DECLARATION

I declare that this written submission represents my ideas in my own words and where others Ideas or words have been included, I have adequately cited and referenced the original sources.

I also declare that I have adhered to all the principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which thus have not been properly cited or from whom proper permission have not been taken when needed.

Aditya Gadekar (XIEIT161707) _____

Atharva Mhatre (XIEIT161725) _____

Rushabh Sarvaiya (XIEIT161747) _____

Date:

Table of Content

List of Tables

List of Figures

Abstract

Right now huge data is accessible on the Internet, it is generally imperative to give the improved system to remove the data rapidly and most proficiently. It is extremely hard for individuals to physically extricate the synopsis of a huge archives of content. There are a lot of content material accessible on the Internet. So there is an issue of scanning for significant records from the quantity of reports accessible, and engrossing applicable data from it. In request to take care of the over two issues, the Automatic Text Summarization is particularly necessary. Text Summarization is the way toward distinguishing the most significant important data in an archive or set of related records and packing them into a shorter form protecting its general implications. Before heading off to the Text Summarization, first we, need to realize that what an Summary is. A Summary is a book that is delivered from at least one messages, that passes on significant data in the first content, and it is of a shorter structure. The objective of Automatic Text Summarization is exhibiting the source content into a shorter adaptation with semantics. The most significant bit of leeway of utilizing a Summary is, it lessens the understanding time. Content Summarization techniques can be characterized into extractive and abstractive synopsis. An extractive Summary technique comprises of choosing significant sentences, sections and so on from the first report and connecting them into shorter structure. An Abstractive outline is a comprehension of the primary ideas in a record and afterward express those ideas in clear regular language

Acknowledgement

I would like to thank Fr (Dr). John Rose S.J. (Director of XIE) for providing us with such an environment so as to achieve goals of our project and supporting us constantly.

I express my sincere gratitude to our Honorable Principal Mr. Dr. Y.D.Venkatesh for encouragement and facilities provided to us.

I would like to place on record our deep sense of gratitude to Prof.Chhaya Narvekar, Head of Department Of Information Technology, Xavier Institute of Engineering, Mahim, Mumbai, for her generous guidance help and useful suggestions.

With deep sense of gratitude I acknowledge the guidance of our project guide Prof. Martina D'Souza. The time-to-time assistance and encouragement by her has played an important role in the development of our project.

I would also like to thank our entire Information Technology staff who have willingly cooperated with us in resolving our queries and providing us all the required facilities on time.

Aditya Gadekar(XIEIT161707) _____

Atharva Mhatre (XIEIT161725) _____

Rushabh Sarvaiya(XIEIT161747) _____

CHAPTER 1

INRTODUCTION

Today we know that machines have become smarter than us and can help us with every aspect of life, the technologies have reached to an extent where they can do all the tasks of human beings like household tasks, controlling home devices, making appointments etc. The field which makes these things happen is Machine Learning. Machine Learning train the machines with some data which makes it capable of acting when tested by the similar type of data. The machines have become capable of understanding human languages using Natural Language Processing. Today researches are being done in the field of text analytics. As the project title suggests, Text Summarizer is a web-based application which helps in summarizing the text. We can upload our data and this application gives us the summary of that data in as many numbers of lines as we want. The product is mainly a text summarizing using Deep Learning concepts. The main purpose is to provide reliable summaries of web pages or uploaded files depends on the user's choice. AI train the machines with a few information which makes it fit for

acting when tried by the comparable kind of information. Themachines have gotten equipped for understanding human dialects utilizing Natural Language Processing. Today investigates are being done in the field of content examination. As the undertaking title suggests, Text Summarizer is an online application which helps in condensing the text.We can transfer our information and this application gives us the rundown of that information in as many numbers of lines as we need. The item is chiefly a content abridging utilizing Deep Learning concepts. The fundamental design is to give dependable outlines of website pages or transferred files depends on the client's decision.

1.1 Problem Definition

It is very difficult for human beings to manually extract the summary of a large documents of text. There are plenty of text material available on the Internet. So there is a problem of searching for relevant documents from the number of documents available, and absorbing relevant information from it.In order to solve the above two problems, the automatic text summarization is very much necessary.Text summarization is the process of identifying the most important meaningful information in a document or set of related documents and compressing them into a shorter version preserving its overall meanings.

1.2 Scope of the project

Generally, there are two approaches to automatic summarization: extraction and abstraction. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. First clean the text file by removing full stop, common words (conjunction, verb, adverb, preposition etc.). Then calculate the frequency of each words and select top words which have maximum frequency. This technique retrieves important sentence emphasize on high information richness in the sentence as well as high Information retrieval. These related maximum sentence generated scores are clustered to generate the summary of the document. Thus we use k-mean clustering to these maximum sentences of the document and find the relation to extract clusters with most relevant sets in the document, these helps to find the summary of the document.

1.3 Existing System

The current text summarization system just refers to the technique of shortening long pieces of text. The intention is to create a coherent and fluent summary having only the main points outlined in the document. Automatic text summarization is a common problem in machine learning and natural language processing. Machine learning models are usually trained to understand documents and distill the useful information before outputting the required summarized texts.

1.4 Proposed System

A text summarizer app will be created where you need to copy the content you need to summarize and then select the number of lines for the summary. This app will have other features like text to speech where you need to select the summarized text and select the TTS option which is a feature by google which will read the entire summary. Another feature is image to text which will be helpful when you don't have a text document. In image to text you need to capture an image from your phone or select an image file which will be converted to text document

CHAPTER 2

REVIEW OF LITERATURE

2.1 Display Methods available for Implementing Text Summarization

There are mainly two ways to make the summary. Extractive and Abstractive

1.Extractive Summarization

Select relevant phrases of the input document and concatenate them to form a summary (like "copy-and-paste"). Pros: They are quite robust since they use existing natural-language phrases that are taken straight from the input. Cons: But they lack in flexibility since they cannot use novel words or connectors. They also cannot paraphrase like people sometimes do.

Categories of extractive summarization.

1. Graph Base

The graph base model makes the graph from the document, then summarize it by considering the relation between the nodes (text-unit). TextRank is the typical graph based method.

2. TextRank

TextRank is based on PageRank algorithm that is used on Google Search Engine. Its base concept is "The linked page is good, much more if it from many linked page". The links between the pages are expressed by matrix (like Round-robin table). We can convert this matrix to transition probability matrix by dividing the sum of links in each page. And the page surfer moves the page according to this matrix.

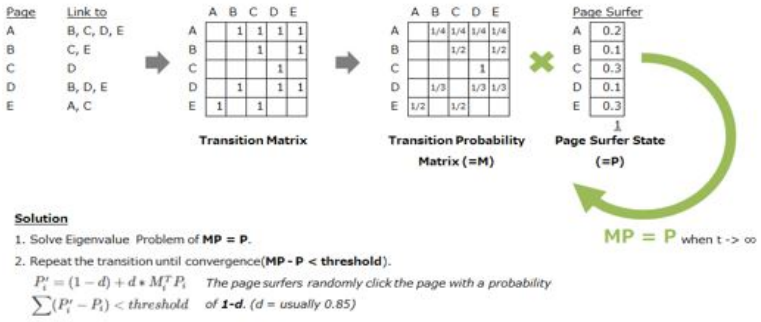

Figure 2.1: Figure 2.1: Text Rank Algorithm

3. Page Rank Algorithm

TextRank regards words or sentences as pages on the PageRank. So when you use the TextRank, following points are important. Define the "text units" and add them as the nodes in the graph. Define the "relation" between the text units and add them as the edges in the graph. You can set the weight of the edge also. Then, solve the graph by PageRank algorithm. LexRank uses the sentence as node and the similarity as relation/weight (similarity is calculated by IDF-modified Cosine similarity). If you want to use TextRank, following tools support TextRank.[7] • gensim • pytextrank

TextRank views words or sentences as pages on the PageRank. So when you utilize the TextRank, following focuses are significant. Characterize the "content units" and include them as the nodesin the chart. Characterize the "connection" between the content units and include

them as the edges in thegraph. You can set the heaviness of the edge too. At that point, comprehend the chart by PageRank algorithm.LexRank utilizes the sentence as hub and the comparability as connection/weight (closeness is calcu-lated by IDF-changed Cosine likeness).

2.Abstractive

Generates a summary that keeps original intent. It's just like humans do. Pros: They can use words that were not in the original input. It enables to make more fluent and natural summaries. Cons: But it is also a much harder problem as you now require the model to generate coherent phrases and connectors. Extractive Abstractive is not conflicting ways. You can use both to generate the summary. And there are a way collaborate with human.

1. Aided Summarization

Combines automatic methods with human input. Computer suggests important information from the document, and the human decide to use it or not. It uses information retrieval, and text mining way. The beginning of the abstractive summarization, Banko et al. (2000) suggest to use machine translatation model to abstractive summarization model. As like the machine translation model converts a source language text to a target one, the summarization system converts a source document to a target summary. Nowadays, encoder-decoder model that is one of the neural network models is mainly used in machine translation. So this model is also widely used in abstractive summarization model. The summarization model that used encoder-decoder model first achieved state-of-the-art on the two sentence-level summarization dataset, DUC-2004 and Gigaword. If you want to try the encoder-decoder summarization model, tensorflow offers basic model. • Text summarization with TensorFlow

2. Encoder-Decoder Model

The encoder-decoder model is composed of encoder and decoder like its name. The encoder converts an input document to a latent representation (vector), and the decoder generates a summary by using it.

But the encoder-decoder model is not the silver bullet. There are many remaining issues are there. • How to set the focus on the important sentence, keyword. • How to handle the novel/rare (but important) word in source document. • How to handle the long document. • Want to make more human-readable summary. • Want to use large vocabulary.

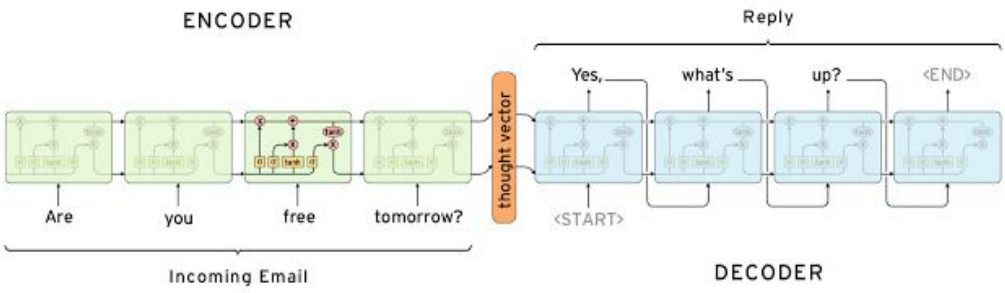

Figure 2.2: Figure 2.2: Encoder Decoder Model

3.Spyder Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. It is released under the MIT license. Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu. Spyder uses Qt for its GUI, and is designed to use either of the PyQt or PySide Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend .

4.HTML and CSS

Cascading style sheets will prove to be one of the most powerful tools available for the web designer. Most people who have used a word processor for any length of time will have come across the principles of style sheets and maybe have used particular style sheets to achieve specific results.[6] Stylesheets are templates containing rules that describe how a browser should display documents on screen, in print, or in other media. Cascading Style Sheets (CSS) are the major tool for controlling the presentation of XHTML and XML documents. Clearly, without some control of presentation, the World Wide Web would be much less robust and visually pleasing an experience than it is today. There are other presentational tools besides CSS, but they have disadvantages.[7] Style sheets are based on the use of the STYLE tag, which allows us to define new styles, for example fonts, to use on our web page. Style sheets or Cascading Style Sheets (CSS), as DHTML authors usually refer to them, allow us also to define blocks of content. Once these are defined we can control their position on the web page, that is we can

cascade them.

5.PHP(Hypertext Processor)

Open-source, server-side scripting language used to generate dynamic web-pages PHP scripts reside between reserved PHP tags. This allows the programmer to embed PHP scripts within HTML pages. Various built-in functions allow for fast development Compatible with many popular databases. PHP scripts are executed on the server. PHP files have a file extension of ".php", ".php3", or ".phtml.

6.Android Studio

Android Studio is the official IDE (Integrated Development Environment) for Android app development. This IDE is recommended by Google and built off of the IntelliJ's powerful code, combined with the increased flexibility and productivity of Android Studio's platform. While there are a lot of IDEs out there and their different uses will better suit certain developers than others, Android Studio is made by Google specifically for designing apps for their service. Going straight to the source is a surefire way to make sure your app will work on their platform. Keep in mind as well that IDEs for developing are kind of like Photoshop. Like with Photoshop, there are countless features, but you will probably only need to learn them as you need them. Android Studio is a good way to gain a ground level knowledge of exactly how to work within their system.

7.Text to Speech

Google Text-to-Speech is a screen reader application developed by Google for its Android operating system. It powers applications to read aloud (speak) the text on the screen which support many languages. Text-to-Speech may be used by apps such as Google Play Books for reading books aloud, by Google Translate for reading aloud translations providing useful insight to the pronunciation of words, by Google Talkback and other spoken feedback accessibility-based applications, as well as by third-party apps. Users must install voice data for each language

8.Image to text Optical character recognition or optical character reader, often abbreviated as OCR, is the mechanical or electronic conversion of images of typed, handwritten or printed

text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast). Widely used as a form of information entry from printed paper data records – whether passport documents, invoices, bank statements, computerised receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitising printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition.

There are various steps and aspects that are needed to be taken into consideration while doing text summarization.

Sentence Ordering: There are several sentences in a paragraph which need to be put in proper Order. The sentences with more importance and the sentence with comparatively less importance need to be arranged appropriately. So it takes several sentences which have been deemed important by the extractive summarizer and these sentences are presented in logical order. [1] In order to do this it is very important to identify key words in a paragraph which is done using Text Rank Algorithm. TextRank is a calculation dependent on PageRank, which regularly utilized in catchphrase extraction and content synopsis. Right now, will assist you with seeing how TextRank functions with a catchphrase extraction model and show the usage by Python [3].

Sentence revision: Sentence revision was verifiably the principal language age task endeavored with regards to outline. Sentence revision includes re-utilizing content gathered from the contribution to the summarizer, however parts of the last outline are consequently changed by subbing a few articulations with other increasingly suitable articulations, given the setting of the new synopsis. Kinds of revisions proposed by early analysts incorporate disposal of pointless pieces of the sentences, mix of data initially communicated in various sentences and substitution of a pronoun with a progressively expressive thing phrase where the setting of the outline requires this. Given that usage of these revision activities can be very mind boggling, analysts in the field in the long run. [1].

Sentence Compression and Multi Level fusion: Automatic sentence compression and MSF are an ongoing expansion to inquire about in automatic summarization. Truth be told, they can be considered both as a sort of rundown and as an integral assignment to remove summarization.

Automatic sentence compression comprises of disposing of the unnecessary components in a sentence. The procedure additionally includes sentence combination. changing language structure and choosing sentences. Sentence compression can, in any case, be considered as one kind of nonexclusive rundown, which contains an improved variant of each sentence. MSF comprises of producing a sentence from the combination of a gathering of k related sentences. In this manner, the fundamental target is to lessen excess, while keeping up the basic data contained in a gathering of sentences.

CHAPTER 3

System Design and Analysis

3.1 Analysis

1.Use Case Diagram

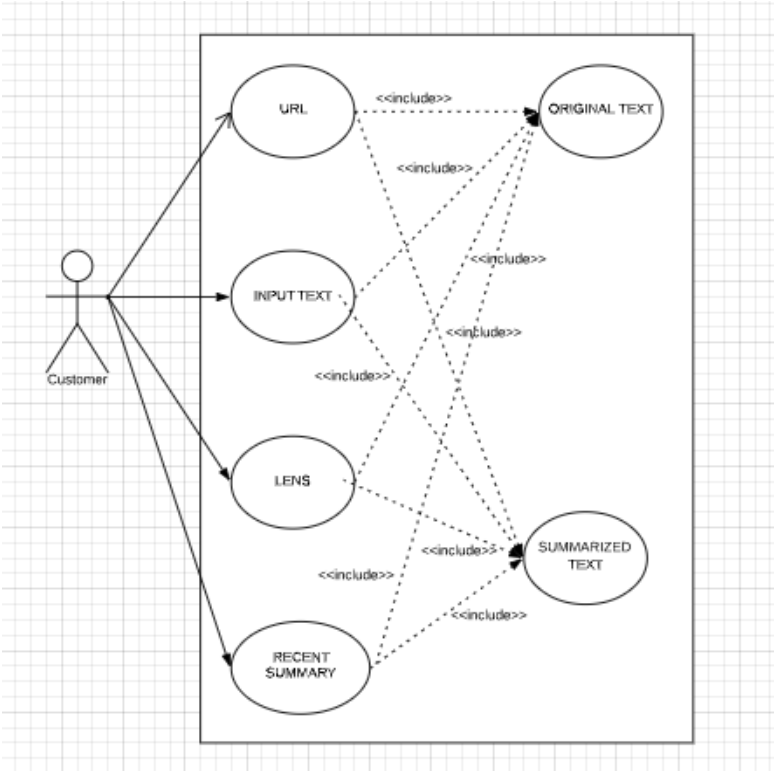

Figure 3.1: Use Case diagram

1.Sequence Diagram

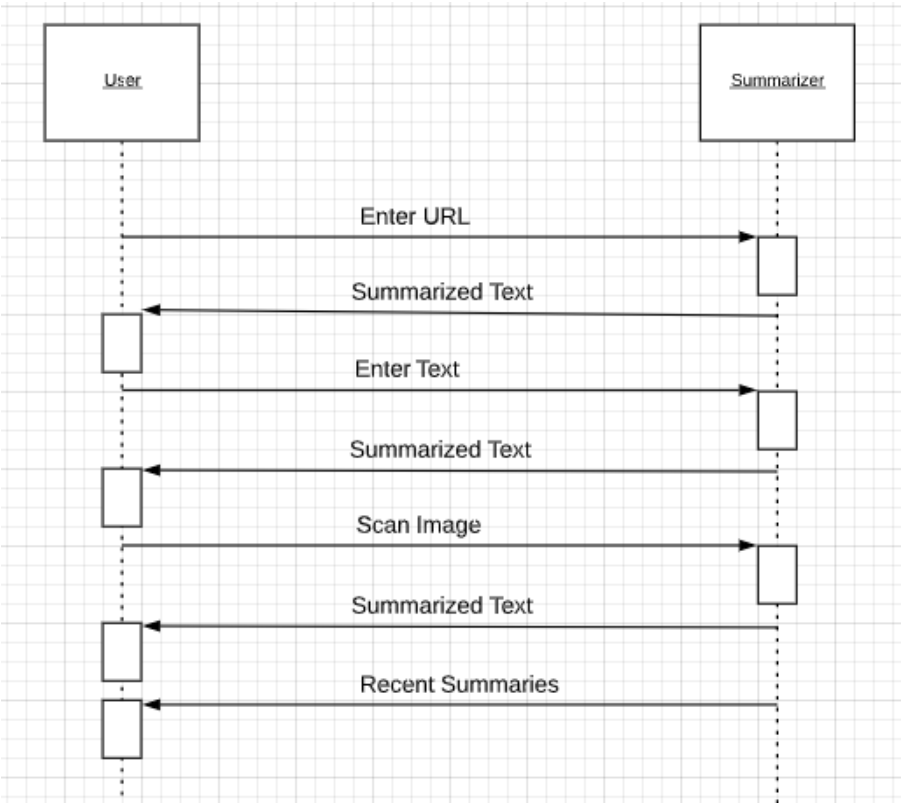

Figure 3.2: Sequence Diagram

3.2 Design

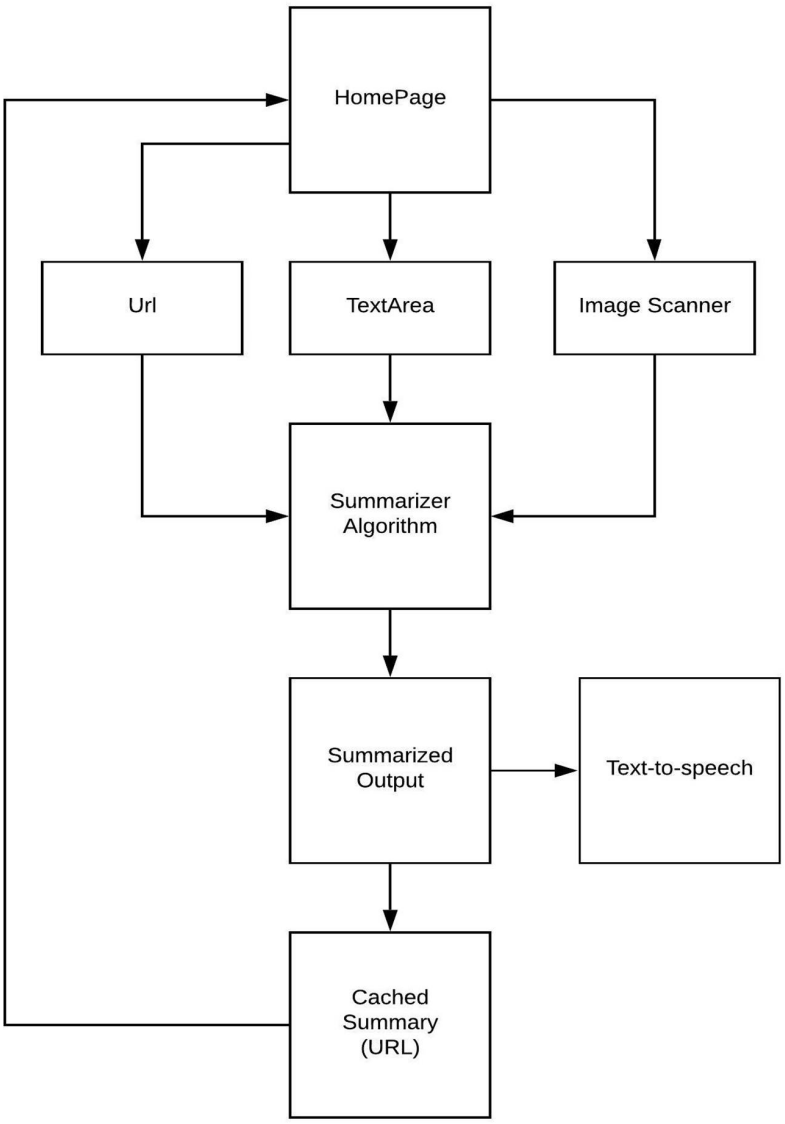

Figure 3.3: Block diagram

3.3 Implementation Methodology

Usually, text summarization in NLP is treated as a supervised machine learning problem (where future outcomes are predicted based on provided data). Typically, here is how using the extraction-based approach to summarize texts can work: 1. Introduce a method to extract the merited keyphrases from the source document. For example, you can use part-of-speech tagging, words sequences, or other linguistic patterns to identify the keyphrases. 2. Gather text documents with positively-labeled keyphrases. The keyphrases should be compatible to the stipulated extraction technique. To increase accuracy, you can also create negatively-labeled keyphrases. 3. Train a binary machine learning classifier to make the text summarization. Some of the features you can use include: Length of the keyphrase Frequency of the keyphrase The most recurring word in the keyphrase Number of characters in the keyphrase 4. Finally, in the test phrase, create all the keyphrase words and sentences and carry out classification for them.

For the most part, content outline in NLP is treated as a managed AI prob-lem (where future results are anticipated dependent on gave information). Ordinarily, here ishow utilizing the extraction-based way to deal with outline writings can work:

1. Introducea technique to separate the justified keyphrases from the source archive. For example,you can utilize grammatical feature labeling, words groupings, or other semantic examples to iden-tify the keyphrases.

2. Accumulate content reports with decidedly named keyphrases. The keyphrases ought to be perfect to the stipulated extraction method. To build air conditioning curacy, you can likewise make contrarily marked keyphrases.

3. Train a twofold machinelearning classifier to make the content synopsis. A portion of the highlights you can useinclude: Length of the keyphrase Frequency of the keyphrase The most repeating wordin the keyphrase Number of characters in the keyphrase

4. At long last, in the test phrase,create all the keyphrase words and sentences and complete arrangement for them

3.4 Details of Hardware & Software

Recommended System Requirements

- **Processors:** Intel R CoreTM i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM
- **Disk space:** 2 to 3 GB
- **Operating systems:** Windows 10, macOS, and Linux

Minimum System Requirements

- **Processors:** Intel Atom® processor or Intel® Core™ i3 processor
- **Disk space:** 1 GB
- **Operating systems:** Windows 7 or later, macOS, and Linux

Software

- **Php**
- **Python**
- **Android Studio**
- **HTML and CSS**
- **JavaScript**

CHAPTER 4

IMPLEMENTATION

We will approach this project in two major phases. These phases are as follows :

1. Web interface 2. Machine learning module

4.1 Web interface

We'll start the project implementation by designing the user interface using php to create a Application. Web interface will be using machine learning module as the base to provide user the experience they deserve. Web interface will be developed using HTML5, CSS3 and BootStrap Android Studio.

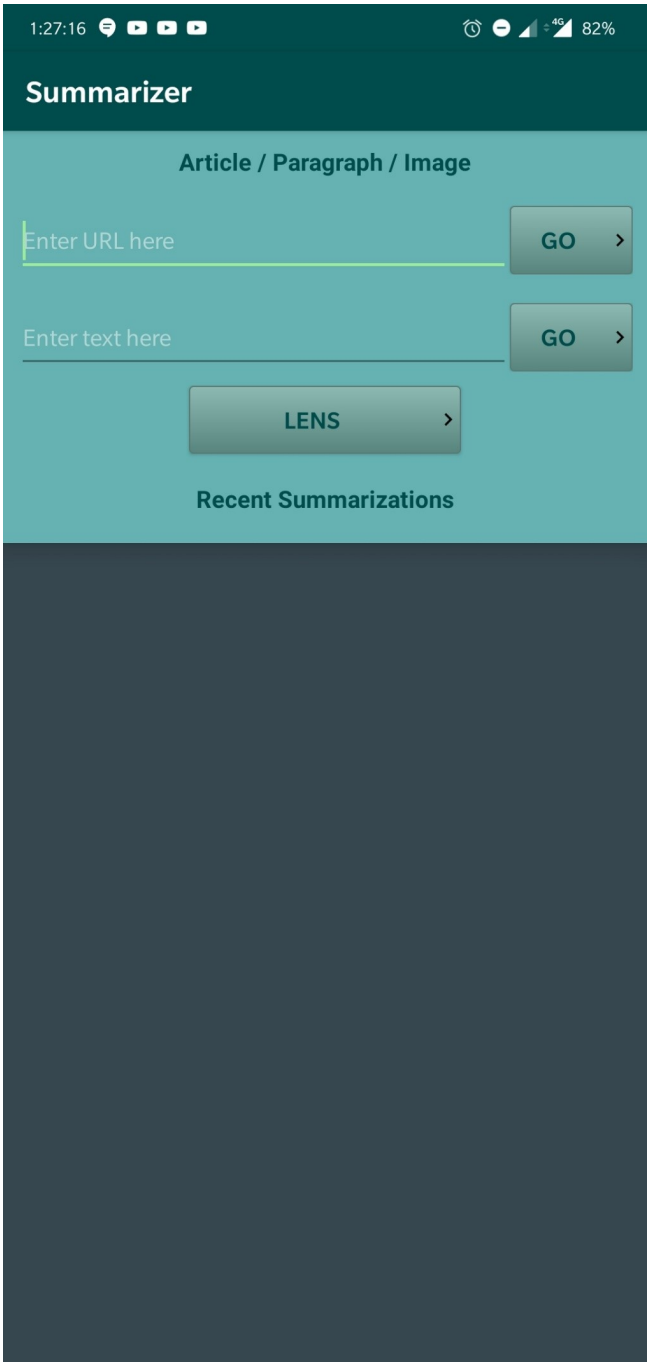

Figure 4.1: Homepage

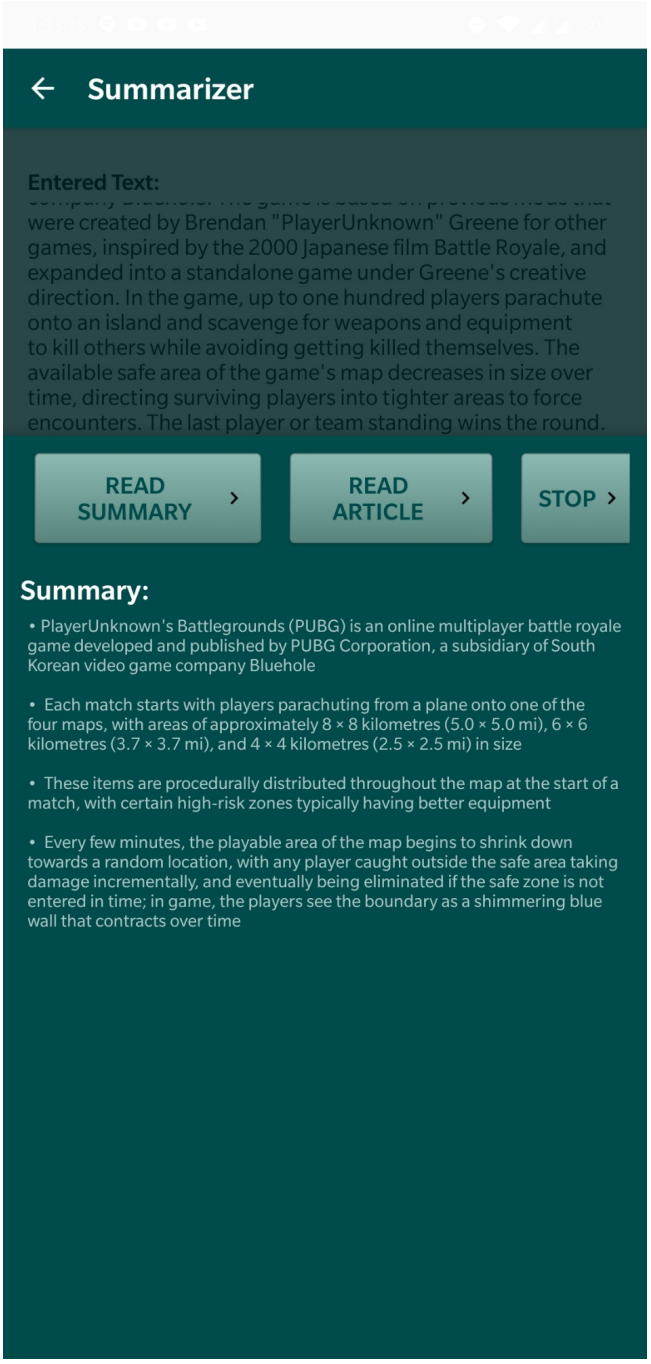

Figure 4.2: Summarized Paragraph

4.2 Machine learning module

This module will be made by using python language. This module is the actual brains of the entire system. In this module an algorithm will be used to learn about the data of the students such as academic scores, communication skills, technical skills, etc along with the criteria of the companies. After obtaining all the information and interpreting them, the algorithm will

Figure 4.3: Summarized text from webpage

classify whether the student will be placed or not for the given company.

4.3 Text Rank Algorithm

Extractive methods — Involves the selection of phrases and sentences from the source document to make up the new summary.

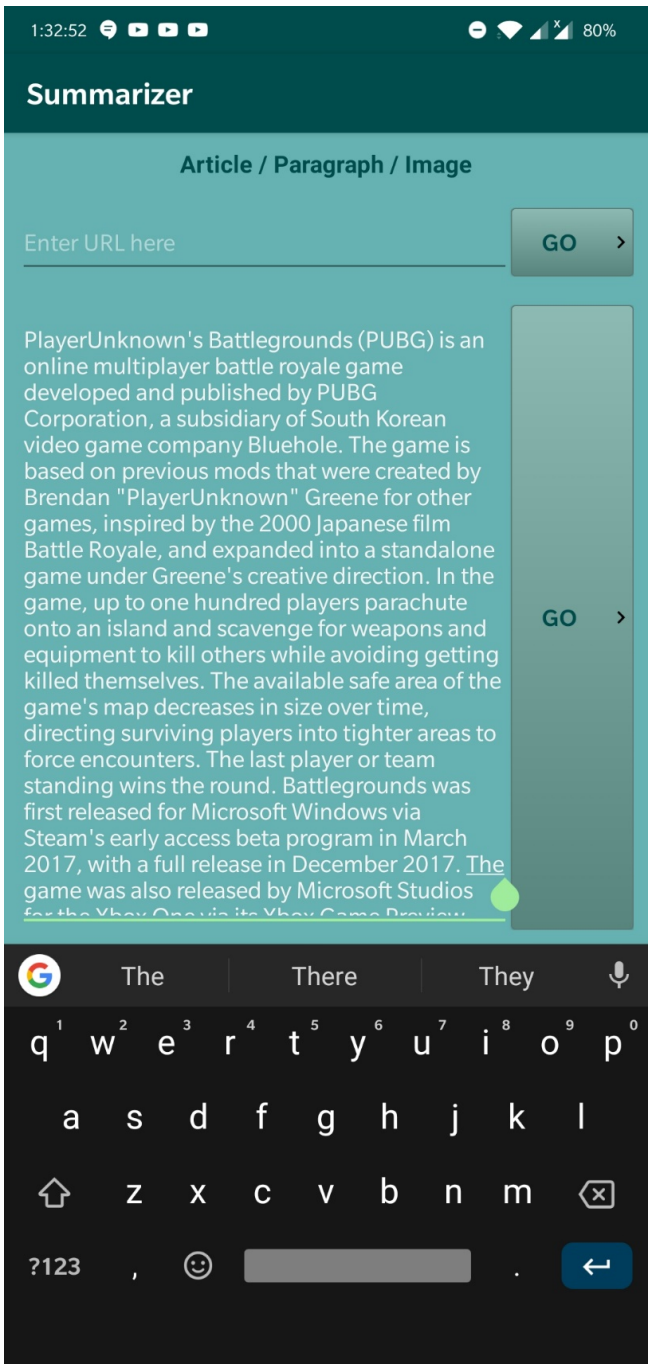

Figure 4.4: Summarizing paragraph

Abstractive methods- It involves generating entirely new phrases and sentences to capture the meaning of the source document.

Gensim is a free Python library designed to automatically extract semantic topics from documents. The gensim implementation is based on the popular TextRank algorithm. It is an open-source vector space modelling and topic modelling toolkit, implemented in the Python programming language, using NumPy, SciPy and optionally Cython for performance.

Text Summarisation with Gensim (TextRank algorithm)- We use the summarization.summarizer

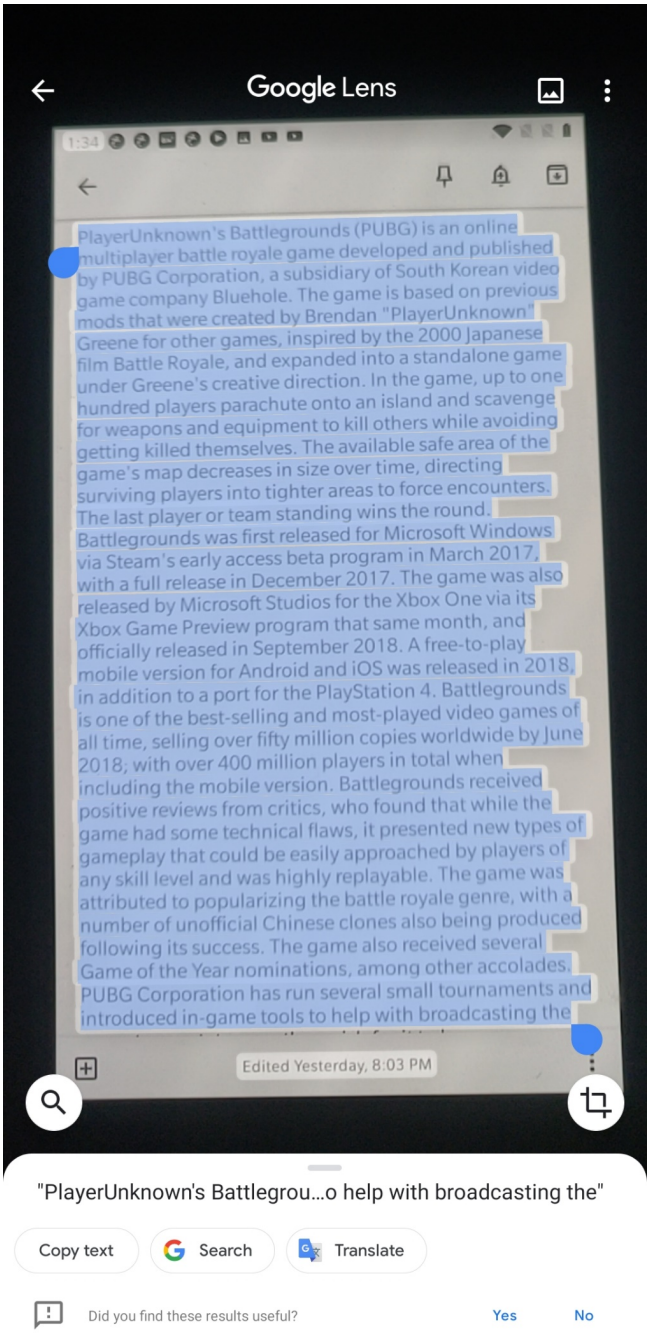

Figure 4.5: Summarizing text from an image

from gensim. This summarising is based on ranks of text sentences using a variation of the TextRank algorithm. TextRank is a general purpose, graph based ranking algorithm for NLP. TextRank is an automatic summarisation technique. Graph-based ranking algorithms are a way for deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph.

TextRank Model - The basic idea implemented by a graph-based ranking model is that of voting or recommendation. When one vertex links to another one, it is basically casting a vote

for that vertex. The higher the number of votes cast for a vertex, the higher the importance of that vertex.

Text as a graph - We have to build a graph that represents the text, interconnects words or other text entities with meaningful relations.

TextRank includes two NLP tasks- Keyword extraction task Sentence extraction task

Keyword Extraction - The task of keyword extraction algorithm is to automatically identify in a text a set of terms that best describe the document. The simplest possible approach is to use a frequency criterion. HOWEVER, this leads to poor results. The TextRank keyword extraction algorithm is fully unsupervised. No training is necessary.

Sentence Extraction - TextRank is very well suited for applications involving entire sentences, since it allows for a ranking over text units that is recursively computed based on information drawn from the entire text. To apply TextRank, we first build a graph associated with the text, where the graph vertices are representative for the units to be ranked. The goal is to rank entire sentences, therefore, a vertex is added to the graph for each sentence in the text.

```
from gensim.summarization.summarizer import summarize
from gensim.summarization import keywords
import urllib.request
from urllib.request import urlopen
from bs4 import BeautifulSoup
import requests
!pip install lxml

#getting a text document from the net
text1=requests.get('http://rare-technologies.com/the_matrix_synopsis.txt').text

#getting a text document from file
fname="../starwars.txt"
with open(fname, 'r') as myfile:
    text2=myfile.read()

def get_only_text(url):

    """
    return the title and the text of the article at the
    specified url
    """

    page=urlopen(url)
    soup=BeautifulSoup(page,"html.parser")
    text=' '.join(map(lambda p: p.text, soup.find_all('p')))
    return soup.title.text, text
```

Figure 4.6: Mapping Algorithm

It is the foundation of TextRank. PageRank used by Google search. Used to compute the rank of web pages. It is not named after its use (ranking pages) but after its creator Larry Page. Fundamentals - Important pages are linked by important pages. The PageRank value of a page is the probability of a user visiting that page.

```
#text from url

url="https://en.wikipedia.org/wiki/Kellogg%27s"
text3=get_only_text(url)

print('Summary:')
print(summarize(str(text3), word_count=300))

#text from web

print('Summary:')
print(summarize(text1, ratio=0.045))


#text file from text document

print('Summary:')
print(summarize(text2, word_count=100))
```

Figure 4.7: Mapping Algorithm

4.4 Code

Algorithm

Algorithm textrANK:

```
package textrank;
import android.util.Log;
import org.jgrapht.graph.DefaultEdge;
import org.jgrapht.graph.DefaultWeightedEdge;
import org.jgrapht.graph.SimpleGraph;
import org.jgrapht.graph.SimpleWeightedGraph;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedHashSet;
```

```
import opennlp.tools.sentdetect.SentenceDetectorME;
import opennlp.tools.sentdetect.SentenceModel;
import opennlp.tools.tokenize.Tokenizer;
import opennlp.tools.tokenize.TokenizerME;
import opennlp.tools.tokenize.TokenizerModel;
public class TextRank {
    private SimpleWeightedGraph<SentenceVertex, DefaultWeightedEdge> graph;
    private SimpleGraph<TokenVertex, DefaultEdge> tokenGraph;
    private SentenceDetectorME sdetector;
    private Tokenizer tokenizer;
    private final double CONVERGENCE_THRESHOLD = 0.0001;
    private final double PROBABILITY = 0.85;
    private final int COOCCURENCE_WINDOW = 2;
    private final HashSet<String> stopwords = new HashSet<String>();
    private final HashSet<String> extendedStopwords = new HashSet<String>();
    public TextRank(InputStream sent, InputStream token,
    InputStream stop, InputStream exstop) throws IOException {
        init(sent, token, stop, exstop);
    }
    private void init(InputStream sent, InputStream token, InputStream stop,
    InputStream exstop) throws IOException {
        // creates a new SentenceDetector, POSTagger, and Tokenizer
        SentenceModel sentModel = new SentenceModel(sent);
        sent.close();
        sdetector = new SentenceDetectorME(sentModel);
        TokenizerModel tokenModel = new TokenizerModel(token);
        token.close();
        tokenizer = new TokenizerME(tokenModel);
        BufferedReader br = new BufferedReader(new InputStreamReader(stop));
        String line;
        while ((line = br.readLine()) != null) {
            stopwords.add(line);
```

```
    }
    br.close();
    br = new BufferedReader(new InputStreamReader(exstop));
    while ((line = br.readLine()) != null) {
        extendedStopwords.add(line);
    }
    br.close();
}
private void createGraph(String article){
    graph =  new SimpleWeightedGraph<SentenceVertex, DefaultWeightedEdge>
    (DefaultWeightedEdge.class);
    String[] sentences = sdetector.sentDetect(article);
    for(String sentence: sentences){
        //Remove punctuation for each sentence
        SentenceVertex sv = new SentenceVertex(sentence, tokenizer.tokenize
        (sentence.replaceAll("\\p{P}", "")));
        graph.addVertex(sv);
    }
    //Create edges
    for(SentenceVertex v1: graph.vertexSet()){
        for(SentenceVertex v2: graph.vertexSet()){
            if(v1 != v2){
                DefaultWeightedEdge dwe = graph.addEdge(v1, v2);
                //If the edge hasn't yet been added
                if(dwe != null) {
                    double weight = calculateSimilarity(v1, v2);
                    if(weight > 0.0) {
                        graph.setEdgeWeight(dwe, weight);
                    }
                    else{
                        graph.removeEdge(dwe);
                    }
```

```
                }
            }
        }
    }
}
private double calculateSimilarity(SentenceVertex v1, SentenceVertex v2){
    //Use HashSet to ensure efficient intersection of tokens [ O(N) ]
    String[] s1TokenArray = v1.getTokens();
    HashSet<String> s1TokenHashSet = v1.getTokensAsHashSet();
    String[] s2TokenArray = v2.getTokens();
    //Loop through tokens in second sentence
    double similarities = 0;
    for (int i = 0; i < s2TokenArray.length; i++) {
        String word = s2TokenArray[i];
        //Check if in first sentence
        if(s1TokenHashSet.contains(word)){
            similarities++;
        }
    }
    int numWordsInSentence1 = s1TokenArray.length;
    int numWordsInSentence2 = s2TokenArray.length;
    double similarity = similarities/(Math.log(numWordsInSentence1)+
    Math.log(numWordsInSentence2));
    return similarity;
}
private double calculateScore(SentenceVertex vi){
    double scorei = (1.0 - PROBABILITY);
    double sum = 0;
    //Iterate over edges of vi
    for(DefaultWeightedEdge eij: graph.edgesOf(vi)){
        double numerator = graph.getEdgeWeight(eij);
        //Get other vertex
```

```
            SentenceVertex vj = graph.getEdgeSource(eij);
            if(vi == vj){
                vj = graph.getEdgeTarget(eij);
            }
            double denominator = 0;
            for(DefaultWeightedEdge ejk: graph.edgesOf(vj)){
                denominator += graph.getEdgeWeight(ejk);
            }
            double scorej = vj.getScore();
            sum += (numerator/denominator)*scorej;
        }
        scorei += PROBABILITY *sum;
        return scorei;
    }
    private void convergeScores(){
        double error = 1;
        int iterations = 0;
        while(error > CONVERGENCE_THRESHOLD){
            for(SentenceVertex v : graph.vertexSet()){
                double newScore = calculateScore(v);
                double lastScore = v.getScore();
                double scoreError = Math.abs(lastScore - newScore)/newScore;
                error += scoreError;
                v.setScore(newScore);
            }
            error = error/(double)(graph.vertexSet().size());
            iterations +=1;
        }
        Log.v("TextRank", iterations + "");
    }
    public ArrayList<SentenceVertex> sentenceExtraction(String text){
        createGraph(text);
```

```
        convergeScores();
        ArrayList<SentenceVertex> sorted = new ArrayList<SentenceVertex>();
        sorted.addAll(graph.vertexSet());
        Collections.sort(sorted, new SentenceVertexComparator());
        return sorted;
    }
    private void createTokenGraph(String article){
        //Remove all punctuation and lowercase for tokenization
        article = article.replaceAll("\\p{P}", "").toLowerCase();
        tokenGraph =  new SimpleGraph<TokenVertex, DefaultEdge>(DefaultEdge.class);
        String[] tokens = tokenizer.tokenize(article);
        ArrayList<String> tokensWithoutStopWords = new ArrayList<String>();
        for(int i = 0; i < tokens.length; i ++){
            String curToken = tokens[i];
            if(!extendedStopwords.contains(curToken)){
                tokensWithoutStopWords.add(curToken);
            }
        }
        //HashSet optimizes removing duplicates
        HashSet<String> lhsTokens = new HashSet<String>(tokensWithoutStopWords);
        //Map String values to their vertices
        HashMap<String, TokenVertex> tokenVertices = new HashMap<String,TokenVertex>();
        //Add each vertex
        for(String token: lhsTokens){
            TokenVertex v = new TokenVertex(token);
            tokenVertices.put(token, v);
            tokenGraph.addVertex(v);
        }
        //Add edges between words within a certain window
        for(int i = 0; i < tokensWithoutStopWords.size() - COOCCURENCE_WINDOW; i++){
            String[] window = new String[COOCCURENCE_WINDOW];
            for(int j = 0; j < COOCCURENCE_WINDOW; j++){
```

```
            String curToken = tokensWithoutStopWords.get(i+j);
            window[j] = curToken;
            for(int k = 0; k < j; k++){
                String other = window[k];
                TokenVertex otherVertex = tokenVertices.get(other);
                TokenVertex curVertex = tokenVertices.get(curToken);
                if (curVertex != otherVertex && !tokenGraph.containsEdge
                (curVertex,otherVertex))
                    tokenGraph.addEdge(curVertex, otherVertex);
            }
        }
    }
}
private double calculateTokenScore(TokenVertex vi){
    double scorei = (1.0 - PROBABILITY);
    double sum = 0;
    //Iterate over edges of vi
    for(DefaultEdge eij: tokenGraph.edgesOf(vi)){
        double numerator = 1.0;
        //Get other vertex
        TokenVertex vj = tokenGraph.getEdgeSource(eij);
        if(vi == vj){
            vj = tokenGraph.getEdgeTarget(eij);
        }
        //Find the denominator
        double denominator = tokenGraph.edgesOf(vj).size();
        double scorej = vj.getScore();
        sum += (numerator/denominator)*scorej;
    }
    scorei += PROBABILITY *sum;
    return scorei;
}
```

```
private void convergeTokenScores(){
    double error = 1;
    int iterations = 0;
    while(error > CONVERGENCE_THRESHOLD){
        for(TokenVertex v : tokenGraph.vertexSet()){
            double newScore = calculateTokenScore(v);
            double lastScore = v.getScore();
            double scoreError = Math.abs(lastScore - newScore)/newScore;
            error += scoreError;
            v.setScore(newScore);
        }
        error = error/(double)(tokenGraph.vertexSet().size());
        iterations +=1;
    }
    Log.v("TextRank", iterations+"");
}
public ArrayList<TokenVertex> keywordExtraction(String text){
    createTokenGraph(text);
    convergeTokenScores();
    ArrayList<TokenVertex> sorted = new ArrayList<TokenVertex>();
    sorted.addAll(tokenGraph.vertexSet());
    Collections.sort(sorted, new TokenVertexComparator());
    return sorted;
}
private static class TokenVertexComparator implements Comparator<TokenVertex> {
    @Override
    public int compare(TokenVertex lhs, TokenVertex rhs) {
        return Double.compare(rhs.getScore(), lhs.getScore());
    }
}
private static class SentenceVertexComparator implements Comparator<SentenceVertex>
{
```

```
    @Override
    public int compare(SentenceVertex lhs, SentenceVertex rhs) {
        return Double.compare(rhs.getScore(), lhs.getScore());
    }
}
public class TokenVertex {
    private String token;
    private double score;
    public TokenVertex(String s){
        token = s;
        score = 1.0;
    }
    public String getToken() {
        return token;
    }
    public double getScore() {
        return score;
    }
    public void setScore(double score) {
        this.score = score;
   }
}
public class SentenceVertex {
    private String sentence;
    private String[] tokens;
    private HashSet<String> tokenHashSet;
    private double score;
    public SentenceVertex(String s, String[] t){
        sentence = s;
        tokens = t;
        tokenHashSet = new HashSet<String>(Arrays.asList(t));
        score = 1.0;
```

```
        }
        public String getSentence() {
            return sentence;
        }
        public String[] getTokens() {
            return tokens;
        }
        public HashSet<String> getTokensAsHashSet() {
            return tokenHashSet;
        }
        public double getScore() {
            return score;
        }
        public void setScore(double score) {
            this.score = score;
        }
    }

}
```

Home page XML:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="website.group4.group4.summarizerx.MainActivityFragment"
    tools:showIn="@layout/activity_main"
    android:background="#37474f"
    android:theme="@style/AppTheme">
    <LinearLayout
```

```
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/parent_linear_layout"
android:elevation="10dp"
android:background="@color/colorBackground">

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAlignment="center"
    android:text=" Article / Paragraph / Image "
    android:id="@+id/textView1"
    android:layout_below="@+id/linearLayout"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:padding="10dp"
    android:password="false"
    android:singleLine="false"
    android:typeface="sans"
    android:textStyle="bold"
    android:textSize="20sp"
    android:capitalize="characters"
    android:textAppearance="@android:style/TextAppearance.DeviceDefault.
    Widget.ActionBar.Title"
    android:theme="@style/AppTheme"
    android:textColor="#004c4c"
    android:background="@color/colorBackground"/>
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
android:weightSum="1"
android:id="@+id/linearLayout"
android:padding="10dp">

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0.2"
    android:id="@+id/url_bar"
    android:imeOptions="actionSearch"
    android:inputType="text"
    android:autoText="false"
    android:autoLink="web"
    android:hint="@string/edit_url_hint"
    android:elegantTextHeight="true"
    android:enabled="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_centerHorizontal="true"
    android:textSize="16dp"
    android:textColor="#f5f5f5"
    android:textColorHint="#b2d8d8" />

<Button
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:drawableRight="@drawable/ic_navigate_next"
    android:layout_weight="0.8"
    android:id="@+id/button"
    android:text="Go"
    android:onClick="handleButtonClick"
    android:singleLine="false"
```

```
        android:textSize="16dp"
        style="@style/Base.TextAppearance.AppCompat.Button"
        android:allowUndo="true"
        android:background="@drawable/mybutton"
        android:textColor="#004c4c" />
</LinearLayout>
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:weightSum="1"
    android:id="@+id/linearLayout1"
    android:padding="10dp">
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.2"
        android:id="@+id/url_bar1"
        android:imeOptions="actionSearch"
        android:inputType="textMultiLine"
        android:autoText="false"
        android:autoLink="web"
        android:hint="Enter text here"
        android:elegantTextHeight="true"
        android:enabled="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_centerHorizontal="true"
        android:textSize="16dp"
        android:textColor="#f5f5f5"
        android:textColorHint="#b2d8d8" />
    <Button
```

```
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:drawableRight="@drawable/ic_navigate_next"
        android:layout_weight="0.8"
        android:id="@+id/button1"
        android:text="Go"
        android:onClick="handleButtonClick1"
        android:singleLine="false"
        android:textSize="16dp"
        style="@style/Base.TextAppearance.AppCompat.Button"
        android:allowUndo="true"
        android:background="@drawable/mybutton"
        android:textColor="#004c4c"/>
</LinearLayout>
<Button
    android:id="@+id/button2"
    style="@style/Widget.AppCompat.Button.Colored"
    android:layout_width="192dp"
    android:layout_height="match_parent"
    android:layout_weight="0.8"
    android:allowUndo="true"
    android:drawableRight="@drawable/ic_navigate_next"
    android:gravity="center"
    android:layout_gravity="center_horizontal"
    android:onClick="handleButtonClick2"
    android:singleLine="false"
    android:text="Lens"
    android:textSize="16dp"
    android:background="@drawable/mybutton"
    android:textColor="#004c4c" />
<TextView
    android:layout_width="fill_parent"
```

```
            android:layout_height="wrap_content"
            android:text="@string/summaries_title"
            android:id="@+id/textView"
            android:layout_below="@+id/linearLayout"
            android:layout_alignParentLeft="true"
            android:layout_alignParentStart="true"
            android:textAlignment="center"
            android:padding="20dp"
            android:password="false"
            android:singleLine="false"
            android:typeface="sans"
            android:textStyle="bold"
            android:textSize="20sp"
            android:capitalize="characters"
            android:textAppearance="@android:style/TextAppearance.DeviceDefault
            .Widget.ActionBar.Title"
            android:theme="@style/AppTheme"
            android:textColor="#004c4c"
            android:background="@color/colorBackground"/>
    </LinearLayout>
    <ListView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_below="@+id/parent_linear_layout"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:background="#99b7b7" />
</RelativeLayout>
```

Article summarization page Java:

```
package website.group4.group4.summarizerx;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.DialogFragment;
import android.app.SearchManager;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.PorterDuff;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.View;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.ProgressBar;
import android.widget.TextView;
import org.apmem.tools.layouts.FlowLayout;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
```

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.util.ArrayList;
import textrank.TextRank;
public class ArticleActivity extends AppCompatActivity {
    private final String TAG = "ArticleActivity";
    private WebView webview;
    private TextView authorText;
    private TextView summaryText;
    private ProgressBar progressBar;
    FlowLayout keywordsContainer;
    private Document document;
    private String summary;
    private String author;
    private String headline;
    private String[] keywords;
    private String selectedKeyword;
    private String url;
    private static TextRank tr;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_article);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        webview = (WebView)findViewById(R.id.webView);
```

```
    summaryText = (TextView)findViewById(R.id.summary);
    authorText = (TextView) findViewById(R.id.authors);
    keywordsContainer = (FlowLayout)findViewById(R.id.keywords);
    progressBar = (ProgressBar) findViewById(R.id.progressBar);
    initializeTextRank();
    initializeWebView();
}
private void runAnalysis(){
    Element article = getArticleFromDocument();
    if(article!=null){
        extractAuthor(article);
        extractHeadline(article);
        summarizeArticle(article);
    }
    else{
        summary = "No article could be extracted.";
        author = "";
    }
    progressBar.setVisibility(View.GONE);
    setProgress(100);

    showResults();
}
private void showResults(){
    //Show results in-app
    summaryText.setText(summary);
    authorText.setText(author);
    keywordsContainer.removeAllViews();
    if(keywords != null) {
        for (final String s : keywords) {
            TextView newButton = new Button(this);
            newButton.setText(s);
```

```
            newButton.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    ArticleActivity.this.selectedKeyword = s;
                    DialogFragment newFragment = new ResearchDialogFragment();
                    newFragment.show(getFragmentManager(), "research");
                }
            });
            newButton.getBackground().setColorFilter(getResources()
            .getColor(R.color.colorAccent), PorterDuff.Mode.MULTIPLY);
            keywordsContainer.addView(newButton);
        }
    }
}
private void initializeTextRank(){
    //Open raw resources to initialize OpenNLP tools for TextRank
    if(tr==null){
        InputStream sent = getResources().openRawResource(R.raw.en_sent);
        InputStream token = getResources().openRawResource(R.raw.en_token);
        InputStream stop = getResources().openRawResource(R.raw.stopwords);
        InputStream exstop = getResources().openRawResource(R.raw.
        extended_stopwords);
        try {
            tr = new TextRank(sent, token, stop, exstop);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
private void initializeWebView(){
    Intent intent = getIntent();
    Uri data = intent.getData();
    String url;
```

```
    //If there's data, the intent was from outside of the app
    if(data!=null) {
        String host = data.getHost();
        String path = data.getPath();
        url = host+path;
    }
    //Otherwise it was initialized from the MainActivity
    else {
        url = intent.getStringExtra("url");
    }
    //In case users didn't include http:// or https://
    if(url.indexOf("http://") == -1 && url.indexOf("https://") == -1){
        url = "http://"+url;
    }
    webview.setWebViewClient(new WebViewClient() {
        @Override
        public void onPageStarted(WebView view, String url, Bitmap favicon){
            progressBar.setVisibility(View.VISIBLE);
            setProgress(0);
            FetchPageTask fetch = new FetchPageTask();
            ArticleActivity.this.url = url;
            fetch.execute(url);
        }
        @Override
        public void onPageFinished(WebView view, String url) {
        }

    });
    webview.loadUrl(url);
}
private class FetchPageTask extends AsyncTask<String, Void, Document> {
```

```
    @Override
    protected Document doInBackground(String... url) {
        Document doc = null;
        try {
            Log.v("JSoup", "Attempting to connect to "+ url[0]);
            doc = Jsoup.connect(url[0]).get();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return doc;
    }

    protected void onPostExecute(Document doc) {
        //New page, clear out past keywords
        keywords=null;
        //Call jsoupAnalysis
        if(doc!=null) {
            Log.v(TAG, "Loaded the page.");
            ArticleActivity.this.document = doc;
            //Only run analysis on successful document load
            runAnalysis();
        }
        else{
            Log.v(TAG, "Couldn't load page.");
            summary = "Page couldn't be loaded for info extraction.";
        }
    }
}


private Element getArticleFromDocument(){
    //Start specific. Some websites have smaller psuedo articles at the top of
```

```
    the page
    Element article = document.select("main").select("article").first();
    if(article == null)
        article = document.select("article").first();
    if(article == null)
        article = document.getElementsByAttributeValueContaining
        ("class", "article").first();
    if(article == null)
        article = document.getElementsByAttributeValueContaining
        ("class", "story").first();
    return article;
}

private void extractAuthor(Element article){
    Element authorContainer = article.getElementsByAttributeValueContaining
    ("class", "byline").first();
    if(authorContainer == null) {
        authorContainer = article.getElementsByAttributeValueContaining
        ("class", "author").first();
    }
    if(authorContainer == null) {
        authorContainer = document.getElementsByAttributeValueContaining
        ("class", "byline").first();
    }
    if(authorContainer == null) {
        authorContainer = document.getElementsByAttributeValueContaining
        ("class", "author").first();
    }
    if(authorContainer!=null) {
        Log.v("Author Information", authorContainer.text());
        author = authorContainer.text();
    }
```

```
    else {
        Log.v("Author Information", "Couldn't extract author data");
        author = "Couldn't extract author data.";
    }
}
private void extractHeadline(Element article){
    Element headlineContainer = article.getElementsByAttributeValueContaining
    ("class", "headline").first();
    if(headlineContainer == null) {
        headlineContainer = article.getElementsByAttributeValueContaining
        ("class", "title").first();
    }
    if(headlineContainer == null) {
        headlineContainer = document.getElementsByAttributeValueContaining
        ("class", "headline").first();
    }
    if(headlineContainer == null) {
        headlineContainer = document.getElementsByAttributeValueContaining
        ("class", "title").first();
    }
    if(headlineContainer!=null) {
        Log.v("Headline Information", headlineContainer.text());
        headline = headlineContainer.text();
    }
    else {
        Log.v("Headline Information", "Couldn't extract headline data");
        headline = "Couldn't extract headline data";
    }
}
private void summarizeArticle(Element article){
    String articleText = createArticleText(article);
    TextView summaryText = (TextView)findViewById(R.id.summary);
```

```
    if(articleText != null && articleText != ""){
        ArrayList<TextRank.SentenceVertex> rankedSentences =
        tr.sentenceExtraction(articleText);
        ArrayList<TextRank.TokenVertex> rankedTokens = tr.keywordExtraction
        (articleText);
        //Get summary
        summary = rankedSentences.get(0).getSentence();
        for(int i = 0; i < 5 && i < rankedSentences.size(); i ++){
            Log.v("Ranked Sentence #" + (i+1), rankedSentences.get(i).
            getSentence());
        }

        //Get best 8 keywords
        keywords = new String[8];
        for(int i = 0; i < rankedTokens.size() && i < keywords.length; i++){
            TextRank.TokenVertex tv = rankedTokens.get(i);
            keywords[i] = tv.getToken();
        }

        //Save to cache
        try {
            saveSummaryToCache();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    else{
        summary = "Unable to process this article";
    }
}
private void saveSummaryToCache() throws IOException {
    File cachedSummaries= new File(getCacheDir(), "summaries.txt");
```

```
    //Open PrintWriter in append mode.
    PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter
    (cachedSummaries, true)));
    pw.println(headline);
    pw.println(summary);
    pw.println(author);
    pw.println(url);
    pw.close();
}
private String createArticleText(Element article){
    String articleText = "";
    Elements articleParagraphs = article.select("p");
    //If the article <p> elements don't make up most of the text in the article
    if((double)article.text().length()*0.4 >= articleParagraphs.text().length()){
        //Hope that they're under the tag paragraph (prime example, CNN,
        who just changed their article HTML layout this week)
        articleParagraphs = article.getElementsByAttributeValueContaining
        ("class", "paragraph");
    }
    //Process each paragraph
    for(int i = 0; i < articleParagraphs.size(); i++){
        Element paragraph = articleParagraphs.get(i);
        //End all sentences with periods so as to ensure sentence separation
        //Otherwise, sentences will appear concatenated
        String pText = paragraph.text();
        if(pText.length()>0 && pText.charAt
        (pText.length()-1) != ' '){
            if(pText.charAt(pText.length()-1)
            == '.' || pText.charAt(pText.length()-1)=='"')
                pText += " ";
            else
                pText += ". ";
```

```
        }
        articleText += pText;
    }
    return articleText;
}

@SuppressLint("ValidFragment")
private class ResearchDialogFragment extends DialogFragment {
    private int selectedItem;
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder
        (getActivity(), AlertDialog.THEME_DEVICE_DEFAULT_DARK);
        builder.setTitle(R.string.research_dialog_title)
                .setSingleChoiceItems(R.array.research_options, 0,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which)
                            {
                                selectedItem = which;
                            }
                        })
                .setPositiveButton(R.string.research_dialog_positive,
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        switch (selectedItem){
                            case 0: {
                                Intent i = new Intent(Intent.ACTION_VIEW);
                                i.setData(Uri.parse("https://www.google.com/search?
                                aq=f&hl=en&gl=us&tbm=nws&btnmeta_news_search=1&q=
                                " + ArticleActivity.this.selectedKeyword));
```

```
                startActivity(i);
            } break;

            case 1: {
                Intent i = new Intent(Intent.ACTION_WEB_SEARCH);
                i.putExtra(SearchManager.QUERY, ArticleActivity.
                this.
                selectedKeyword);
                startActivity(i);
            } break;

            case 2: {
                Intent i = new Intent(Intent.ACTION_VIEW);
                i.setData(Uri.parse("https://en.wikipedia.
                org/wiki/" + ArticleActivity.this.
                selectedKeyword));
                startActivity(i);
            } break;

            case 3: {
                Intent i = new Intent(Intent.ACTION_VIEW);
                i.setData(Uri.parse("http://www.imdb.com/find?q="
                + ArticleActivity.this.selectedKeyword +
                "&s=all"));
                startActivity(i);
            } break;
        }
    }
})
.setNegativeButton(R.string.research_dialog_negative, new
DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
```

```
                            // User cancelled the dialog
                        }
                    });
            // Create the AlertDialog object and return it
            return builder.create();
        }
    }
}
```

Article summarization page Xml:

```
<com.sothree.slidinguppanel.SlidingUpPanelLayout xmlns:android="http://schemas.android.c
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:sothree="http://schemas.android.com/apk/res-auto"
    android:id="@+id/sliding_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="website.group4.group4.summarizerx. ArticleActivityFragment"
    android:orientation="vertical"
    android:gravity="bottom"
    sothree:umanoPanelHeight="100dp"
    sothree:umanoShadowHeight="8dp"
    android:background="@color/colorBackground"
    android:theme="@style/Base.ThemeOverlay.AppCompat.Dark">

    <WebView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/webView"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
```

```
    android:layout_alignParentStart="true" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="0.75"
    android:orientation="vertical"
    android:padding="12dp"
    android:id="@+id/swipeableStats">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:weightSum="5">

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="2.5"
            android:textAppearance="?android: attr/textAppearanceLarge"
            android:text="Summary:"
            android:id="@+id/summaryLabel" />

        <ProgressBar
            android:layout_gravity="center_vertical|top"
            android:id="@+id/progressBar"
            android:layout_width="30dp"
            android:layout_height="30dp" />
    </LinearLayout>

    <TextView
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android: attr/textAppearanceSmall"
    android:id="@+id/summary"
    android:padding="5dp"
    android:autoText="false"/>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="@android:style /TextAppearance.DeviceDefault.Large"
    android:text="Keywords:"
    android:id="@+id/keywordsLabel" />

<org.apmem.tools.layouts.FlowLayout
    xmlns:android="http://schemas.android.com /apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/keywords">
    >
</org.apmem.tools.layouts.FlowLayout>

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="@android:style
    /TextAppearance.DeviceDefault.Large"
    android:text="Authors:"
    android:id="@+id/authorsLabel" />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

```
            android:textAppearance="?android:attr
            /textAppearanceSmall"
            android:id="@+id/authors"
            android:padding="5dp"
            android:autoText="false" />
    </LinearLayout>
</com.sothree.slidinguppanel.SlidingUpPanelLayout>
```

CHAPTER 5

CONCLUSION

The system has developed an automatic text summarization system using natural language processing technique i.e. semantic chaining in combination with feature extraction using fuzzy logic and handling of correlation of sentences. It has used a dataset of 250 small sized documents and 250 medium-sized documents in domain of sports and technical from BBC news corpus for testing the performance of the implemented summarizer. TextRank uses the intuition behind the PageRank algorithm to rank sentences TextRank is an unsupervised method for computing the extractive summary of a text. No training is necessary. Given that it's language independent, can be used on any language, not only English. In this article, we used as the similarity measure, the cosine similarity. We convert sentences to vectors to compute this similarity.

REFERENCES

[1] Rada Mihalcea and Paul Tarau, "Text-rank: Bringing Order into Texts," Proceeding of he Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 2015. accessed on february 13, 2019 07:00 pm

[2] M. S. Patil, M. S. Bewoor, S. H. Patil "A Hybrid Approach for Extractive Document Summarization Using Machine Learning and Clustering Technique", International Journal of Computer Science and Information Technologies, Vol. 5, IssueNo. 2, ISSN: 0975-9646, pp.1584-1586, 2019, accessed on March 13, 2019 07:20 pm.

[3] RasimAlguliev, RamizAliguliyev, "Evolutionary Algorithm for Extractive Text Summarization." Intelligent Information Management, 1, pp. 128-138, November 2016, April 13, 2019 06:20 am.

[4] Naresh Kumar Nagwani, Dr. Shrish Verma Associate "A Frequent Term and Semantic Similarity based Single Document Text Summarization Algorithm", International Journal

ofComputer Applications(0975- 8887) Volume 17- No.2, March 2015, March 15, 2019 07:20 pm

[5] A. Nenkova, and K. McKeown, "Automatic summarization,". Foundations and Trends in Information Retrieval, 5(2-3):103–233, 2017, March 19, 2019 02:20 pm

[6] K. Sparck Jones, "Automatic summarizing: factors and directions,". Advances in Automatic Text Summarization, pp. 1–12, MIT Press, 2018.

[7] R. Mihalcea, and P. Tarau, "Textrank: Bringing order into texts,". In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004, March 29, 2019 07:20 pm

[8] G. Erkan, and D. R. Radev, "LexRank: graph-based lexical centrality as salience in text summarization,". Journal of Artificial Intelligence Research, v.22 n.1, p.457-479, July 2018 March 03, 2019 13:20 pm.

[9] J. Jagadeesh, P. Pingali, and V. Varma, "Sentence Extraction Based Single Document Summarization", Workshop on Document Summarization, 19th and 20th March, 2016, March 29, 2019 01:20 pm.

[10] A. Nenkova and K. McKeown, Automatic Summarization, Foundation and Trends R in Information Retrieval, vol 5, nos 2–3, pp 103–233, 2011.

[11] Juan-Manuel Torres-Moreno, Automatic Text Summarization, John Wiley Sons, page 38.

[12] https://towardsdatascience.com/textrank-for-keyword-extraction-by-python-c0bae21bcec0 accessed on 28th November 2018 at 7:30 PM.

[13] https://www.link-assistant.com/news/google-page-rank-2019.html accessed on 11th January 2019 at 9:20 PM.