

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology (Deemed to be University)

Submitted in partial fulfillment of the requirements for the award
of Bachelor of Technology Degree in

Information Technology

By

Aswin J Richards
(Reg. No. 37120011)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119



SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JUNE 2019

(Established under Section 3 of UGC Act, 1956)

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai -

600119 HYPERLINK "<http://www.sathyabama.ac.in/>" \www.sathyabama.ac.in

SCHOOL OF COMPUTING



BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Aswin J Richards** (Reg.no. 37120011) who carried out the project entitled
“**Multilingual Automatic Extractive Text Summarizer**”
under our supervision from Jan 2019 to April 2019.

Internal Guide

Dr.S.REVATHY., M.E., Ph.D.,

Head of the Department

Dr. R.SUBHASHINI.,M.E., Ph.D.,

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **ASWIN.J.RICHARDS(Reg. No. 37120011)** hereby declare that the Project Report entitled **“Multilingual Automatic Extractive Text Summarizer”** done by me under the guidance of **Dr.S.REVATHY., M.E., Ph.D.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.SASIKALA M.E.,Ph.D., Dean**, School of Computing , **Dr. R.SUBHASHINI M.E.,Ph.D.,** Head of the Department of Information and Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S.REVATHY M.E.,Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-Teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of my project.

TRAINING CERTIFICATE

CERTIFICATE OF COMPLETION



Aswin J Richards has completed Python internship for a duration of two months in AR Projects Ltd.

20-12-2020

Date

Director
AR PROJECTS

Signature

ONLINE CERTIFICATION

Certificate of Completion

This is to certify that **Aswin.J.Richards** successfully completed 33.5 total hours of **The Complete Python Masterclass: Learn Python From Scratch** online course on Jan. 19, 2021

Ashutosh Pawar
Ashutosh Pawar, Instructor

&
 **Udemy**

Certificate no: UC-fac5f87f-ebda-49ca-8e84-95d2c9c8b3da
Certificate url: [ude.my/UC-fac5f87f-ebda-49ca-8e84-95d2c9c8b3da](https://udemy.com/UC-fac5f87f-ebda-49ca-8e84-95d2c9c8b3da)
Version 3

#BeAble

ABSTRACT

This project “**Multilingual Automatic Extractive Text Summarizer**” deals with the Natural Language Processing (NLP) which summarizes the large document into a shorter version of the text. This automatic extractive text summarization is done by using Extraction based text summarization which involves selecting sentences of high relevance (rank) from the document based on word and sentence features and put them together to generate a summary. It summarizes the text faster and effectively as this algorithm uses Graph-based sentences ranking. In this new era, where remarkable information exists on the Internet, it is most important to provide an improved mechanism to extract the information quickly and most efficiently.

There is a plenty of text material available on the Internet. So there is a setback of searching for significant documents from the number of documents available and absorbing appropriate information from it. To solve the above two problems, the text summarization is very much necessary. This summarizer can reduce the paperwork and time, as the summarized text is converted into audio format and even share the summary as they wish by just one tap.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF TABLES	vii
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1.	INTRODUCTION	1
	1.1 Project Overview	1
	1.2 Objective	1
	1.3 Natural Language Processing	2
	1.4 Text Summarization	2
	1.5 Extractive Summarization	3
	1.5.1 Unsupervised Learning	4
	1.5.2 Text Rank Algorithm	4
2.	SYSTEM REQUIREMENTS AND SPECIFICATION	18
	2.1 Functional Requirements	18
	2.2 Non Functional Requirements	18
	2.3 System Requirements	19
	2.3.1. Hardware Requirements	19
	2.3.2. Software Requirements	20

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	iii
	LIST OF TABLES	vii
	LIST OF FIGS	viii
	LIST OF ABBREVIATIONS	ix
3.	SYSTEM DESIGN	21
	3.1 Methodology	21
	3.1.1 GUI	21
	3.1.2 Text Pre-processing	21
	3.1.3 Summarization	22
	3.1.4 Word Frequency	22
	Algorithm	23
	3.1.5 Text Rank Algorithm	
	3.1.6 Audio and QR Code	24
	Generator	25
	3.2 Flow Diagram	
4.	SYSTEM IMPLEMENTATION	26
	4.1 Software Tools	26
	4.1.1. Python Programming	26
	4.1.2. Python IDLE	26
	4.2 Package Requirements	27
	4.3 GUI – Application Sample	28
	Code	
	4.4 Function For Summarization	30
	4.5 Functions For Converting	33
	Text To Speech	

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
4.6	Function For Converting Text To QR Code	34 35
4.7	Function For Clearing Results And To Reset The Scrolled Text	
5.	CONCLUSION AND FUTURE ENHANCEMENTS	39 41
	APPENDIX 1	46
	REFERENCES	

LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
1.1	Process Flow for Extractive Summarization	3
1.2	Process Flow Of Text Rank Algorithm	5
1.3	Sentence Vector Graph	6
4.1	Text Pre processing	21
4.2	Flow Chart	25
6.1	Rouge-N Values	38
A.1	Home Tab	41
A.2	Customized Summarization Tab	41
A.3	Basic Algo	42
A.4	About Tab	42
A.5	Summary-Text Rank	43
A.6	QR Code Generation	43
A.7	Extracted Keywords	44
A.8	Summary-Ratio	44
A.9	Summary-Word Count	45
A.10	Summary-Word Frequency	45

LIST OF ABBREVIATIONS

ACRONYM	EXPANSION
NLP	Natural Language Processing
GUI	Graphical User Interface
AI	Artificial Intelligence
RAM	Random Access Memory
ROM	Read Only Memory
QR	Quick Response
IDLE	Integrated Development and Learning Environment
IDE	Integrated Development Environment
NLTK	Natural Language Toolkit
GTTS	Google Text To Speech
TK	TKinter
LCS	Longest Common Subsequence
DSC	Dice Similarity Coefficient

CHAPTER 1

INTRODUCTION

1.1. PROJECT OVERVIEW

A Multilingual Automatic extractive text summarizer is the GUI based text summarization that uses the Text Rank algorithm to summarize the massive texts into a straight meaningful summary for multiple languages. The significance of Natural Language Processing (NLP) is ruling the AI industry for making humans to perform complex tasks with ease. As humans should not waste much time in reading the entire large text to recognize the subject of the larger document, just by one tap they could save their valuable time. In today's world, the Internet is the platform where all the study materials, news articles, even textbooks, etc., are searched. But the problem arises when grasping relevant information from the massive materials on the internet. So this application will provide the best-consolidated results will be given as a summary and extracts keywords for making humans understand the shortest version of the articles or text that describes the subject. This summarizer automatically summarizes the larger texts of any language and converted into audio for reducing the time for reading the subject. And also the summary can be easily shared by anyone by just scanning the QR code of the summary. The Complete application is shown in appendix.

1.2. OBJECTIVE

- To provide an effective summary of the large document automatically.
- To reduce reading time and paperwork.
- To select relevant informative subjects precisely in the research process.
- To generate summary automatically and faster than human understandings.
- To extract the keywords of the large passage for making summarization even more relevant to the subject.
- To share the summarized study-material of relevant topics in textbooks via QR code scanning of the summarized text to make it useful for others.

1.3. NATURAL LANGUAGE PROCESSING (NLP)

Natural Language Processing is the process of building machines to understand, to read and to derive the meaning of the human language and making responses. It is a discipline that focuses on the interaction between data science and human language. NLP is a subfield of AI. In other words NLP is a process of programming machines to analyze and process huge amounts of natural language data.

1.4. TEXT SUMMARIZATION

Text summarization is the process of creating a short, precise and fluent summary of a longer text document. Automatic text summarization methods are very much needed to deal with the huge amount of text data existing online to get through relevant information quicker.

Text Summarization has two approaches namely Extractive Summarization and 2. Abstractive Summarization. The former extracts the words, phrases and sentences from the original text having high scorings based on algorithms. An abstractive based summarization produce a human-like summary, in which the meaning will be extracted and sentences will be reframed according to the algorithms. In this project we use an extractive method to summarize the huge texts automatically.

The extractive method was used since it can perform faster than abstractive methods and also less complex when compared to abstractive methods as it uses statistical approaches.

1.5. EXTRACTIVE SUMMARIZATION

Extractive summarization means extracting key sections of the passage and generating them a subset of the sentences from the original passage. When compared to automatic abstractive summaries, the Extractive method generates better results. This is because of the actuality that abstractive summarization methods deal with troubles such as semantic representation, inference and also natural language generation which is comparatively harder than data-driven approaches such as sentence extraction.

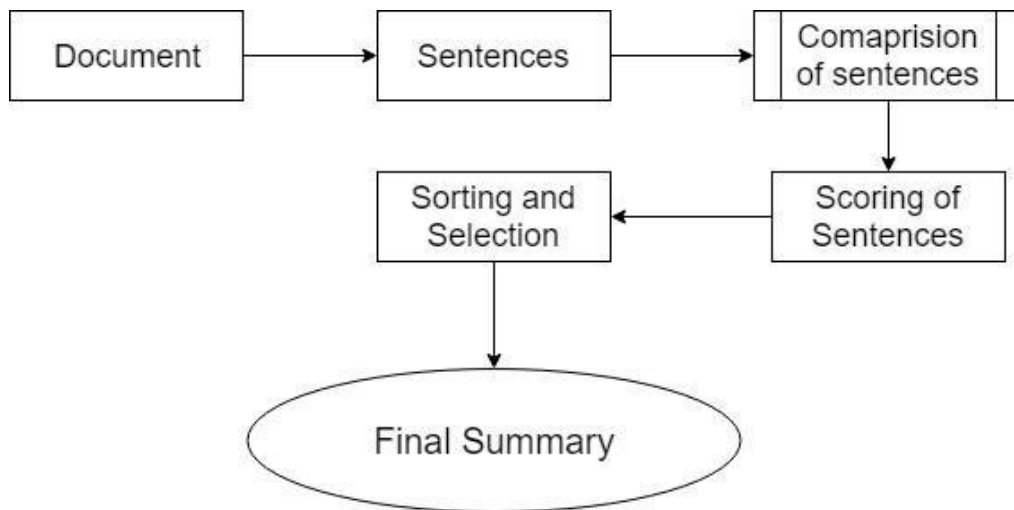


Fig:1.1 : Process Flow for Extractive Summarization

A General extractive method- word frequency algorithm follows the following approach,

1. Loading the input text.
2. Tokenizing the sentences and words from the text.
3. Remove Stop words from the text and lemmatizing the words.
4. Scoring sentences based on algorithm.
5. Ranking Sentences based on scores.
6. Joining the ordered sentences as a summary

1.5.1. UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task which tries to infer a function to depict hidden structure from unlabeled data. As the examples prearranged to the learner are unlabeled, there is no error or reward signal to estimate a probable solution – this differentiates unsupervised learning from supervised learning and also reinforcement learning. Unsupervised learning can also be associated with the summarization of text as the word vectors can be clustered and the key features can be easily extracted.

In this project, we use Text Rank Algorithm for the unsupervised Graph-based methodology to summarize the text.

1.5.2. TEXT RANK ALGORITHM

Text Rank is an unsupervised algorithm for summarizing the texts automatically that can be used to attain the most significant keywords in a document. This algorithm is a base of Page Rank over a graph constructed particularly for the summarization task. The only difference is what page rank uses ranking of pager while text rank used to rank sentences by using symmetrical graphs. This uses the ranking of the elements in the graph: the most essential elements are the ones that better describe the text. Text Rank used to build summaries without the need of a training corpus and allows summarizing different languages.

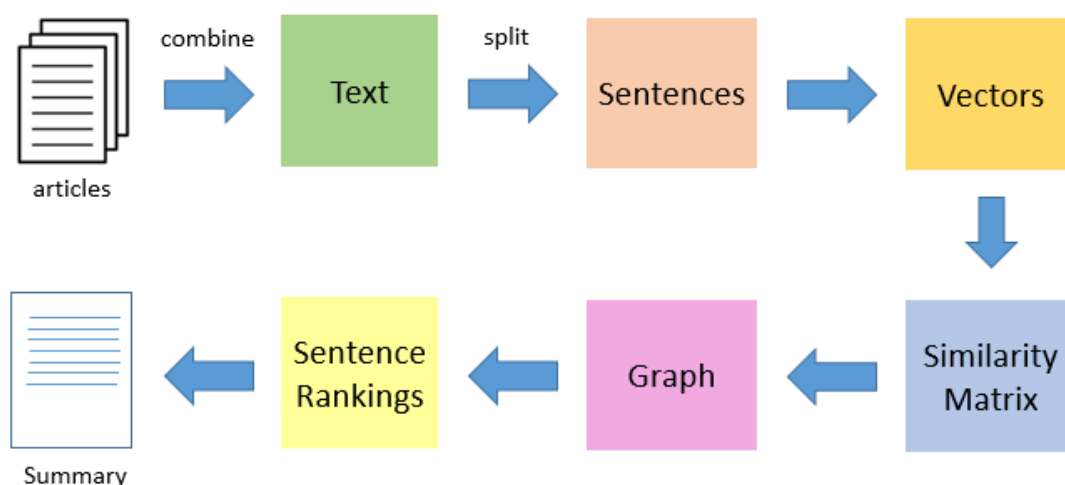


Fig: 1.2 : Process Flow of Text Rank Algorithm

1. Concatenating the text available in the article.
2. Then splitting the text into individual sentences.
3. Finding vector representation for every sentence.
4. Finding the similarities between sentence vectors are then calculated and stored as a matrix.
5. The similarity matrix is then converted into a graph, with vertices as sentences and edge as a similarity scores , for ranking sentence calculation.

6. The original Page Rank definition for graph-based ranking is assuming un-weighted graphs. However, in our model, the graphs are build from natural language texts, and may include multiple or partial links between the units (vertices) that are extracted from the text. It may be therefore useful to indicate and incorporate into the model the “strength” of the connection between two vertices and as a weight added to the corresponding edge that connects the two vertices.

7. A graph is then generated from this cosine similarity matrix. We will then apply the Page Rank ranking algorithm to the graph to calculate scores for each sentence.

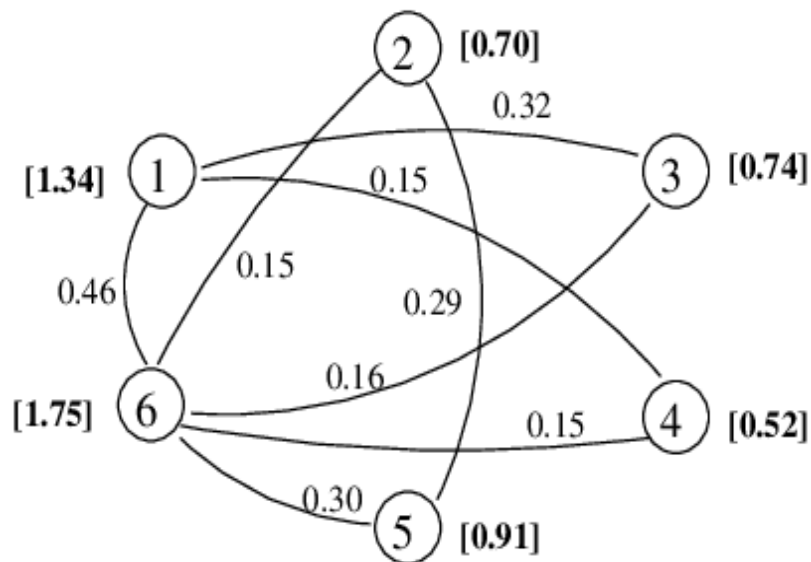


Fig: 1.3 : Sentence Vector Graph

8. In the case of Text Rank, we generate a cosine similarity matrix where we have the similarity of each sentence to each other. A graph is then generated from this cosine similarity matrix.

9. We will then apply the Page Rank ranking algorithm to the graph to calculate scores for each sentence.
10. With the help of graph the sentence scores will be obtained and join the top-ranked sentences which forms the final summary. Text Rank algorithm is explained in chapter 3.1.5.

CHAPTER 2

SYSTEM REQUIREMENTS AND SPECIFICATION

System requirement specification gives the complete description of the behavior about the system developed by this project. This includes specification of functional and nonfunctional requirements of the application.

The interaction of the users with the application is represented with the help of use cases and their analysis. This also includes the description about feasibility, risk analysis and external interface requirements to accomplish this project.

2.1 FUNCTIONAL REQUIREMENTS

The functional requirements will describe the features and functionality of the system. Functional requirements record the operation that must be done. Functional requirements are based for non-functional requirements.

2.2 NON FUNCTIONAL REQUIREMENTS

The non-functional requirements define how the system will do certain operation. Non-functional requirements are usually called as “quality attributes”. The system should also meet the non-functional requirements along with the functional requirements.

Non-functional requirements for this project are:

- **Performance**

The way in which the system meets its performance targets is for it to be specified clearly and explicitly. The system itself might not need anything specifically for its basic operation, but the complete system along with the components connected may have some performance requirements.

- **User-friendly**

The graphical user interface (GUI) is user-friendly.

- **Usability**

Usability determines how difficult it is to learn and use the system.

- **Availability**

It means for how long the system is available for its users and for how long the system will be operational.

- **Reliability**

Reliability determines how often the software fails. The measurement is expressed in Mean Time between Failures (MTBF). The system is completely tested for robustness before the deployment. The module developed thus maintains data consistency.

2.3. SYSTEM REQUIREMENTS

The two major kinds of system requirements are hardware and software requirements.

2.3.1. HARDWARE REQUIREMENTS:

A System with minimum configuration of 2 GB RAM, 500 MB ROM , Dual core processor and should have an Internet connection.

2.3.2. SOFTWARE REQUIREMENTS

2.3.2.1. PYTHON IDLE

Python IDLE (Integrated Development and Learning Environment) is an integrated development environment (IDE) for Python. IDLE can be used to execute a single statement just like Python Shell and also to create, modify and execute Python scripts. IDLE provides a fully-featured text editor to create Python scripts that includes features like syntax highlighting, auto completion and smart indent. It also has a debugger with stepping and breakpoints features. Python is available for use with Windows, Linux, Mac OS as well as certain other platforms such as IBM AS/400, iOS, Solaris, etc.

CHAPTER 3

SYSTEM DESIGN

The system is designed as a desktop application in which the user can get the required summaries by just interacting with the GUI. The full code is written in Python programming language as it reduces the LOC and even performs faster for AI related codes. And also python has numerous Library functions which would help programmer to code without effort. The working flow of the application is discussed in chapter 3.2 and also the methodology and working of algorithm has been explained in chapter 3.1 and outputs for every module are shown in appendix 1.

3.1 METHODOLOGY

3.1.1. GUI

The GUI is built using Python Tkinter Package which helps to call in-built function to create Windows, Tabs, Scrolled Text, etc.,. In this project the entire GUI is built using python programming language. This desktop application has been designed of four tabs. The Window and tabs are designed using python as we can see its code in chapter 3.2. The complete GUI is shown in appendix A.1, A.2, A.3 and A.4.

3.1.2. TEXT PREPROCESSING



Fig: 3.1 : Text Preprocessing

As we discussed in chapter 1.5, the text should be tokenized that is to separate the words and even sentences and storing it in the dictionary. And then the stop words, that is the words which has no meaning when stand-alone, for example a, the, is, was, so these kind of words are called as stop words which should be removed as it will be huge in count in the input text, it will be the disturbance while scoring sentences. Using `nltk.corpus.stopwords` package, stop words can be neglected.

3.1.3. SUMMARIZATION

This Application uses two summarization techniques namely, Text Rank algorithm which is similar to page rank algorithm and other one is Word frequency algorithm. Let us discuss these two methods of summarization in depth.

3.1.4. WORD FREQUENCY ALGORITHM

The Word frequency algorithm is the simplest way of finding the sentence score and ranking them to obtain a meaningful summary.

STEP 1: The input text should be preprocessed and ready for creating a frequency table. The Frequency table is the list of the count of each word in the preprocessed texts.

STEP 2: From the frequency weight for individual tokens that is each word should be calculated by dividing its frequency by the frequency of the most occurring word.

$$W = \frac{\text{FrequencyOfTerm}}{\text{FrequencyOfMostOccuringWord}}$$
$$W = \frac{\text{FrequencyOfTerm}}{\text{FrequencyOfMostOccuringWord}} \quad (1)$$

STEP 3: The sentence scores are calculated by the weights of words for each sentence.

STEP 4: Taking average on weights of all sentences and consider it as a threshold value.

STEP 5: The sentences which has weight value greater than or equal to the threshold value is taken as a summary. The output is shown in appendix A.10

3.1.5. TEXT RANK ALGORITHM

Text Rank is an unsupervised algorithm for summarizing the texts automatically that can be used to attain the most significant keywords in a document. This algorithm is a base of Page Rank over a graph constructed particularly for the summarization task. The only difference is what page rank uses ranking of pager while text rank used to rank sentences by using symmetrical graphs.

This uses the ranking of the elements in the graph, The most essential elements are the ones that better describe the text. Text Rank used to build summaries without the need of a training corpus and allows summarizing different languages. This text rank summarization is possible by using Gensim Package in python.

STEP 1: Pre process the raw text.

STEP 2: Tokenizing the sentences.

STEP 3: Finding vector representation for every individual sentence.

STEP 4: The Similarities between sentence vectors are then calculated using a cosine similarity matrix. In TextRank, we calculate a cosine similarity matrix in which we have the similarity of individual sentences to each other sentences.

The cosine similarity is calculated as,

$$\text{similarity}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} + \sqrt{\sum_{i=1}^n B_i^2}} \quad \text{similarity}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n A_i^2} + \sqrt{\sum_{i=1}^n B_i^2}}$$

(2)

where A and B are the two sentence vectors .

Values range between -1 and 1, where

-1 is perfectly dissimilar and 1 is perfectly similar

STEP 5: The similarity matrix is then converted into a graph, with sentences as vertices and similarities scores as edges, for sentence rank calculation. For avoiding long sentences with no keywords, we are using a normalization factor, and divide the content overlap of two sentences with the length of each sentence. Formally, given two sentences Si and Sj, with a sentence being represented by the set of Ni words that appear in the sentence: Si = Wi 1 , Wi 2 , ..., Wi Ni , the similarity of Si and Sj is defined as:

$$\text{Similarity}(\mathbf{S_i}, \mathbf{S_j}) = \frac{|\mathbf{W_k}|\mathbf{W_k} \in \mathbf{S_i} \& \mathbf{W_k} \in \mathbf{S_j}|}{\log(|\mathbf{S_i}|) + \log(|\mathbf{S_j}|)}$$

$$\text{Similarity}(\mathbf{S_i}, \mathbf{S_j}) = \frac{|\mathbf{W_k}|\mathbf{W_k} \in \mathbf{S_i} \& \mathbf{W_k} \in \mathbf{S_j}|}{\log(|\mathbf{S_i}|) + \log(|\mathbf{S_j}|)} \quad (3)$$

STEP 6: Finally, a certain number of top-ranked sentences form the final summary by using BM25 Ranking Algorithm.

BM25 ranking function is used as the state of the art for Retrieval of information tasks. It is an in-built function of Gensim python which extract sentences from graph as like in Page Rank algorithm. The output of this module is shown in appendix A.5, A.8 and A.9.

3.1.6. AUDIO AND QR CODE GENERATION

The Audio is generated by using GTTS (Google Text To Speech) Toolkit available in python as a callable function. GTTS can speak almost every languages. Using this package the summarized text in converted into audio and played using Play sound package. For QR code generation the QR Code generating package is there in python. The output of this module is shown in appendix A.6.

3.2. FLOW DIAGRAM

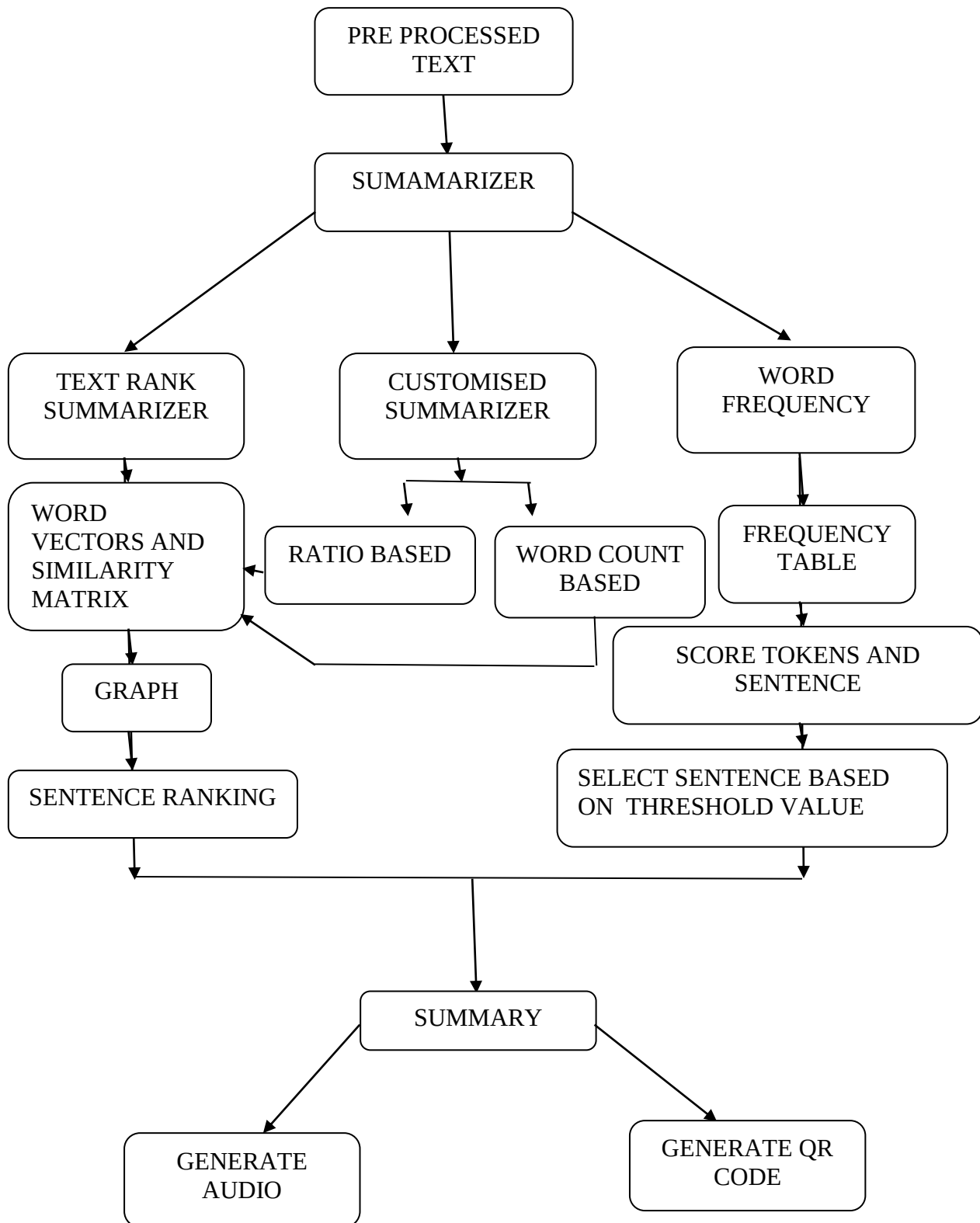


Fig: 3.2 : Flow Chart

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1. SOFTWARE TOOLS

Software tools helps in building the project as per the aim. Following are the software tools used for the project.

4.1.1. PYTHON PROGRAMMING

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to Java or C++. It has a comprehensive and large standard library that has automatic memory management and dynamic features. It has a comprehensive and large standard library that has automatic memory management and dynamic features.

4.1.2. PYTHON IDLE

Python IDLE (Integrated Development and Learning Environment) is an integrated development environment (IDE) for Python. IDLE can be used to execute a single statement just like Python Shell and also to create, modify and execute Python scripts. IDLE provides a fully-featured text editor to create Python scripts that includes features like syntax highlighting, autocompletion and smart indent. It also has a debugger with stepping and breakpoints features. Python is available for use with Windows, Linux, Mac OS as well as certain other platforms such as IBM AS/400, iOS, Solaris, etc.

4.2. PACKAGE REQUIREMENTS

```
#For Summarizer GUI Support
```

```
import tkinter as tk
```

```
from tkinter import *
```

```
from tkinter import ttk
```

```
from tkinter.scrolledtext import *
```

```
#For NLP
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize

#For detecting the language
from langdetect import detect

#For Text Rank Algorithm
import gensim
from gensim.summarization import keywords

#For handling QR images
import cv2

#For QR Code generation
import qrcode

#For playing Audio File
import playsound
from playsound import playsound
#For Google Assistant to convert text to speech for all languages
import gtts
from gtts import gTTS
from googletrans import Translator

#For creating mp3 file name randomly as it is not rewritable and after the
#usage,mp3 file will be destroyed using os library
import string
import random
import os
```

4.3. GUI APPLICATION SAMPLE CODE

```
# Layout for GUI App
window = Tk()
window.title("Extractive Text Summarizer")
window.geometry("900x650")
window.config(background='black')
style = ttk.Style(window)
style.config('lefttab.TNotebook', tabposition='wn',)

# Layout Tabs
tab_control = ttk.Notebook(window,style='lefttab.TNotebook')
tab1 = ttk.Frame(tab_control)
tab2 = ttk.Frame(tab_control)
tab3 = ttk.Frame(tab_control)
tab4 = ttk.Frame(tab_control)

# Tabs in notebook
tab_control.add(tab1, text=f{"Home":^40s'})
tab_control.add(tab2, text=f{"Customized Summarizer ":^25s'})
tab_control.add(tab3, text=f{"Basic Algo":^40s'})
tab_control.add(tab4, text=f{"About ":^42s'})
label1 = Label(tab1, text= 'Summarizer',padx=5, pady=5)
label1.grid(column=0, row=0)
label2 = Label(tab2, text= 'Customized Summarizer',padx=5, pady=5)
label2.grid(column=0, row=0)
label3 = Label(tab3, text= 'Basic Algo',padx=5, pady=5)
label3.grid(column=0, row=0)
label4 = Label(tab4, text= 'About',padx=5, pady=5)
label4.grid(column=0, row=0)
tab_control.pack(expand=1, fill='both')
```

```

# Home page Tab
l1=Label(tab1,text="Enter Text To Summarize")
l1.grid(row=1,column=0)
entry=ScrolledText(tab1,height=13)
entry.grid(row=2,column=0,columnspan=3,padx=5,pady=5)
button1=Button(tab1,text="Reset",command=clear_text1,
width=12,bg='#03A9F4',fg='fff')
button1.grid(row=4,column=0,padx=10,pady=10)
button2=Button(tab1,text="Summarize",command=summy,
width=12,bg='red',fg='fff')
button2.grid(row=4,column=1,padx=10,pady=10)
window.mainloop()

```

4.4 FUNCTIONS FOR SUMMARIZATION

#Function for using text rank package

```

def summy():
text=str(entry.get('1.0',tk.END))
summary=gensim.summarization.summarizer.summarize(text,
                                                    ratio=0.2,word_count=None, split=False)

tab1_display.insert(tk.END,summary)
count="\n\n\nOriginal Text : "+str(countWords(text))+"\nSummary :
                                                    "+str(countWords(summary))

tab1_display.insert(tk.END,count)

```

#TextRank-parameter adjusting - Ratio based

```

def ratio():
ratio=int(en1.get())/100
text=str(entry2.get('1.0',tk.END))
summary=gensim.summarization.summarizer.summarize(text,
                                                    ratio=ratio, word_count=None, split=False)

tab2_display.insert(tk.END,summary)
count="\n\n\nOriginal Text : "+str(countWords(text))+"\nSummary :
"+str(countWords(summary))
tab2_display.insert(tk.END,count)

```

```

#TextRank parameter adjusting - Word count based
def wordCount():
    wc=int(en2.get())
    text=str(entry2.get('1.0',tk.END))
    summary=gensim.summarization.summarizer.summarize(text,
                                                        ratio=0.2, word_count=wc, split=False)
    tab2_display.insert(tk.END,summary)
    count="\n\n\nOriginal Text : "+str(countWords(text))+"\nSummary :
                                                "+str(countWords(summary))

    tab2_display.insert(tk.END,count)

#Function for counting words in summary and original text
def countWords(text_string):
    total = len(word_tokenize(text_string))
    return total

#Functions for Basic word frequency and sentence ranking algorithm
def summarize():
    text_string = str(entry3.get('1.0',tk.END))
    #Create the word frequency table
    freq_table = _create_frequency_table(text_string)
    #Tokenize the sentences
    sentences = sent_tokenize(text_string)
    #Important Algorithm: score the sentences
    sentence_scores = _score_sentences(sentences, freq_table)
    #Find the threshold
    threshold = _find_average_score(sentence_scores)
    #Important Algorithm: Generate the summary
    summary = _generate_summary(sentences, sentence_scores, 1.5 * threshold)
    tab3_display.insert(tk.END,summary)
    count="\n\n\nOriginal Text : "+str(countWords(text_string))+"\nSummary :
"+str(countWords(summary))

```

```

)
tab3_display.insert(tk.END,count)

def _create_frequency_table(text_string) -> dict:
    stopWords = set(stopwords.words())
    words = word_tokenize(text_string)
    ps = PorterStemmer()
    freqTable = dict()
    for word in words:
        word = ps.stem(word)
        if word in stopWords:
            continue
        if word in freqTable:
            freqTable[word] += 1
        else:
            freqTable[word] = 1
    return freqTable

def _score_sentences(sentences, freqTable) -> dict:
    sentenceValue = dict()
    for sentence in sentences:
        word_count_in_sentence = (len(word_tokenize(sentence)))
        for wordValue in freqTable:
            if wordValue in sentence.lower():
                if sentence[:10] in sentenceValue:
                    sentenceValue[sentence[:10]] += freqTable[wordValue]
                else:
                    sentenceValue[sentence[:10]] = freqTable[wordValue]
            sentenceValue[sentence[:10]] = sentenceValue[sentence[:10]] //
word_count_in_sentence
        #print(sentenceValue,end="\n")
    return sentenceValue

```

```

def _find_average_score(sentenceValue) -> int:
    sumValues = 0
    for entry in sentenceValue:
        sumValues += sentenceValue[entry]
    # Average value of a sentence from original text
    average = int(sumValues / len(sentenceValue))
    #print(average,end="\n")
    return average

def _generate_summary(sentences, sentenceValue, threshold):
    sentence_count = 0
    summary = ""
    for sentence in sentences:
        if sentence[:10] in sentenceValue and sentenceValue[sentence[:10]] >
                                                    (threshold):
            summary += " " + sentence
            sentence_count += 1
    #print(summary)
    return summary

```

4.5 FUNCTION FOR CONVERTING TEXT TO SPEECH

```

#Function for Audio generation-General
def aUdio(raw_text):
    lan=detect(raw_text)
    letters = string.ascii_letters
    random_text = "".join(random.choice(letters) for i in range(10))
    summarized_text=gTTS(raw_text,lang=lan)
    audio=random_text+".mp3"
    summarized_text.save(audio)
    playsound(audio)
    os.remove(audio)

```



```
#Funtion for getting audio output for tab 1
def getAudio1():
    raw_text = str(tab1_display.get('1.0',tk.END))
    aUdio(raw_text)
```

```
#Funtion for getting audio output for tab 2
def getAudio2():
    raw_text = str(tab2_display.get('1.0',tk.END))
    aUdio(raw_text)
```

```
#Funtion for getting audio output for tab 3
def getAudio3():
    raw_text = str(tab3_display.get('1.0',tk.END))
    aUdio(raw_text)
```

4.6 FUNCTION FOR CONVERTING TEXT TO QR CODE

```
#Function for QR code generation-General
def getQRcode(raw_text):
    qr = qrcode.QRCode(version = 1,
        error_correction = qrcode.constants.ERROR_CORRECT_H,
        box_size = 10,
        border = 4,)
    qr.add_data(raw_text)
    qr.make(fit=True)
    img = qr.make_image()
    img.save("summary.jpeg")
    dim=(400,400)
    pic=cv2.imread("summary.jpeg")
    pic=cv2.resize(pic,dim,interpolation = cv2.INTER_AREA)
    cv2.imshow('Summary',pic)
    cv2.waitKey(0)
```

```

#Funtion for generating QR Code for tab 1
def generateQRCode1():
    raw_text = str(tab1_display.get('1.0',tk.END))
    getQRcode(raw_text)
#Funtion for generating QR Code for tab 2
def generateQRCode2():
    raw_text = str(tab2_display.get('1.0',tk.END))
    getQRcode(raw_text)
#Funtion for generating QR Code for tab 3
def generateQRCode3():
    raw_text = str(tab3_display.get('1.0',tk.END))
    getQRcode(raw_text)

```

4.7 FUNCTION FOR CLEARING RESULTS AND TO RESET SCROLLED TEXT

```

# Clear entry widget - Reset and Clear Results
def clear_text1():
    entry.delete('1.0',END)
def clear_display_result1():
    tab1_display.delete('1.0',END)
def clear_text2():
    entry2.delete('1.0',END)
def clear_display_result2():
    tab2_display.delete('1.0',END)
def clear_text3():
    entry3.delete('1.0',END)
def clear_display_result3():
    tab3_display.delete('1.0',END)

```

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

The GUI-based Automatic extractive Text summarization was designed and coded which emerges trends in education domains, biomedicine, product review, emails and blogs. This is because there is information overload in these areas, especially on the World Wide Web. Automated summarization is an important area in NLP research. The purpose of extractive document summarization is to automatically select several indicative sentences, passages, or paragraphs from the original document. The objective of text summarization is to present the source text into an appropriate version with semantics. The primary plus of using summarization is to reduce reading time.

Most summarization techniques are based on extractive methods. Abstractive summarization requires complex code for language generation and it is difficult to make a model into the domain-specific areas. So using the extractive method of summarization, the Text rank algorithm made the summarizer application perform faster than any other extractive method and even it has the feature of converting text into audio and also into QR code for making efficient sharing of content to others by just scanning the QR code. Since the Text rank algorithm is of statistical-based computations and graph-based sentence ranking it has a high ROUGE-1 score than any other extractive method of summarization.

The sentences that are highly relevant to other sentences in the source text are likely to be more informative for the given text, and will be therefore given a higher score as being useful for the overall understanding of the text. As a future enhancement, a simple neural network will be modeled for ranking sentences through encoding and decoding, which can make the application system to perform as like as human beings.

APPENDIX 1

SCREEN SHOTS

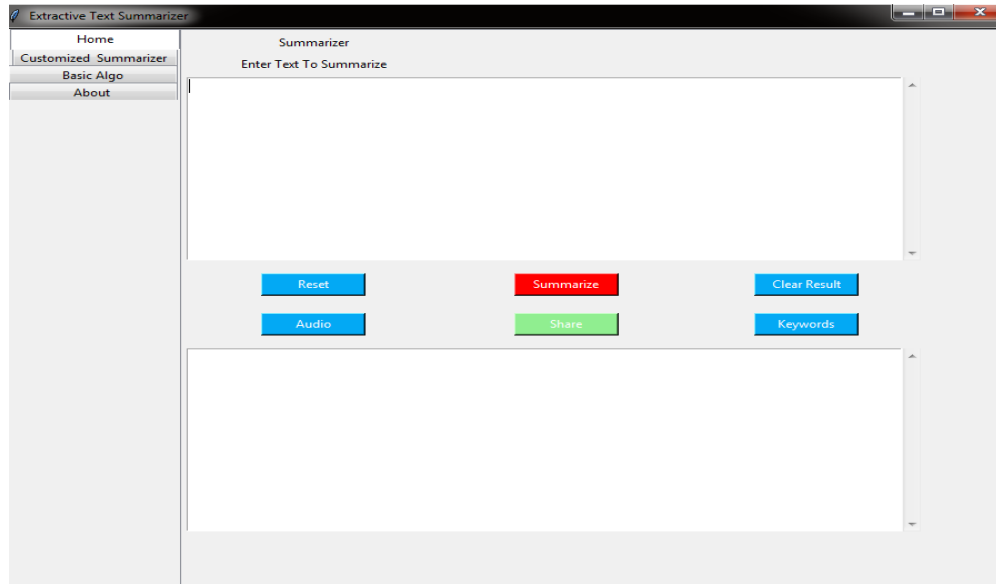


Fig: A.1 : Home Tab

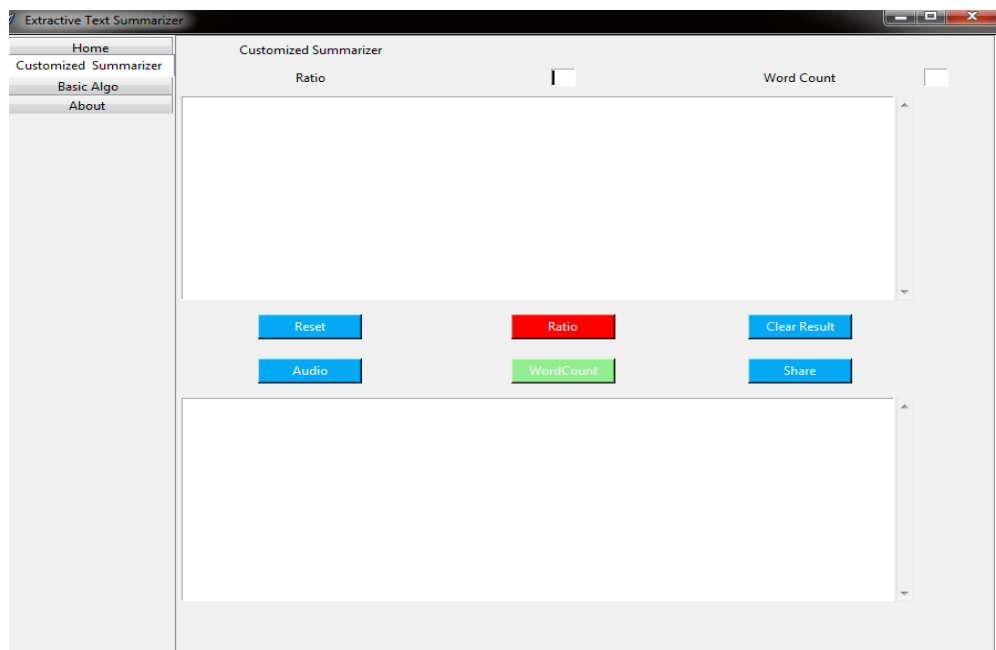


Fig : A.2 : Customized Summarization Tab

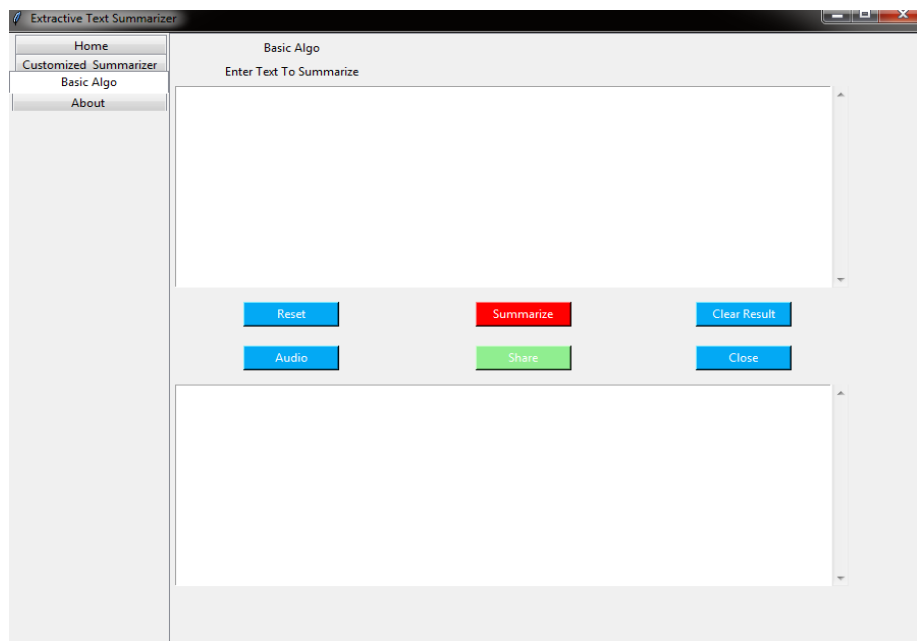


Fig : A.3 : Basic Algo Tab

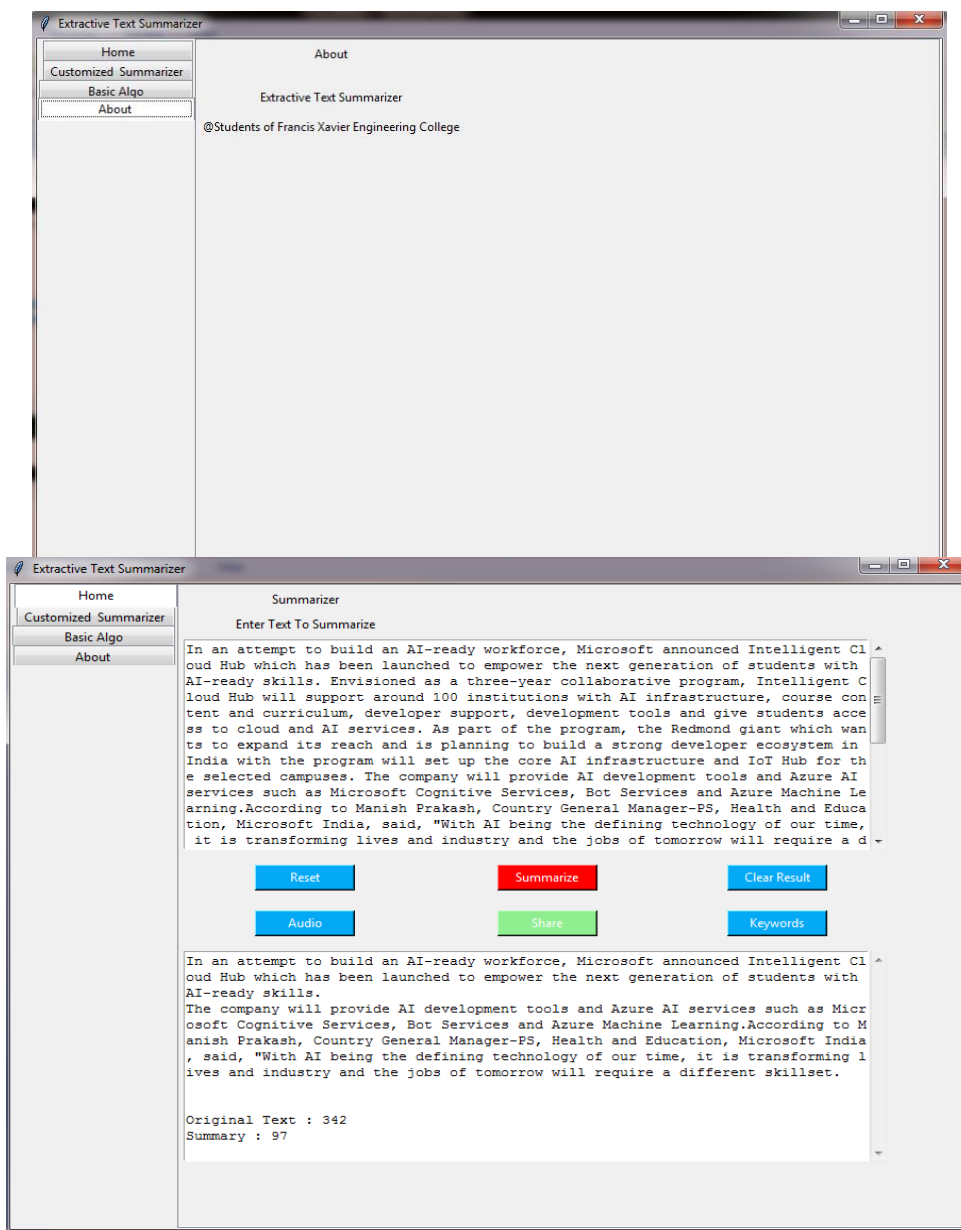


Fig :
About

A.4 :
Tab

Fig : A.5 : Summary-Text Rank

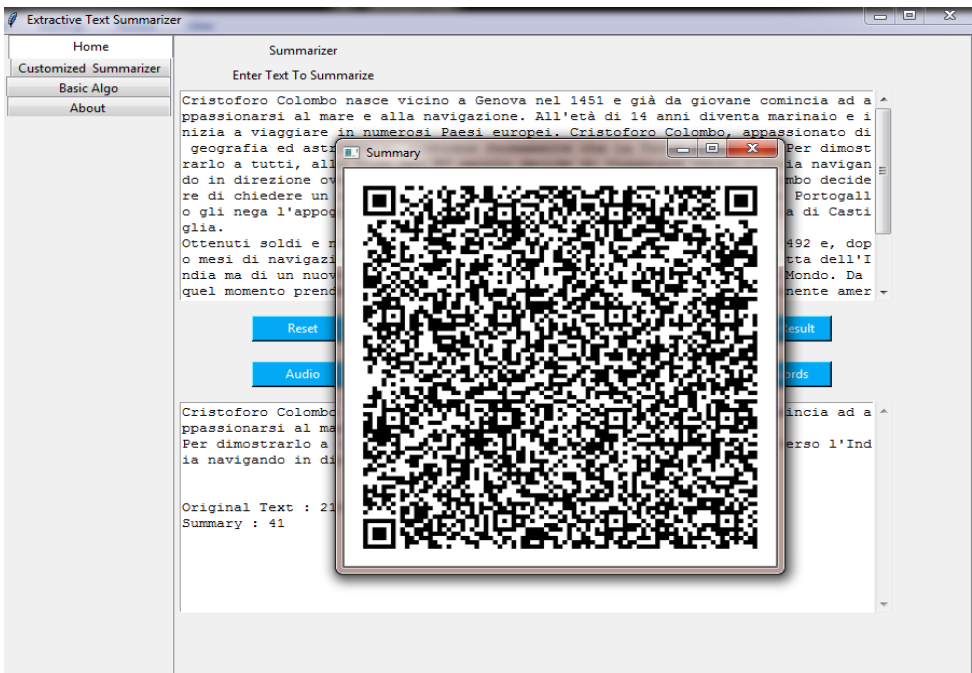


Fig : A.6 : QR code generation

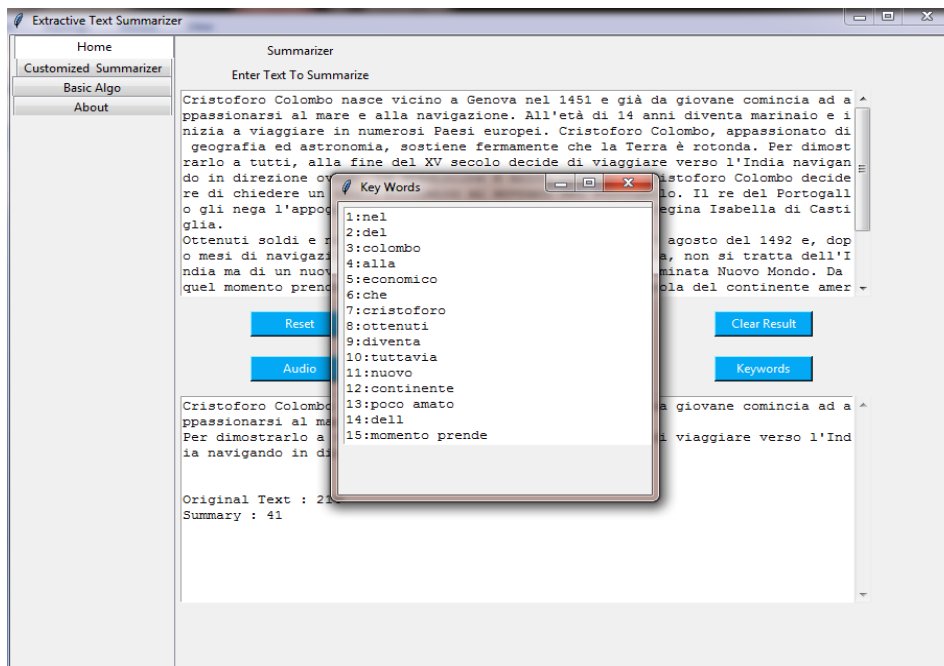


Fig. : A.7 : Extracted keywords

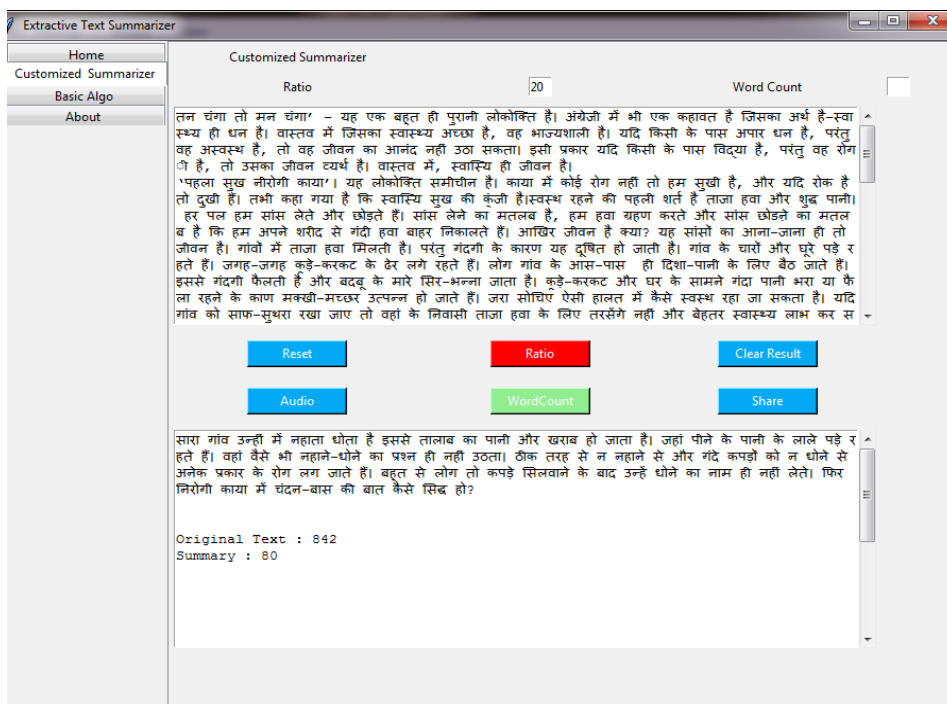


Fig : A.8 : Summary-Ratio

Fig :
A.9 :
Sum
mar
y-
Wor
d
count

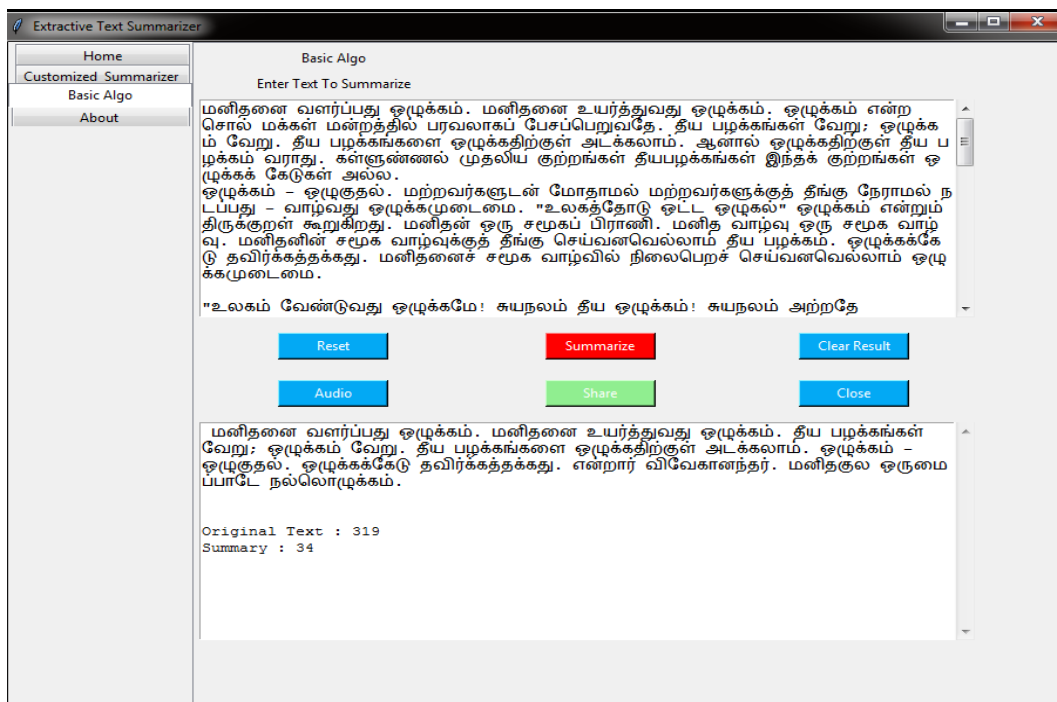


Fig : A.10 : Summary-Word frequency

REFERENCES

1. Angel, C.H. Rene, A.G.H. Yulia, L. And Christian, E.H.(2020) 'Extractive Automatic Text Summarization Based on Lexical- Semantic Keywords', IEEE Access, Vol.8, pp. 49896 – 49907.
2. Alvee, R. Fahim, M.D. Ramkrishna, S. Ruhit, R. and Hossain, A. (2019) 'Bengali Text Summarization using TextRank, Fuzzy C-Means and Aggregate Scoring methods', IEEE Region 10 Symposium (TENSYP), pp. 331-336.
3. Akshi, K. Aditi, S. Sidhant, S. and Shashwat, K. (2017) 'Performance Analysis of Keyword Extraction Algorithms Assessing Extractive Text Summarization', International Conference on Computer, Communications and Electronics, pp. 408 – 414.
4. Devika, R. and Subramaniaswamy, V. (2019) 'A semantic graph-based keyword extraction model using ranking method on big social data', Wireless Networks, Springer, pp. 1-13.
5. Dijana, K. and Vlado, K. (2019) 'Automatic Text Summarization of News Articles in Serbian Language', 18th International Symposium INFOTEH-JAHORINA(INFOTEH), pp. 1 – 6.
6. Madhuri, J.N. and Ganesh, K.R. (2019) 'Extractive Text Summarization Using Sentence Ranking' International Conference on Data Science and Communication (IconDSC), pp. 1-3.
7. Mona, T.K. Mohan, S.S. H S S S Raviteja, D. Sai, K.P.V. (2018) 'Smart Summarizer for Blind People', 3rd International Conference on Inventive Computation Technologies (ICICT) , pp. 15-18.

- 8.** Mudasir, M. Rafiya, J. and Muzaffar, S. (2019) 'Text Document Summarization using Word Embedding', Expert Systems with Applications, Vol.143, No.1, p. 112958.
- 9.** Reda, E. Gamal, B. and Akram, E.K. (2019) 'Extractive Arabic Text Summarization Using Modified Page Rank Algorithm', Egyptian Informatics Journal, Vol.21, No.2, pp. 73-81.
- 10.** Suzan, V. Emiel, K. Sander, W. and Antal, V.B . (2019) 'Query-based summarization of discussion threads', Natural Language Engineering, Vol.26, No.1, pp. 3-29.
- 11.** Umadevi, K.S , Romansha, C. Nivedita, S. Likitha, A. and Jagadeesh, K.R.(2018) 'Text Summarization of Spanish Documents', International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1793-1797.
- 12.** Yogesh, K.M. Dinesh, G.(2014) 'Analysis of Sentence Scoring Methods for Extractive Automatic Text Smmarization', International Conference on Information and Communication Technology for Competitive Strategies (ICTCS), pp. 1-6.