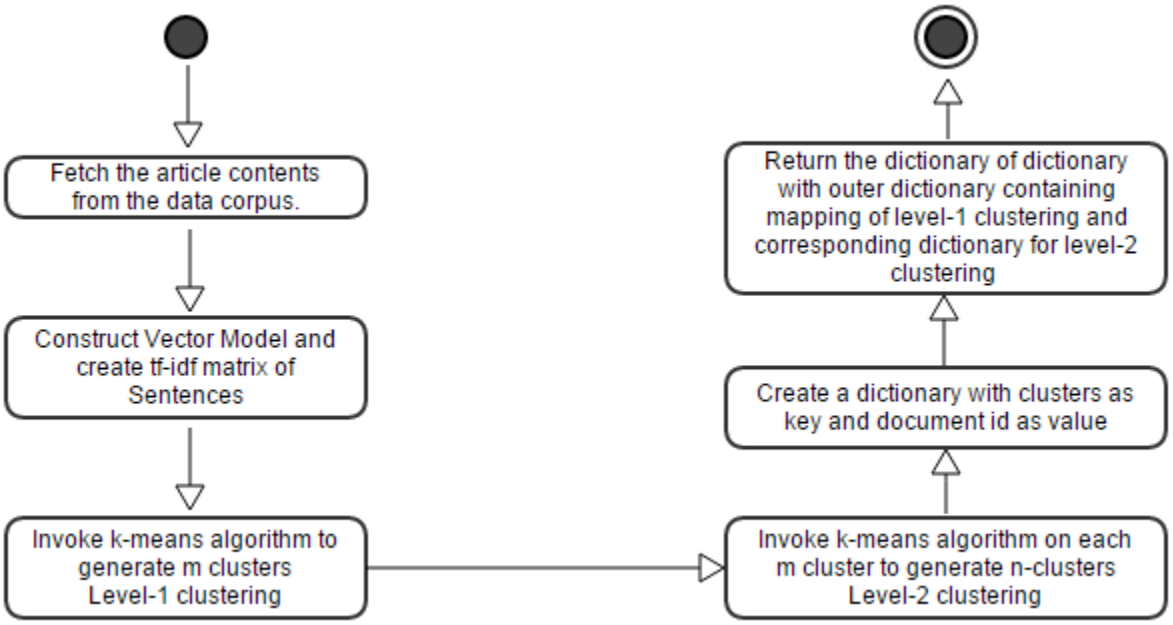


Fig 3.2 : Activity Diagram for Document Clustering

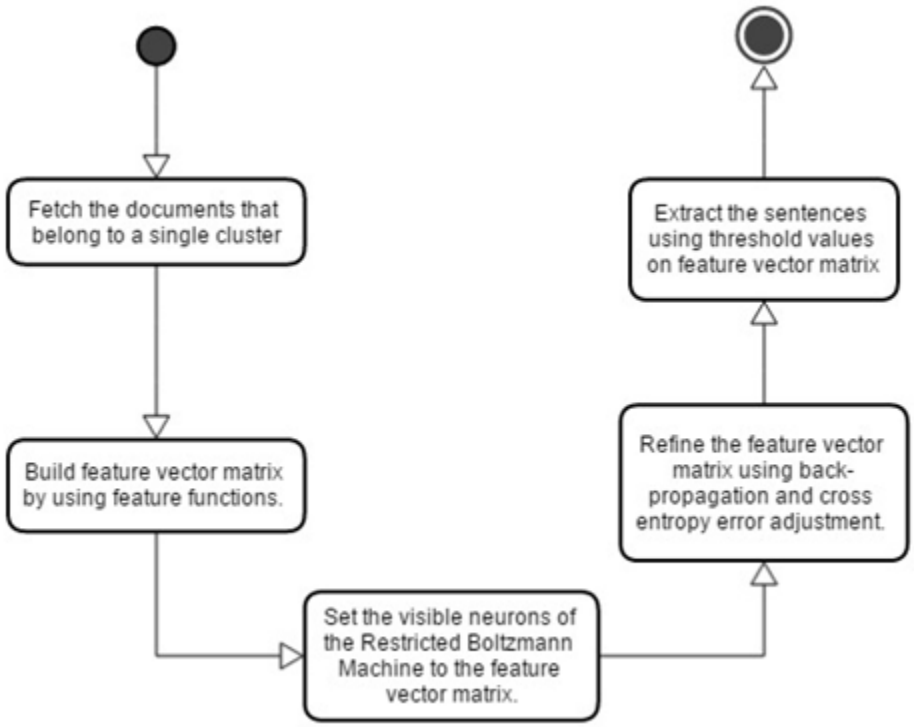


Fig 3.3 : Activity Diagram for Deep Learning

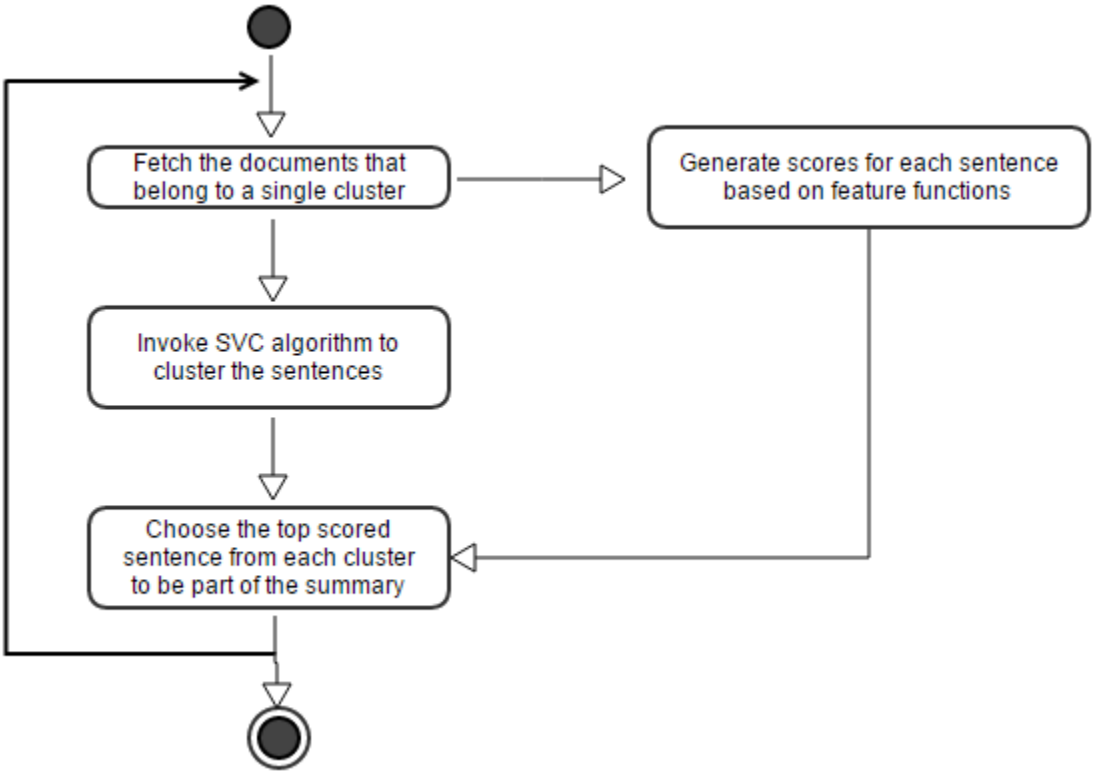


Fig 3.4 : Activity Diagram for SVC

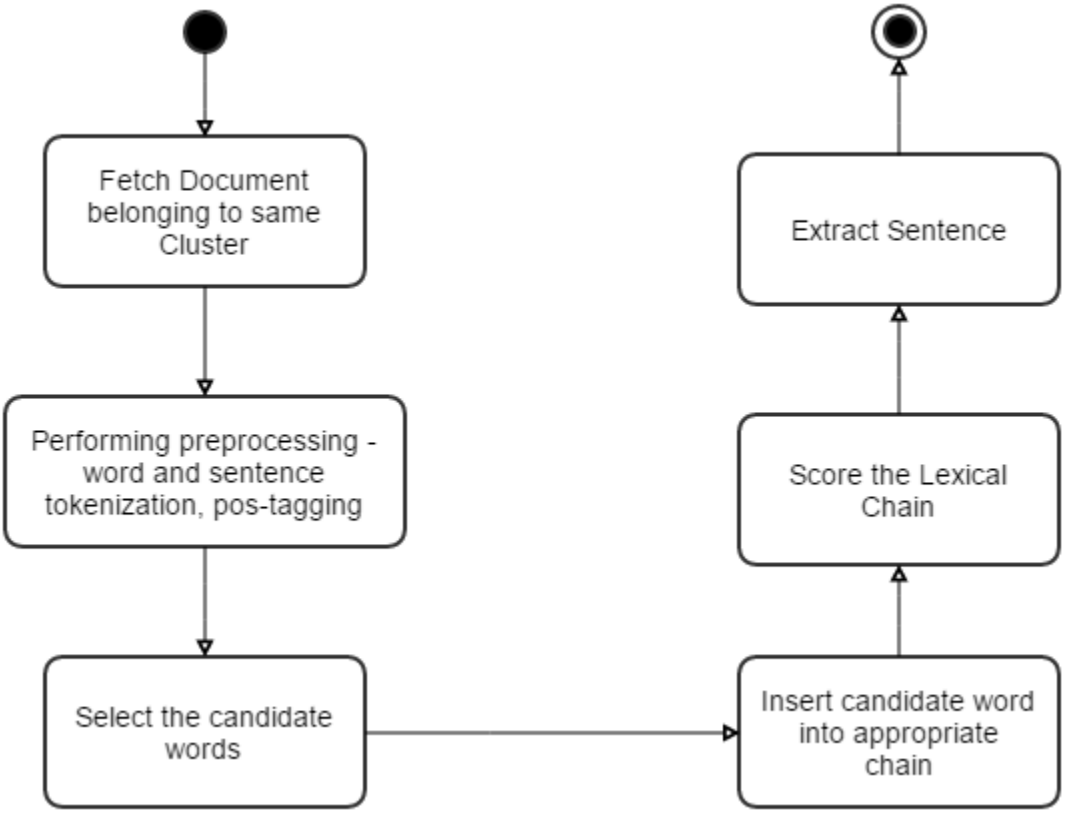


Fig 3.5 : Activity Diagram for Lexical Chains

3.3. Workflow Diagram

Workflow may be a view or representation of real work, thus serving as a virtual representation of actual work. The flow being described may refer to a document, service or product that is being transferred from one step to another.

3.3.1 Automated Modules

These are modules that were taken readily and were not implemented by the developing team. The workflow in such modules needs to be understood before making use of them. The automated modules that were used in the project were k-means, SVM from sci-kit library and RAKE python tool for keyphrase extraction. These modules are represented in a diagram below.

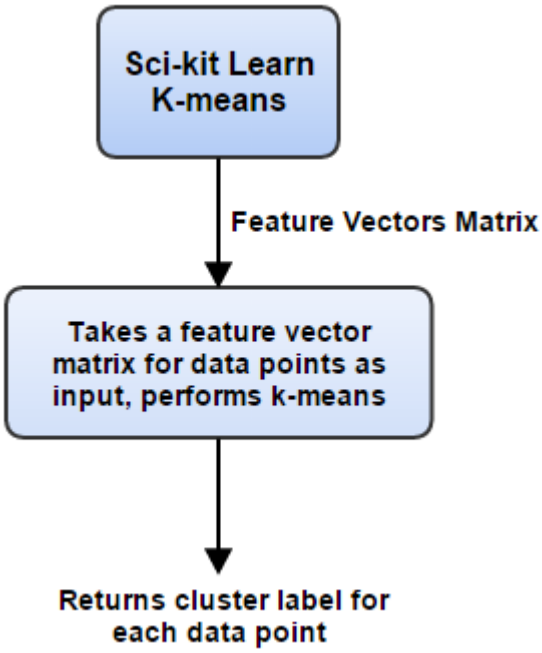


Fig 3.6 : Automated Module : Sci-kit Learn K-means

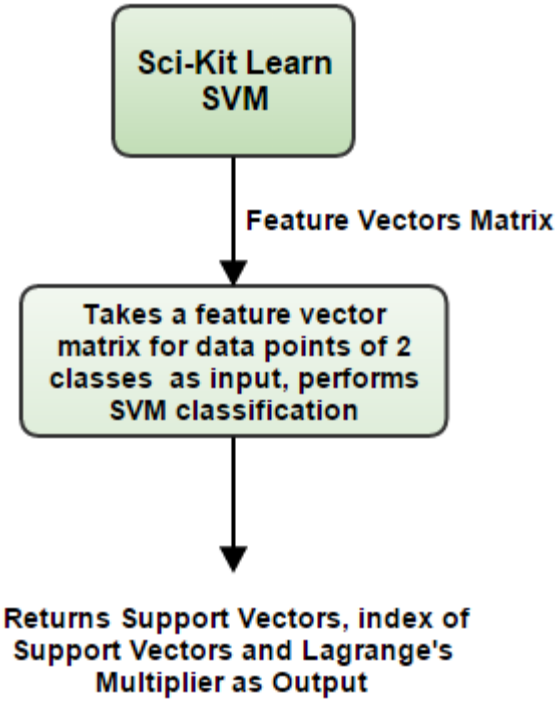


Fig 3.7 : Automated Module : Sci-kit Learn SVM

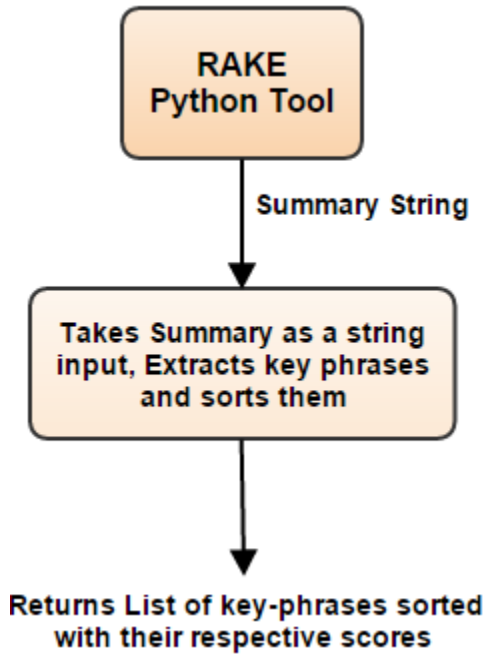


Fig 3.8 : Automated Module : RAKE tool

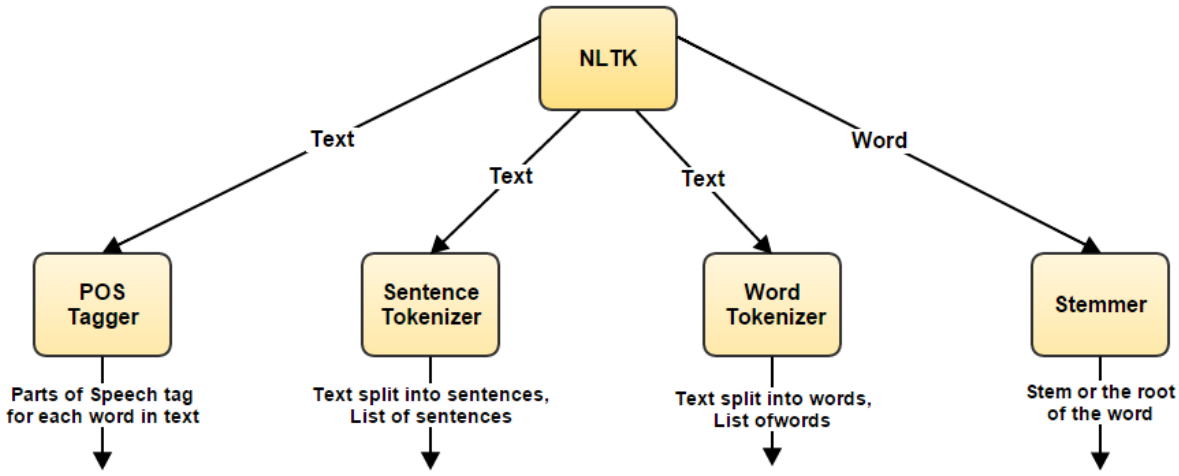


Fig 3.9 : Automated Module : NLTK

3.3.2 Manually Coded Modules

These are the modules that were implemented by the developing team of the project and they reflect the functionality of the product. The workflow diagrams for manually implemented modules are given below.

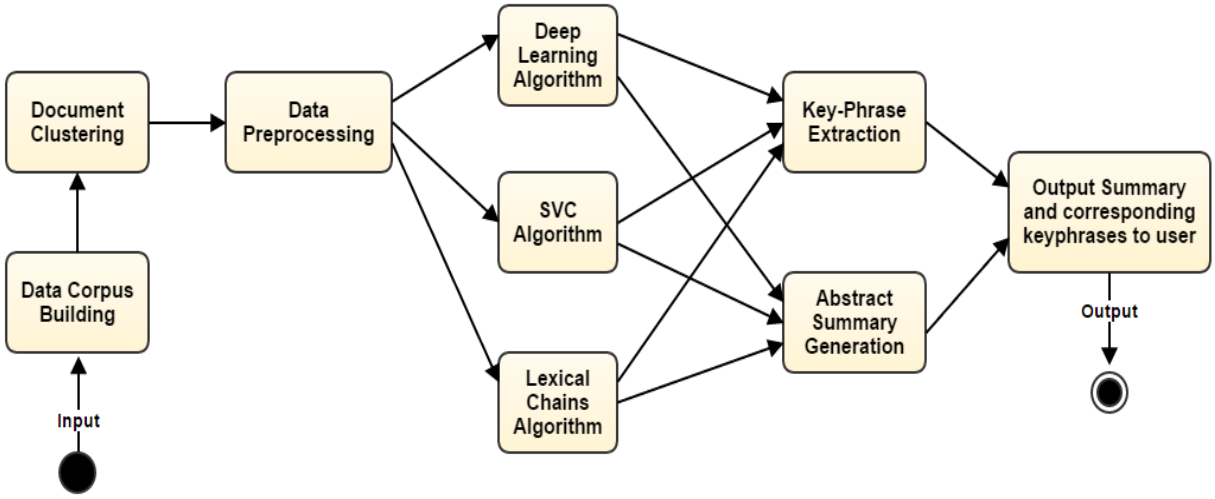


Fig 3.10 : Workflow Diagram for Manually Coded Manuals

The above diagram depicts the different manually coded manuals and the flow of process between them.

- Data Corpus Building module accepts input (Business News Articles) and builds a properly formatted json object.
- This object is then fed to Document Clustering module for grouping the similar documents into one.
- The data preprocessing module applies different pre-processing techniques and prepares data for further processing.
- Further, one of the three algorithms - deep learning, SVC or Lexical Chains can be invoked to perform the extraction of important sentences for each of the clusters obtained from applying document clustering on data corpus.
- These sentences are then passed as an input to keyphrase extraction module to generate important keyphrases which are one of the outputs of the system.
- Also the sentences are given to Abstractive Summary Generation module to construct the final summary to be output.

3.4. Constraints

3.4.1 What does the Project do

It is important to understand what the project does, the kind of inputs it works with and the output it produces. The following points outline the same.

- It produces abstractive summaries of the news articles.
- It collects and builds the data corpus across different business news rss feed.
- It produces the summaries after clustering the documents into clusters which contain new articles that discuss similar events.

- It can perform three different algorithms to achieve summarization - deep learning, SVC or lexical chain.
- It generates key-phrases for each of the summaries generated, which can be used by the user to know, if the topic of summary interests him.

3.4.2 What does the Project not do

It is equally important to know what the project cannot achieve and what should not be expected in the output of the system. The following points outline the same.

- It does not accept user input data corpus.
- It does not generate titles for the summaries generated.
- It does not summarize the image, graphs, videos or any other multimedia contents, which can be found in the news websites.
- It does not evaluate the summary using evaluation tools or algorithms. The evaluation is done manually based on predefined parameters.
- It does not let user view complete single news article, if he wishes to on portal. This can be viewed by looking in to json object if need be.

Chapter - 4

System Requirements Specification

The software and hardware requirements specify a complete set of prerequisites that are required to build the system. It describes the behavior of the system to be developed. The document includes the non-functional requirements for the software to be developed.
The assumptions are a set of constraints that determine the success of the project. Assumption lists various factors or system or environmental states that have to be fulfilled for success of the project .

4.1 Hardware Requirements

- Processor : Pentium Class PC (2.7 GHz or greater; faster processor or multiple processors recommended)
- Memory 4 GB of RAM or more recommended
- Operating system independent.

4.2 Software Requirements

- Python 2.6 or greater
- NumPy (>= 1.6)
- SciPy (>= 0.11)
- SciKit_Learn (>=0.15.2)
- NLTK (>= 3.0.1)
- Theano Library Latest Version

Note :- For all the above mentioned python libraries, the compatible version of it with respect to Python version are to be installed.

- Latest version of WAMP. We are using WAMP (2.5) with - Apache 2.4.9, PHP 5.5.12.

4.3 Non-Functional Requirements

- Time taken by the algorithm to display the output will vary depending on the size of the data corpus fed to the algorithm.
- It also depends on the algorithm used.
- The runtime of algorithm should not be drastically increased when the input data corpus size is increased.

4.4 Assumptions

- The input data corpus is a properly formatted json file containing the required field and absence of any information in it, is replaced by a suitable value.
- The feature functions and algorithm can handle the higher dimensional data, where the dimension of input increases with increase in data corpus. Hence, there is no need of dimensionality reduction algorithms.

4.5 User Requirements

This is being specified by the Use Case diagram. In use case diagram, the audience of the software and what are the operations they can perform will be depicted. Below is the Use Case diagram for Abstractive Multi-Document Text Summarization.

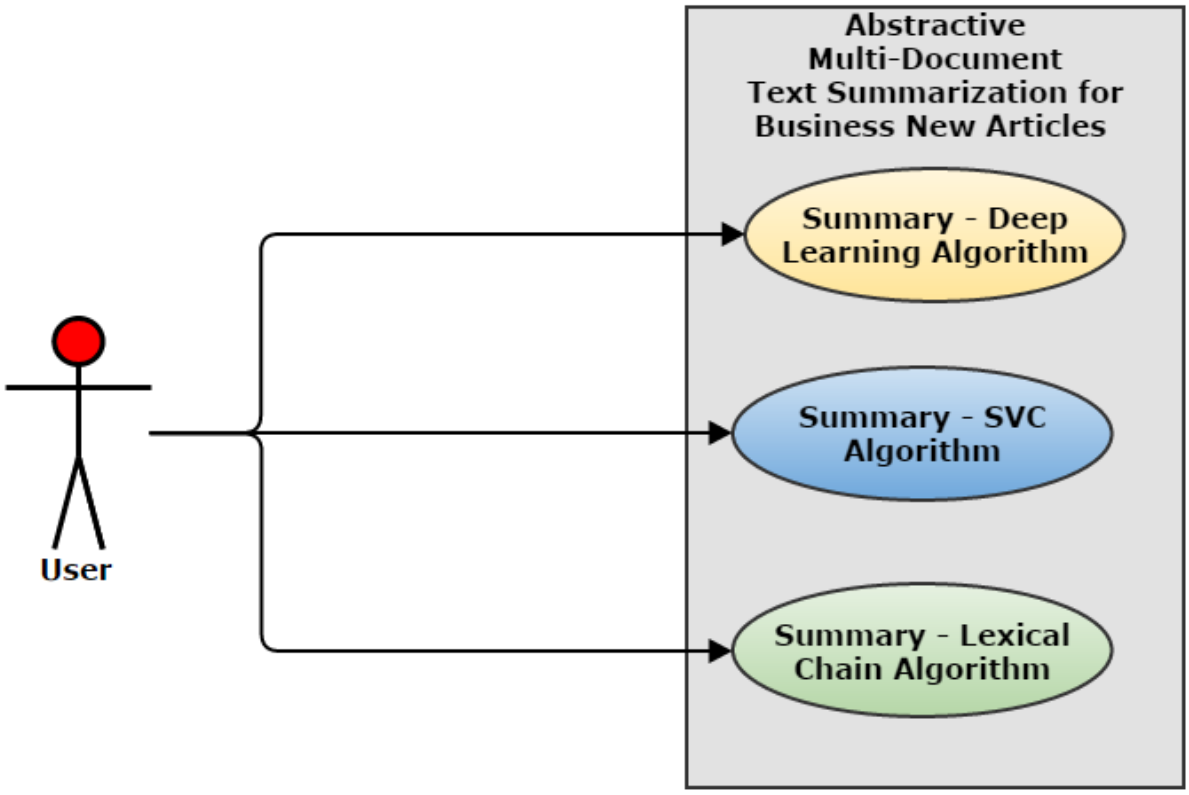


Fig 4.1 : Use Case Diagram for the System

As depicted in the figure above, the user can use one of the three options to view news summaries. The three options provided by the system are the view summary generated by three different algorithms -

- Deep Learning Algorithm
- Support Vectors for Clustering Algorithm
- Lexical Chains Algorithm

Each of the option calls the respective algorithm module at the back-end, performs the summarization task and displays the final summaries on the web portal interface, which forms the output of the system.

Chapter - 5

System Design

This section of the document describes the high level design aspects of the project and also covers the low level description of the underlying components and their design structure. Topics covered include the following:

- System Overview
- Implemented Modules
- Data Design

5.1. Block Diagram

Block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. Below is the block diagram for Abstractive Multi-document Summarization project.

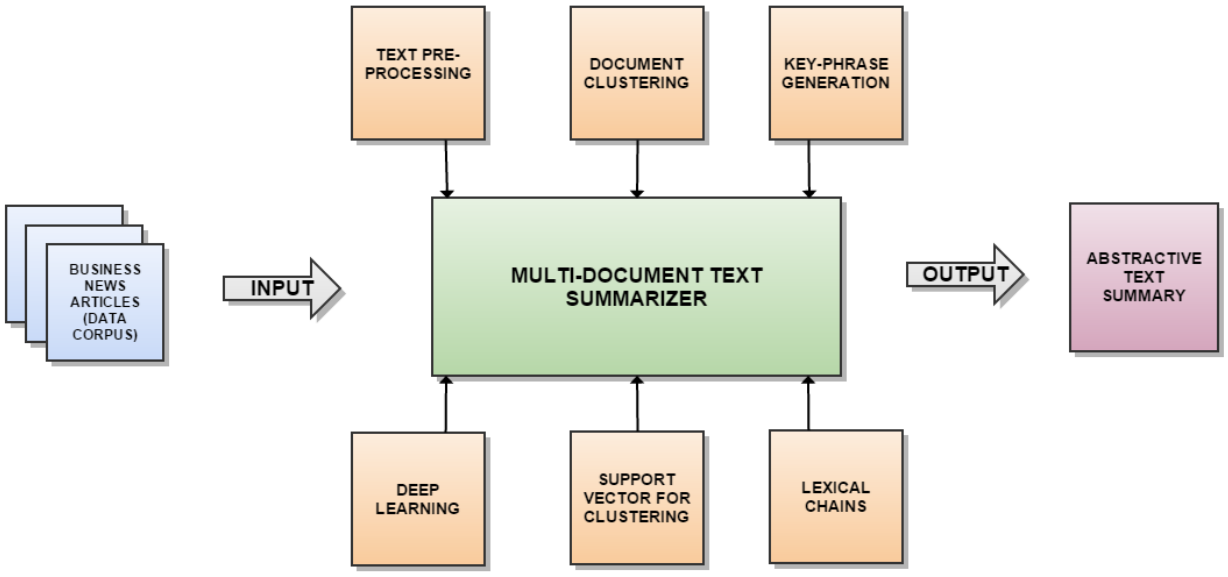


Fig 5.1 : Block Diagram

5.1.1 System Overview

- **Functionality** : The Multi-Document Text Summarizer system, takes in an input of multiple Business news articles and tries to generate an abstractive summary out of them. Document clustering is performed to create clusters of similar documents. It uses different algorithms - deep learning, svc and lexical chains to extract important sentences and key-phrase generator to generate significant phrases which will serve as tags.
- **Context** : The system generates the summary using the algorithm chosen by the user. There are 3 options for algorithm to be chosen. It runs the algorithm on the pre-collected data corpus (business news articles) and outputs the summary to user. The different components with which the system interacts is depicted in the block diagram.
- **Design** : The design of the system is modular. Each module performs a definite task. The interactions between these modules are simple and easy to understand. The design of the system is chosen to ensure maximum modularity and at the same time remain effective and perform desired tasks efficiently.

5.2. Implemented Modules

- **Data Corpus Collection :**

  The module crawls through the rss feeds list and fetches news articles. It formats the contents received, which is required for further processing and return a well formatted json for data corpus.

- **Document Clustering :**

  The module takes in an input of data corpus as json object and performs k-means clustering at two levels. First level happens on all the news articles which generate m clusters. Second level happens on

each of the m clusters to generate n clusters within. The output is the cluster labels and document ids which belong to that cluster.

- **Deep Learning :**

  The module trains the Restricted Boltzmann Machine (RBM), updates the cost and modal parameters. It uses contrastive divergence to train the RBM. Gibbs sampling is performed and the end of Gibb's sample gives the optimal feature vector set. Those sentences whose optimal feature vector set satisfies the predefined threshold values, are selected.

- **Support Vector Clustering :**

  The module performs clustering of sentences, for each document cluster, using support vectors (SVC algorithm) and generates clusters of sentences. A sentence scoring algorithm is then applied to obtain top scoring sentences from each cluster. The top scored sentences are returned which will form the summary of the respective document cluster.

- **Lexical Chains :**

  This module generates lexical chains by considering the candidate words in the input data. These lexical chains are scored using an algorithm. For each chain in the summary representation, the sentence that contains the first appearance of a representative chain member in the text is chosen.

5.3. Data Design

The data flow diagram(DFD) capture the flow of data through the entire system. The following is a Level-0 diagram which gives an overview of data flow through the system, which is composed of three significant modules.

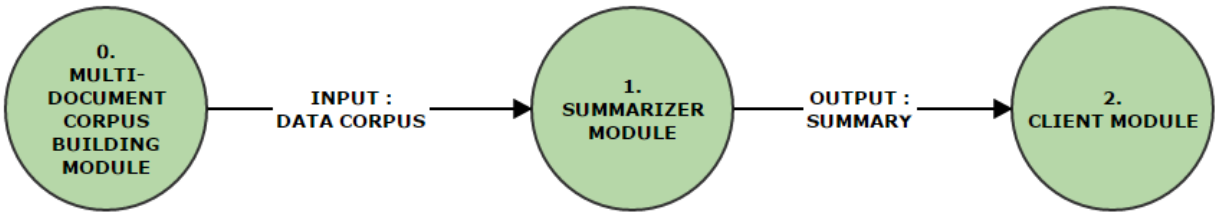


Fig 5.2 : DFD Level-0

The Multi-Document Corpus Building module builds the data corpus. It involves crawling through the links given by rss feed, collecting the news articles and creating a proper formatted json file of data which is one of the output of this module. It also performs document clustering and returns an output of dictionary of cluster labels and document ids.

The Summarizer Module takes the json object as input, performs different tasks in stages - preprocessing, extracting important sentences, creating abstractive summary. The extracting of important sentences can be performed by any of the three algorithms - deep learning, SVC or lexical chains, which would be an user choice. The summary is the output of the module.

The Client module takes the summaries as input, generate keyphrases for each the summaries. The summaries and corresponding keyphrases are displayed to user through a web-portal.

Chapter - 6

Detailed Design

The detailed design consists of low level data flow diagrams. Each module is further split into its low granular modules and data processing at each level is described below.

6.1. Data Flow Diagram - Level 1

In this section, we present the data flow diagram at level - 1 for each of the modules depicted in data flow diagram at level - 0, provide details about input(s) and output(s) to the module and give a brief description of the task accomplished by the module.

6.1.1 Module Name : Multi-document Corpus Building

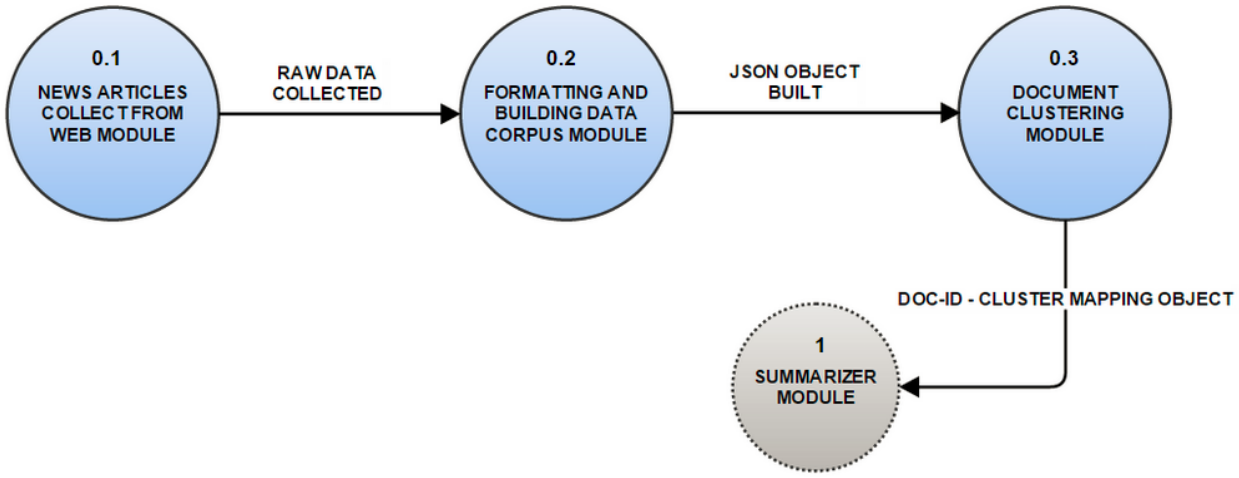


Fig 6.1 : DFD - Multi-document Corpus Building Module

Input : RSS feed links of different newspapers and websites.

Description : The module accomplishes two major functionalities - building the data corpus and clustering the documents. A list of predefined RSS feed links are given to a php program, which crawls through these feed one by one on a periodic basis and retrieves the articles. Along with these articles,

other relevant fields that are also collected are title of the article, publishing date and publisher name. All these information and a unique document id associated with each document is stored in the form of a json object, which forms one output of the module.

This json building is done by a python module which reads the data collected, removes duplicate articles based on time-stamp (publishing date) and selects unique articles. The second task is that of grouping together the documents based on similarity measures, so that we can generate summary for each group. This is the sole functionality of Document Clustering Module. It performs a two level k-means clustering on the data obtained in previous step. The no. of clusters to be obtained at each level is an input parameter to be passed or configured. It further creates a dictionary, which maps the cluster labels to the unique id of documents which form the cluster.

Output : JSON Object Representation of Data, Document to Cluster mapping dictionary.

6.1.2 Module Name : Summarizer Module

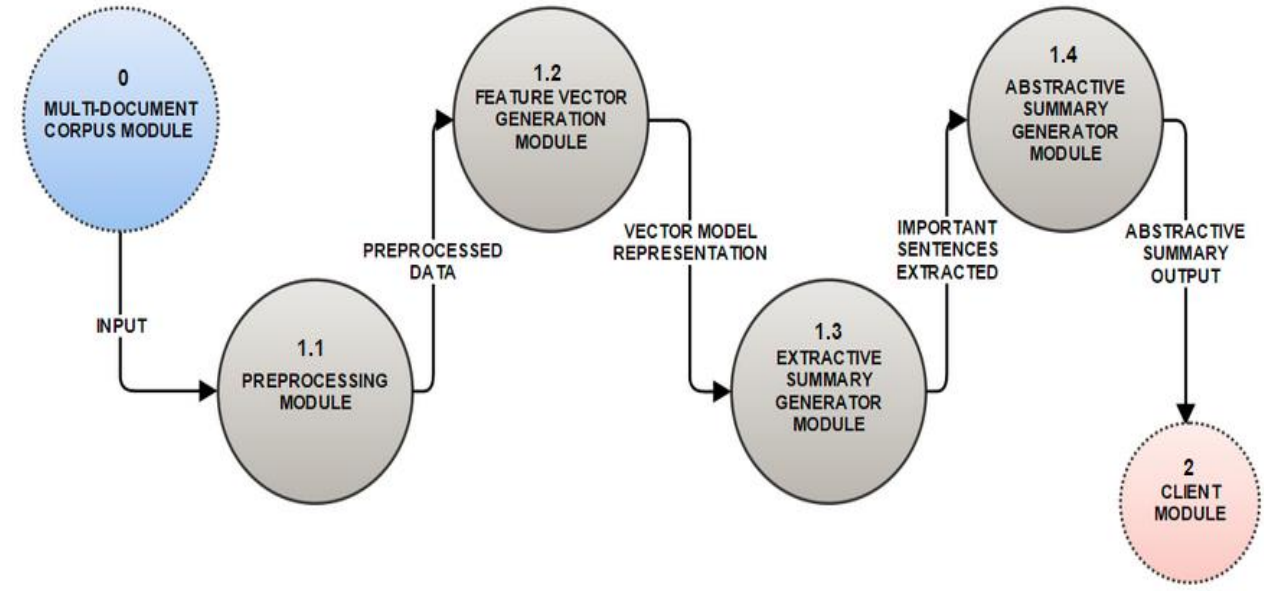


Fig 6.2 : DFD - Summarizer Module

Input : JSON Object Representation of Data, Document to Cluster mapping dictionary.

Description: This is the core module of the entire system. It accepts the input from the previous module - Multi-Document Corpus Module, which is json representation of data and document to cluster mapping. It first performs preprocessing on the data which involve - sentence tokenization, word-tokenization, stemming, punctuation removal, pos-tagging and others which are dependent on the algorithm applied to generate the summary. The next stage is feature-vector generation in which feature vector representation of data is generated based on the requirement of algorithm to generate summary. Some of the common feature functions are - sentence length, position of sentence, title-sentence similarity, term frequency, similarity of sentences to the first sentence in the article etc.

In the next stage, the feature vectors and document-cluster mapping is input to the Extractive Summary Generation module for summary generation. This module calls one of three algorithms (based on user's choice) and generates an extractive summary for each of the cluster by considering all the documents in that cluster. The summary generated is passed to Abstractive Summary generation module to generate abstractive summary. The final summary generated from this module for each cluster forms the output of this module.

Output : Summaries for different clusters

6.1.3 Module Name : Client Module

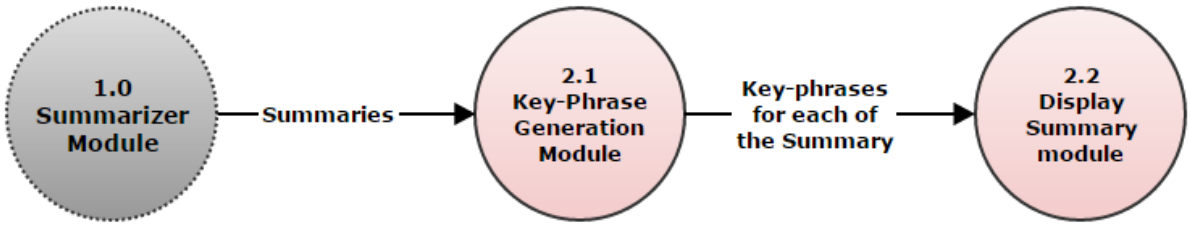


Fig 6.3 : DFD - Client Module

Input : The summaries generated

Description: The client module takes as an input the summaries generated by Summarizer module. It invokes the key-phrase generation module, which generates top four key-phrases for the summaries. The summaries and their corresponding key-phrases are then given to Display Summary module

Output : The summaries along with the keyphrase shown on an user interface.

Chapter - 7

Implementation

In this section, we present the pseudo code for the main modules involved in developing the project.

Pseudo code for Data - Corpus collection and building.

//Module : rss.php

function createDataset(newsWebsiteURL)

```
for each newsWebsiteURL:
    rss = load the rss feeds
    data = [] //empty array
    for each item in rss:
        title = item -> title
        link = item-> link
        description = item->description
        article content = file_get_contents ( description)
        date = item->pubdate // The date is converted to IST if it
         is in GMT
        array_push($data, array("id" => id, "title" => title,
        "content" => obj->content, "pubdate"=> pubdate,
        "newspaper" => newspaper))
write_to_json_file with a name `results<k>.json`, where k >=1
```

//Module : BuildDataCorpus.py

function buildJson(self):

```
os.chdir(self.input_file_dir)
files = filter(path.isfile, os.listdir("."))
json_array = []
index = 0
article_id = 0
for json_file in files:
    data = json.loads(open(json_file).read())['root']
    file_date = time.ctime(os.path.getmtime(json_file))
    for news in data:
        dt = news["pubdate"]
        t1 = datetime.strptime(dt, "%a, %d %b %Y %H:%M:%S
        +0530")
        if index == 0 or t1 >= t2:
            article_id +=1
            news["id"] = article_id
            json_array.append(news)
    t2 = datetime.strptime(file_date,"%a %b %d %H:%M:%S %Y")
    index += 1
data = {}
data["root"] = json_array
os.chdir(self.output_file_dir)
```

```
        with open(self.corpus_filename, 'w') as outfile:
        json.dump(data, outfile)
```

Pseudo code for Document Clustering

//Module : docClustering.py

```
function DocPreprocess(articles):
    processedArticles = [] //empty list
    for each article in articles :
        processed_article : article.lower().remove_punctuations
        processedArticles.append(processed_article)
    return processedArticles


function stem_tokens(tokens, stemmer):
    return stemmed token for each token in tokens using stemmer


function tokenize(text):
    tokens = nltk.word_tokenize(text)
    stems = stem_tokens(tokens,stemmer)
    return stems
```

function tfidfCompute(text):

```
tfidf = TfidfVectorizer(tokenizer=tokenize, stop_words='english')

tfs = tfidf.fit_transform(text)

return (tfs, tfs.shape)
```

function kmeansCluster_docs(text, n_clusters=1):

```
text = docPreprocess(text)

tfs_tuple = tfidfCompute(text)

//based on the length of text, pass the correct number of clusters

if len(text) > n_clusters :

    //call KMeans function from sci-kit learn library to perform  k-means clustering.

    km = KMeans(n_clusters, init='k-means++', max_iter=100, n_init=1, verbose=False)

else:

    km = KMeans(n_clusters=len(text), init='k-means++', max_iter=100, n_init=1, verbose=False)

endif

km.fit(tfs_tuple[0])

return km.labels_
```

//main function of the module

function docClus (json_file):

articles = json.load(json_data)["root"]

contents_dict = create a dictionary with id as key and content of article as the value

//clusterings at level 0

tfs_tuple = kmeansCluster_docs(contents_dict.values(), n_clusters=5)

top_clustered : create dictionary of cluster to document mapping by looping through tfs_tuple

//clustering at level 1

docsClustered : dictionary of dictionaries of list

for cluster in top_clustered.keys():

similarDocs : create a dictionary similar documents for cluster

tfs_tuple_doc = kmeansCluster_docs(similarDocs,n_clusters=3)

docsClustered : create a dictionary of indices to cluster mapping

//Return the mapping

return docClustered

Pseudo code for Deep Learning

//Module : MyRBM.py

function preProcessing(articles) :

perform sentence and word segmentation, remove stop words, obtain root word, perform pos_tagging

function buildFeatureMatrix(data) :

for each article in the input data :

for each sentence in an article :

evaluate the feature function for each sentence

function summarize(json_file, clusterings) :

Load the json_file

generateSummaryForClusters (json_file, clusterings)

function generateSummaryForClusters(json_file, clusterings) :

for each cluster in the set of clusteres:

obtain the article ids

generateSummary(data, article_ids)

function genereateSummary(data, article_ids) :

data = preProcessing(data, article_ids)

feature_vecctor_matrix = buildFeatureMatrix(newsData, wmap, wcount, all_sentences, func_obj)

// The sentence matrix is written to a pickle file.

results = test_rbm(pickled_file)

for feature_vector in results :

Check if the feature_vector satisfies the minimum theshold values.

if yes : extract the sentence.

else : skip it.

return extracted_sentences

function test_rbm(pickled_file):

initialise the train_set

rbm = rbm() // initializes no of hidden neurons. no of visible neurons and input

cost, updates = rbm.getCostUpdates(learning_rate,no_of_gibbs_steps)

train_rbm(cost, updates, train_set) // RBM

// Converge the Markov chain using Gibbs sampling (k-steps of gigs_h_given_v)

sample_func() // considers the end of Markov chain

function getCostUpdates(learning_rate, no_of_gibbs_steps):

This functions implements one step of Contrastive Divergence sample_h_given_v(input)

gibbs_hvh() // perform k-Gibbs steps

Returns a proxy for the cost and the updates dictionary.

(The dictionary contains the update rules for weights and biases.)

function propUp():

P(h|v) = sigm(Wi . v + Ci)

function sample_h_given_v(v):

propUp(v)

function gibbs_hvh():

sample_h_given_v

sample_v_given_h

Pseudo code for Support Vector for Clustering

//Module : svcClustering.py

function readCorpus(json_file):

json_data = json.load(json_file)[“root”]

content_dict : create a dictionary with id as key and content of article

as value

global content_dict

function clusSentenceGen(clusterings):

for each article in clusterings :

tokenize the article in cluster to sentences

create feature vectors list by invoking featurefunction.py module with inputs as sentences of articles

compute score of each sentence, create a list of it

return (sentences, sentence_scores)

function generateClassB(classA, rows, columns):

generate points randomly for class B , size =size(class A) and they lie in same hyperplane consisting class A points

return ClassB

function computeCosine(sent_tfidf, R, R_indices):

for each index in R_indices :

distances.append(cosineSimilarity(sent_tfidf, index))

max_dist = max(distances)

return index corresponding to max_dist

function svClustering(clusSentList, feature_vectors):

preprocessed_list = docClus2.docPreprocess(clusSentList) #preprocess sentences

tfs_tuple = docClus2.tfidfCompute(preprocessed_list)

classA : create by itering through tfs_tuple

classB = generateClassB(classA, rows(classA), columns(classA))

input_list : create a list with classA and classB points appended and labelled with class name

//invoke svc (support vector classification) function of sci-kit library

clf = svm.SVC(C=C_0, gamma=sigma, kernel='rbf')

clf.fit(input_X,input_labels)

non_bounded_support_vectors_A : obtain lagrange's coefficent from clf object and compute the non-bounded vectors for class A

R : non_bounded_support_vectors_A

R_indices : indices of non_bounded_support_vectors_A

unclustered_indices = create a list of unclustered indices

//assign the unclustered points to their nearest cluster

for each unclus_index in unclustered_indices :

cluster = computeCosine(tf_idf of sentence of index

unclus_index, R, R_indices)

clusSentIndices : create dictionary with key of cluster and value as unclus_index

clusSentList : create dictionary with key of cluster and value as as corresponding sentence.

summary=generateSummary(clusSentIndices, feature_vectors , clusSentList)

return summary

//main function of the module

function svc (data, clusterings):

readCorpus(data)

summaries = [] //empty list

keys1 = clusterings.keys()

for each i in keys1: //level 1 clusters

keys2 : clusterings[i].keys()

for each j in keys2: //level 2 clusters

```
            cluster : clusterings[i][j]
            (clusSentList, feature_vectors) =
            clusteredSentenceGen(cluster)
        summaryi = svClustering(clusSentList, feature_vectors)
        summaries.append(summaryi)
    return summaries
```

//Module feature_function.py

function length_score(clusSentList):

compute and return the length score for all sentences in clusSentList

function position_score(sentence_tokenize_list):

compute and return the position score for the sentence in sentence_tokenize_list

function title_similarity_score(title, sentence_tokenize_list):

compute and return the title similarity score for each sentence in sentence_tokenize_list

function first_sentence_score(sentence_tokenize_list):

compute and return the similarity with first sentence score for each sentence in sentence_tokenize_list

//Module extSummary.py

function computeScore(feature_vectors, clusSentIndices):

compute and return a list of total feature function score of sentence for each sentence with index in clusSentenceIndex

function generateSummary(clusSentIndices, feature_vectors, clusSentList, summary_length=6):

scores = computeScore(feature_vectors, clusSentIndices):

summary = “” //empty string

while i less than summary_length:

summary += choose top scoring sentence from each cluster in round robin fashion

return summary

Pseudo Code for Lexical Chains

//Module extSummary.py

function summarize(json_file, clusterings) :

Load the json_file

summaries = generateSummaryForClusters (json_file, clusterings)

return summaries

function generateSummaryForClusters(json_file, clusterings) :

for each cluster in the set of clusteres:

obtain the article ids

summary = generateSummary(data, article_ids)

summaries.append(summary)

return summaries

function genereateSummary(data, article_ids) :

data = preProcessing(data, article_ids) // Concatenates all the articles belong to the cluster and returns it

l = LexicalChain()

lexical_chains = l.Processing(data)

sentence_ids = getSentenceId(lexical_chains)

content = extractSentence(data, sentence_ids)

return content

function extractSentence(data, sentence_ids) :

```
data = data.replace("-\n","")
data = sent_tokenize(data)
id_list = list(sentence_ids)
extracted_sentences = []
for id in id_list:
    extracted_sentences.append(data[id])
return extracted_sentences
```

function preProcessing(input) :

```
input = input.replace("-\n","")
input = sent_tokenize(input)
input = [[pos_tag(word_tokenize(sent)) for sent in input]]
mc = GalleyMcKeownChainer(data=input) // Creates the Lexical
Graph. This graph contains LexNodes. The LexNodes that belong to the
same lexical chain, have links between.
chains = mc.computeChains()
return chains
```

function computeChains(input) :

```
// Reduce the LexGraphs
for each chain :
    score = compute_score(chain)
```

```
            scoredChains.append((chain, score))
        return scoredChains
```

Pseudo Code for Abstractive Summary Generation

//Module : ReplacePronoun.py

```
read corpus_file
get pos_tags and ner_tags for words
for each pos_tag_tuple in pos_tags:
        if pos_tag_tuple[“pos”] equals “pronoun”:
                replace with the latest person/organisation name
        else if pos_tag_tuple[“word”] in (company, organisation):
                if prev_word is “this” or “that”:
                        replace the previous word with person/organisation
return processed_data_corpus
```

//Module : absSummary.py

```
read summary_json file
for each summary in summaries_list:
        sentence tokenize the summary
        for each sentence in summary:
```

```
            for each word in summary:
                if word in notwords:
                    mark word for deletion
            delete marked words
        remove unwanted punctuations and spaces in the beginning of the
        sentences
write the updated json output to file
```

Chapter - 8

Testing

The purpose of this chapter is to enable the project management to know the testing status of the project and the quality level of the application. It provides an insight on the various test cases and the scope of activity that has led to the development of this document.

Testing process was divided into unit testing and integration testing.
Unit testing phase was used to perform the test cases on individual units of the application. After this, the entire application was tested to verify that the individual units of the application work in precision when run together.
The unit testing involved the checking of each unit namely, the front end-GUI, the navigation, content of the web pages were verified. In the testing of the control logic, compilation, exception handling and other errors were checked for. Testing was also performed to verify if the database entries are correct as per the last day of modification of the database.
An integration testing was performed once the unit testing was completed. The application was tested for the three algorithm options given to users - "deep learning", "svc" and "lexical chains". The system also was tested by varying the feature functions.

8.1 Assumptions

- Sufficient data is available - the size of data corpus is reasonable enough to hold the results of test cases run to be valid.
- When the size of data-corpus is increased, the test case would still pass. That is the result of test case is independent of size of data corpus.

8.2 Constraints

- Needs a data-corpus of reasonable size.
- Requires the input data to be well formatted.
- Proper/Valid filenames and folder names should be given as input. Also they should have the required permission to access it - read, write and modify.
- The required automated and manually coded modules should be imported and be available on the system where testing is performed.

8.3 Risks

- Schedule Risk: Project schedule gets slipped when project tasks and schedule release risks are not addressed properly. They can be due to wrong time estimation, resources are not tracked properly, failure to identify complex functionalities and time required to develop those functionalities.

- Operational Risks: Risks of loss due to improper process implementation or failed system. This can be caused due to failure to resolve responsibilities, no resource planning or no communication in the team.

- Technical risks: Technical risks lead to failure of functionality and performance of the web-app. This may be due to the complexity in the project implementation.

8.4 Summary of Assessment

The following is a summary of the test case results obtained for the reported test effort. Refer to subordinate sections of this document for detailed results and explanations of any reported variances.

Table 8. 1: Test Case Summary Results

Summary Assessment	Total Number Test Cases	% of Total Planned
Test Cases Planned	10	100%
Test Cases Run	9	90%
Test Cases Reviewed	9	90%
Test Cases Passed	8	80%
Test Cases Failed	1	10%
Test Cases To Be Run	1	10%

The following is a summary of the test incidents (i.e., unexpected results, problems, and/or defects) that were reported during the testing:

Table 8. 2: Test Incident Summary Results

Impact/ Severity Level	Total Reported	Total # Resolved	% Total Resolved	otal # Un-Resolved	% Total Unresolved
High	1	1	100%	0	0%
Moderate	4	2	50%	2	50%
Low	2	1	50%	1	50%
Combined Totals	7	4	58%	3	42%

8.5 Detailed Test Results

The table below summarizes the test cases employed for various test categories and the test results obtained for each test case :

Table 8. 3: Summary of Types of Test

Test Case/Script ID	Test Case/Script Description	Pass/ Fail	Comments
Validation Testing	Testing the correctness provided by the software.	Pass	The summary generated by the Automatic Summarizer for different

			combination of inputs (three algorithms and the data corpus) were tested and verified
Implementation testing	Testing the implementation details of the software	pass	

Table 8. 4: Validation Test Table

Test Case	Test Case Performed	Output	Pass/Fail	Comments
Validation testing for Deep Learning	Summary generation for the algorithm with parameters - No of feature functions = 8 No of iterations of one Gibbs	Average length of generated summary is 6	Pass	Desired output was obtained.

	Step= 50 Fixed set of threshold values			
Validation testing for SVC	Summary generation for the algorithm with parameters :- RBF Kernel - c=8 and sigma = 0.1 sentence score parameters (0.4, 0.3, 0.2, 0.1)	Summaries with average length of five was generated	Pass	Desired output was obtained.
Validation testing for Lexical Chain	Summary generation for the algorithm with parameters : relation type : , sentence	Average length of generated summary is 6	Pass	Desired output was obtained.

	distance and paragraph distance			

Table 8. 5: Implementation Test Table

Test Case	Test Case Performed	Output	Pass/Fail	Comments
Implementation testing for the overall correctness of the software	Summary output for the particular summary algorithm that was chosen, clearing the already generated summaries from div when new option was chosen	The summaries and tags were displayed in proper place, the old output data cleared when new option was chosen	Pass	Works as expected
Implementation testing for the	Passing of data	Indicated the	Pass	The modules have been

overall correctness of the software	different modules and error handling with erroneous data was tested	presence of erroneous data and the data is passed between different modules as expected		integrated properly and proper error handling messages are displayed

8.6 TEST INCIDENTS

The following were the unexpected results and defects that occurred during testing:

- The data collected is in Unicode format. Hence certain input/output operations and data processing could not be performed and errors were generated.
- The number of clusters to be formed during document clustering is pre-defined. Hence when sufficient no of documents were not available in a cluster after level-0, exceptions would be thrown.
- After the removal of stressed words in abstract summary generation module, unwanted spaces and punctuations would occur in the beginning of the sentence.

Resolved Test Incidents

In the above mentioned test incidents 1 and 2 were resolved.

Partially Resolved Test Incidents

In the above mentioned test incident 3 was partially resolved.

Chapter - 9
Results and Discussion

The "Abstractive Multi-Document Text Summarization" is an Automatic Summarizer system. It generated abstractive summaries for the input Business News Articles Data Corpus.

The data corpus module collected about 800 articles over a period of 10 days. The collected articles were combined and well formatted json representation was created. This was passed as an input to document clustering module. The document clustering was performed with 13 clusters at level-1 and 12 clusters at level-2 and total of 156 clusters were generated.

Deep Learning

This algorithm generates summary for 20 clusters. 6 feature functions were used. The feature functions used decide the quality of the summary generated.

Support Vectors For Clustering

The algorithm generates summary for 40 clusters. The RBF kernel parameters for SVM classification was c=8 and sigma=0.1. These parameters decide the number of clusters that will be generated by SVC and this varies from one execution to another. The summaries are generated were on an average of length 6.

Lexical Chain

The algorithm computes lexical chains. It performs scoring of the chains formed, based on the parameters – sentence distance, paragraph distance and relation type. The sentence that contains the first appearance of a representative chain member in the text is chosen.

We tried to run the above algorithms to generate summaries for the complete dataset. But due to memory constraints of the system on which they were executed, the system could not generate summaries for such huge data.

The above generated summaries were given as an input to key-phrase extraction tool. It generates top four key-phrases for each of the summary. The summaries and their corresponding key-phrases were given as an input to client module which creates a web view and displays them to the user.

Chapter - 10

Snapshots

Screenshot of Document Clustering Output

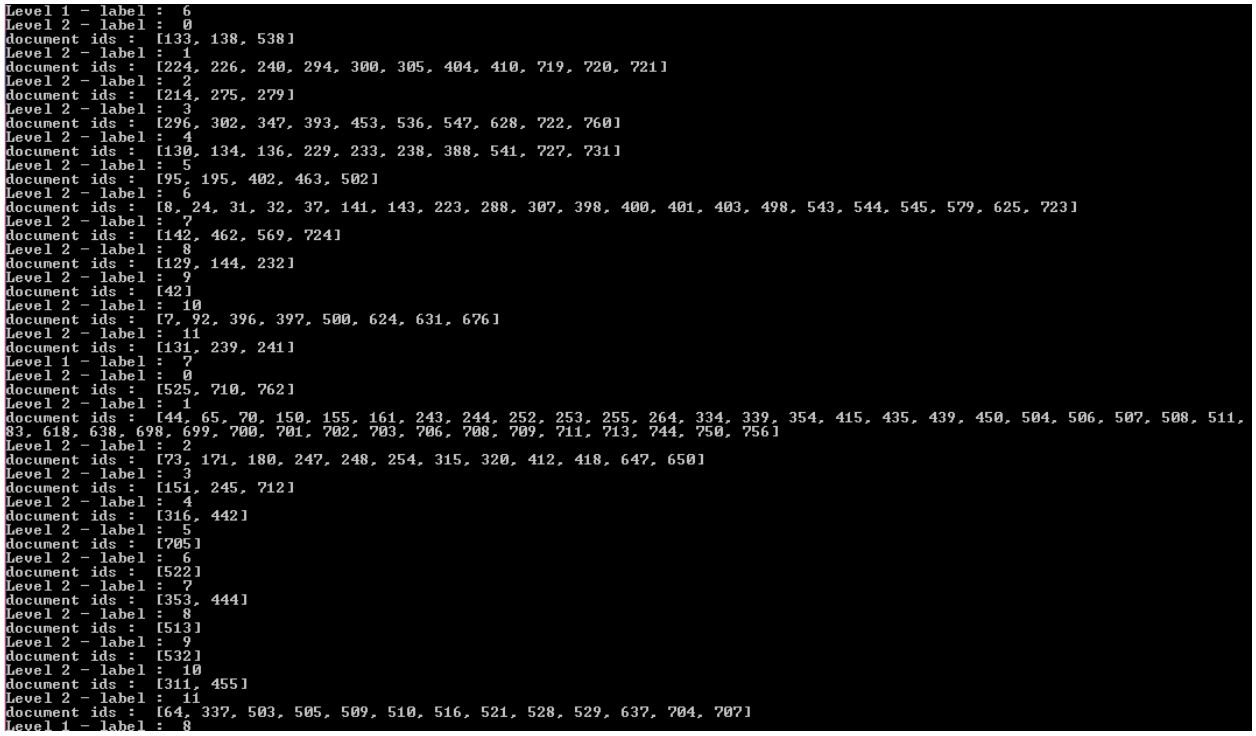


Fig 10.1 : Output of Document Clustering

The above figure is a screenshot of the output of Document Clustering algorithm. It shows the labels of clusters at level 1 and level 2 and the document ids that belong to that particular cluster.

Screenshot of User-Interface - I

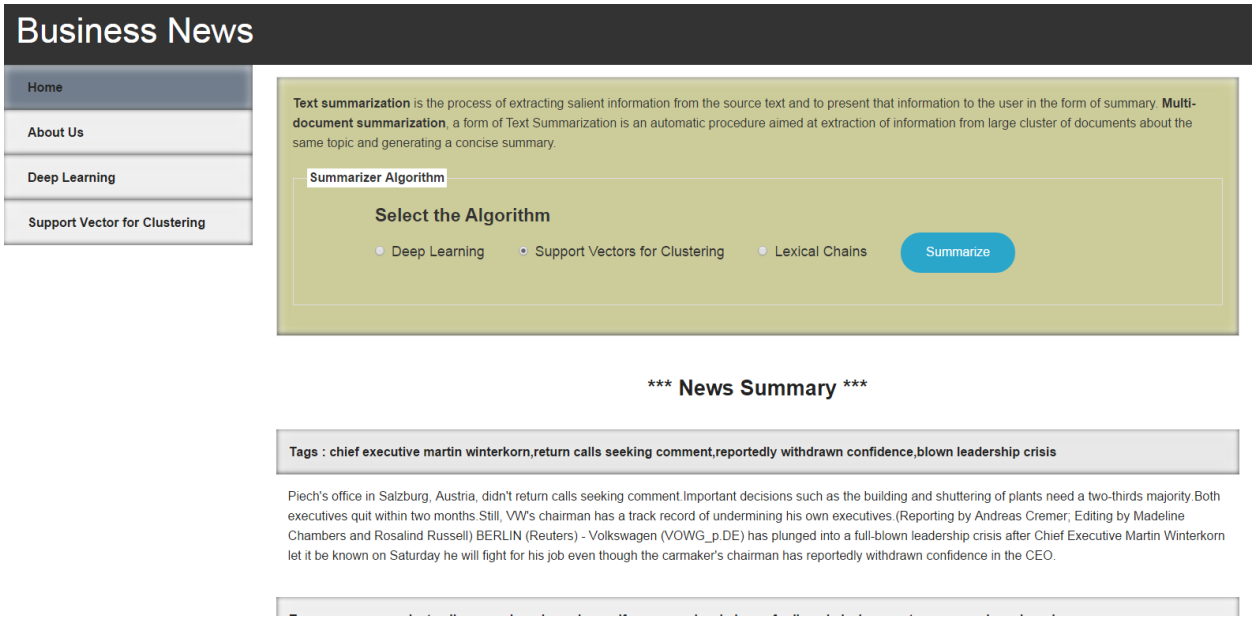


Fig 10.2 : Screenshot of web-portal I

The above screenshot was taken after choosing to generate the summaries from one of the algorithms. We can see the key-words tab along with their corresponding summary. The summaries will be generated in a tab below the terminologies and their definitions.

Screenshot of User - Interface – II

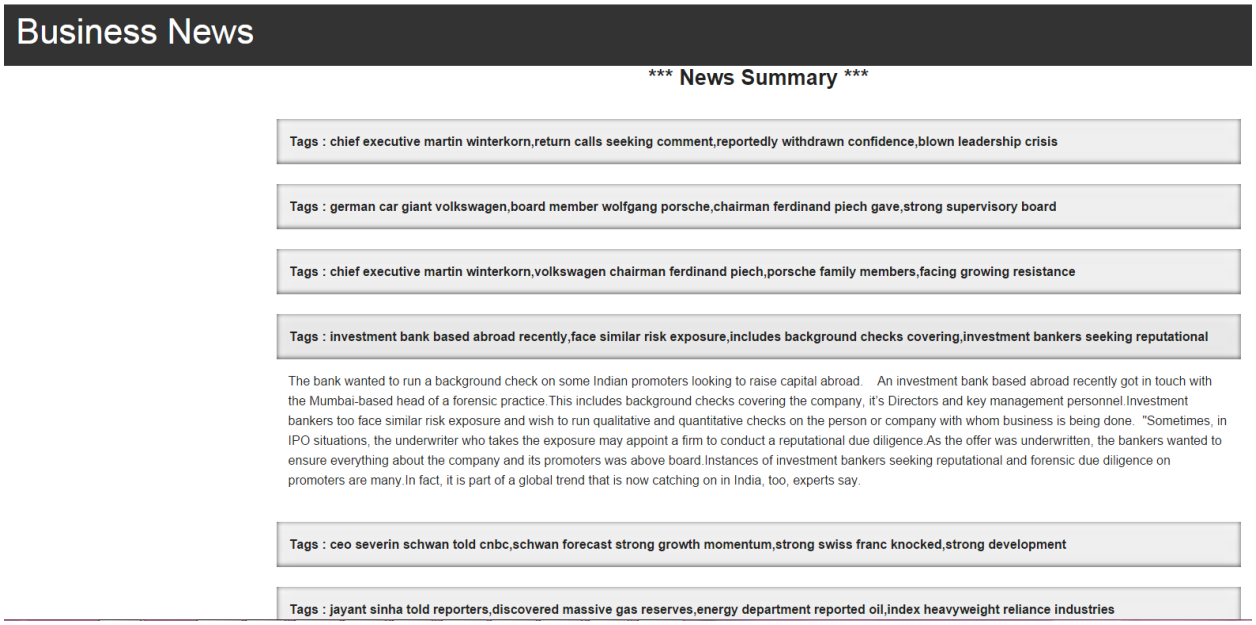


Fig 10.3 : Screenshot of web-portal II

The above figure shows another screen image of web-portal. This image shows the keyphrases generated for all the summaries. The tab for summary content for one cluster is active. To read the other summaries, we need to click on the corresponding key-phrases(Tags) tab.

Screenshot of Deep Learning Output

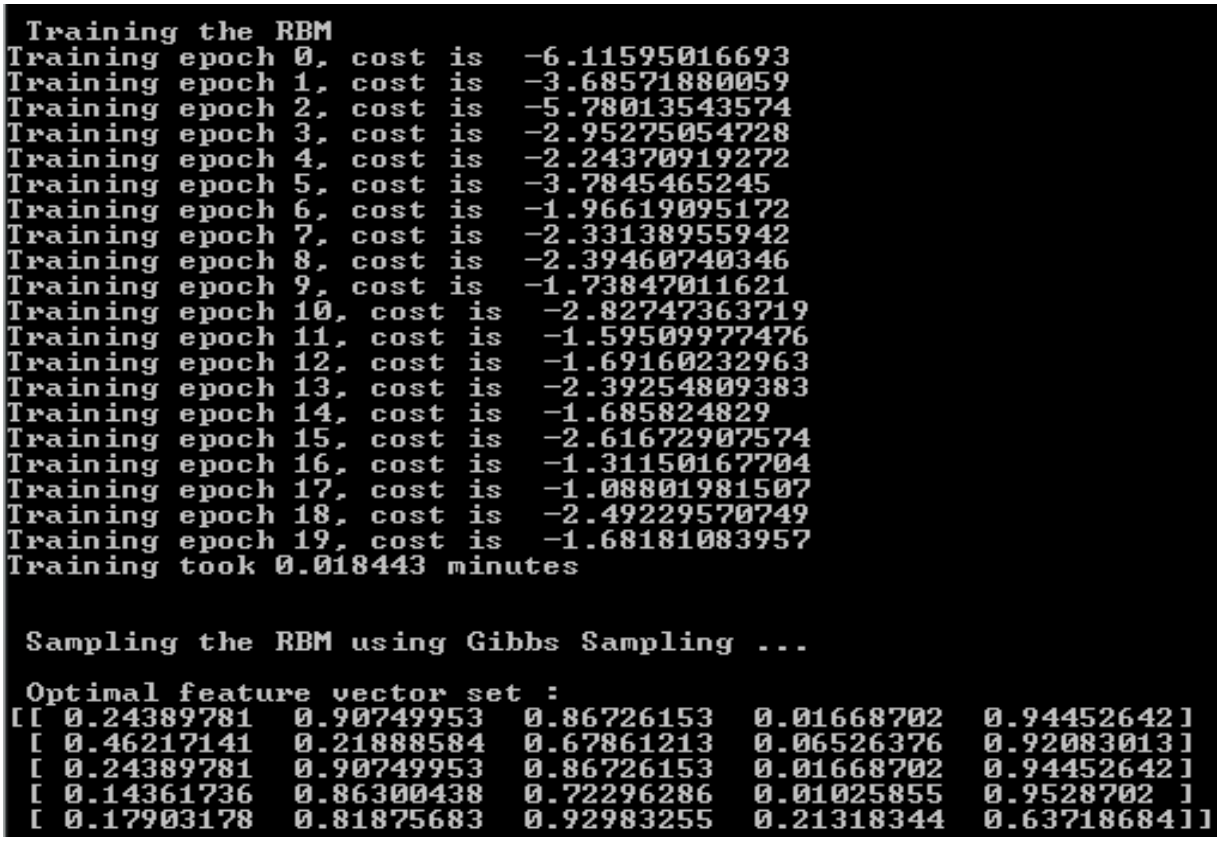


Fig 10.4 : Screenshot of Deep Learning Output

The above screenshot displays the output of Deep Learning module. It shows the cost for the training epochs. It also shows the optimal feature vector set obtained after sampling.

Screenshot of Lexical Chain Output

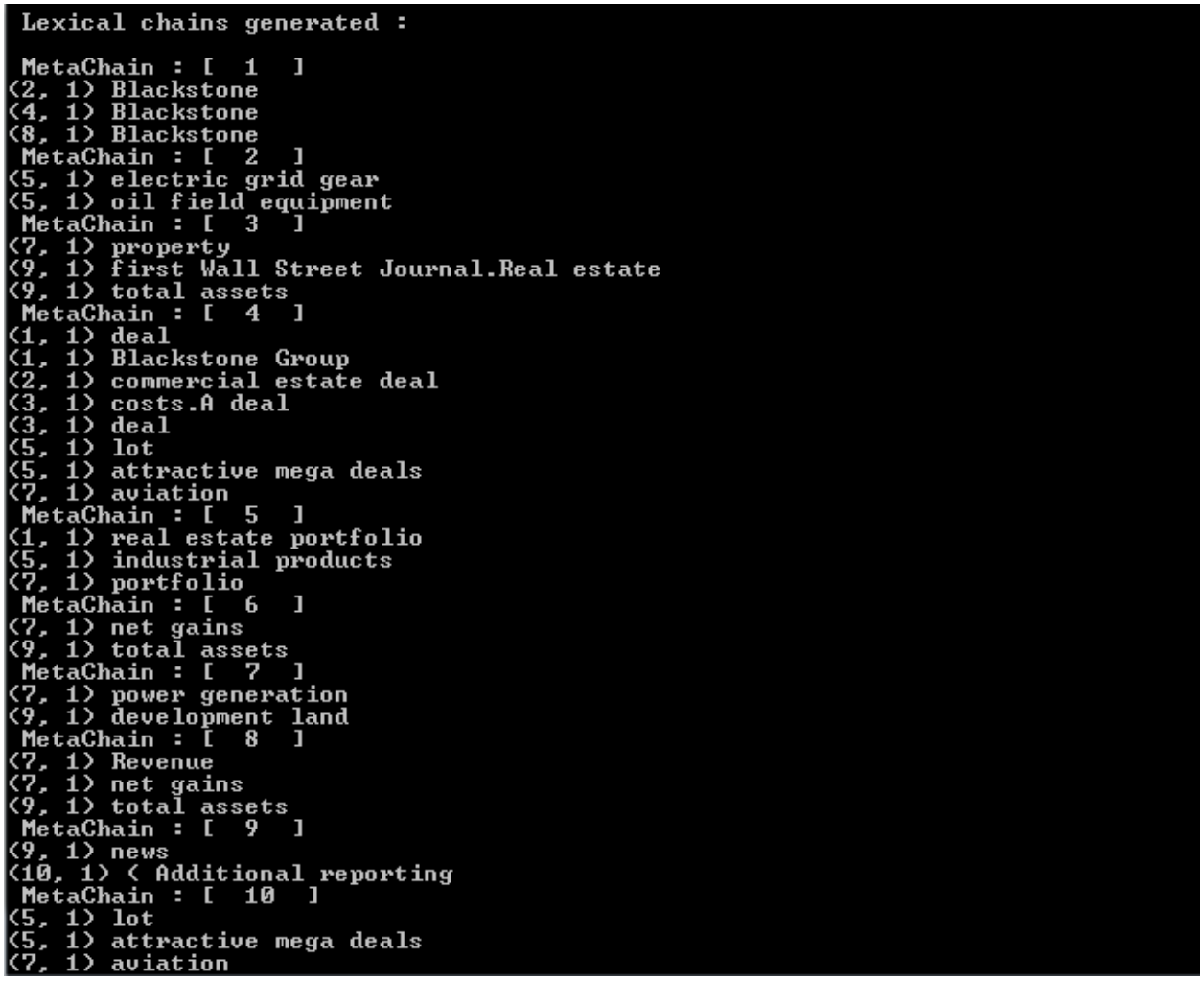


Fig 10.5 : Screenshot of Lexical Chains Output

The above screenshot displays the output of Lexical Chain Module. It shows the lexical chains that are generated using the algorithm.

Chapter - 11

Conclusion

The project "Abstractive Multi-Document Text Summarization", undertaken was completed in the duration of about four months. The performance of this system was satisfactory and the results obtained were as desired and conforms to the specific requirements. We implemented "Deep Learning", "Support Vector Clustering" and "Lexical Chain" to generate summaries. "Deep Learning" and "Support Vector Clustering" are our contributions to the field of Natural Language Processing specifically to Text Summarization. The system currently generate summary using a hybrid (Extractive and Abstractive) summarizer. The project can be further enhanced and developed into a full-fledged product.

Chapter - 12

Future Enhancement

The domain of "Text Summarization" is quite huge and challenging by itself. Each component that make up to a final Automatic Text Summarizer is a research topic today. Hence, there is always a very good scope to enhance a summarizer system in terms of its capabilities and performance and add new and different dimensionalities to it.

That being said, some of the future enhancements to the project, "**Abstractive Multi-Document Text Summarization**", can be the following.

- The algorithms currently run on a corpus of size - 0. To enhance the usability and apply the project in real-time, different parameters used needs to be fine-tuned more. Hence, we need to run the algorithms on an even larger corpus.

- The document clustering algorithm can be further fine-tuned to improve the performance of the system. Also, a new algorithm or a different algorithm could be used, such that number of clusters of documents to be generated can be decided dynamically based on number of articles in corpus.

- More feature functions to generate feature vectors can be added. The feature functions can represent semantic aspects. The quality of feature functions determines the quality of output of algorithms. Hence more and better feature functions will enhance the performance of the system as a whole.

Bibliography

[1] Daniel Jacob Gillick, "The Elements of Automatic Summarization", thesis submitted to University of California, Berkeley, 2011.

[2] Vasileios Hatzivassiloglou, Luis Gravano, Ankineedu MagantiAn, "Investigation of Linguistic Features and Clustering Algorithms for Topical Document Clustering", Proc. of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00), 2000.

[3] Kathleen R. McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L. Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, Sergey Sigelman, "Tracking and Summarizing News on a Daily Basis with Columbia's Newsblaster", Department of Computer Science 450, Computer Science Building, Columbia University.

[4] Michael Steinbach George Karypis Vipin Kumar, "A Comparison of Document Clustering Techniques", Department of Computer Science and Engineering, University of Minnesota.

[5] Vishal Gupta, "A Survey of Text Summarization Extractive Techniques", Journal Of Emerging Technologies In Web Intelligence, Vol. 2, No. 3, August 2010.

[6] Dipanjan Das, Andre F.T. Martins, "A Survey on Automatic Text Summarization", Language Institute of Technology, CMU November, 2007.

[7]Atif Khan, Naomie Salim, "A Review On Abtsractive Summarization Methods", Journal of Theoretical and Applied Information Technology

[8] Jackie C U Cheung, "Comparing Abstractive and Extractive Summarization of Evaluative Text: Controversiality and Content Selection", thesis submitted to University Of British Columbia, 2008

[9] PadmaPriya, G. and K. Duraiswamy, "An Approach For Text Summarization Using Deep Learning Algorithm", Journal of Computer Science: 1-9, 2014.

[10] Ben-Hur, Horn, Siegelmann and Vapnik, "Support Vector Clustering", Journal of Machine Learning Research 2 (2001): 125-137.

[11] Kees Jong, Elena Marchiori, Aad van der Vaart, "Finding Clusters using Support Vector Classifiers", Department of Mathematics and Computer Science Free University Amsterdam The Netherlands.

[12] Regina Barzilay and Michael Elhadad, "Using Lexical Chains for Text Summarization", Mathematics and Computer Science Dept., Ben Gurion University of Negev

[13] Alyona Medelyan, "https://www.airpair.com/nlp/keyword-extraction-tutorial", website

[14] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin and Craig G. Nevill-Manning, "KEA: Practical Automatic Keyphrase Extraction"

[15] G¨onen¸c Ercan, "Automatic Text Summarization and Keyphrase Extraction", a Master Degree thesis submitted to department of Computer Engineering, Bilkent University, September, 2006