A Project Report

on

Text Summarization Method

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

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

17WH1A0512	Ms. S VARSHA
17WH1A0533	Ms. K PRASEEDA
17WH1A0555	Ms. D KEERTHANA

under the esteemed guidance of

K. Rajesh

Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD College of Engineering for Women

(NBA Accredited EEE, ECE, CSE, IT B.Tech. Courses,

Accredited by NAAC with ‘A’ Grade)

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad

May 2021

DECLARATION

We hereby declare that the work presented in this project entitled **“Text Summarization Method**” submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering For Women**’**, Hyderabad is an authentic record of our original work carried out under the guidance of Mr.K.Rajesh,Assistant Professor, Department of CSE.

Sign with date:
Ms.S.Varsha
(17WH1A0512)

Sign with date:
Ms.K.Praseeda
(17WH1A0533)

Sign with date:
Ms.D.Keerthana
(17WH1A0555)

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

6Certificate

This is to certify that the Project Work report on "**Text Summarization method**" is a bonafid work carried out by Ms.S.Varsha(17WH1A0512);Ms.K.Praseeda(17WH1A0533) ; Ms.D.Keerthana(17WH1A0555) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department	**Guide**
Dr.Srinivasa Reddy Konda	**Mr. K.Rajesh**
Professor and HOD,	**Assisant Professor**
Department of CSE	

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr.K.Srinivasa Reddy, Professor**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr.K.Rajesh, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

We would like to record my sincere thankfulness to the major project coordinator **Dr. Ganti Naga Satish, Professor, CSE,** for his valuable guidence and skillful management.

Sign with date:
Ms.S.Varsha
(17WH1A0512)

Sign with date:
Ms.K.Praseeda
(17WH1A0533)

Sign with date:
Ms.D.Keerthana
(17WH1A0555)

TABLES OF CONTENTS

ABSTRACT

In this new era, where tremendous information is available on the internet, it is most important to provide the improved mechanism to extract the information quickly and most efficiently. It is very difficult for human beings to manually extract the summary of a large documents of text. Text summarization has become very essential to gain just the right amount of information from huge texts. Mostly, the methods described and discuss in this is produce Abstractive (ABS) or Extractive (EXT) summaries of text documents and about the structured based and semantic based approaches for summarization of the text documents.

LIST OF FIGURES

1.INTRODUCTION

Before going to the Text summarization, first we, have to know that what a summary is. A summary is a text that is produced from one or more texts, that conveys important information in the original text, and it is of a shorter form. The goal of automatic text summarization is presenting the source text into a shorter version with semantics. The most important advantage of using a summary is, it reduces the reading time.

Text Summarization methods can be classified into

1)Extractive and

2)Abstractive summarization.

EXTRACTIVE SUMMARIZATION: An extractive summarization method consists of selecting important sentences, paragraphs etc. from the original document and concatenating them into shorter form.

EXAMPLE:

Source text: Joseph and Mary rode on a donkey to attend the annual event in Jerusalem. In the city, Mary gave birth to a child named Jesus.

Extractive summary: Joseph and Mary attend event Jerusalem. Mary birth Jesus.

ABSTRACTIVE SUMMARIZATION: Abstractive summarization is an understanding of the main concepts in a document and then express those concepts in clear natural language

EXAMPLE:

Source text: Joseph and Mary rode on a donkey to attend the annual event in Jerusalem. In the city, Mary gave birth to a child named Jesus.

Abstractive summary**:** Joseph and Mary came to Jerusalem where Jesus was born.

1.1 Abstractive based summarization

An **abstract** is a brief summary of a research article,thesis, review, conference proceeding, or any in-depth analysis of a particular subject and is often used to help the reader quickly ascertain the paper's purpose. When used, an abstract always appears at the beginning of a manuscript or typescript, acting as the point-of-entry for any given academic paper or patent application. Abstracting and indexing services for various academic disciplines are aimed at compiling a body of literature for that particular subject. Academic literature uses the abstract to succinctly communicate complex research. An abstract may act as a stand-alone entity instead of a full paper. As such, an abstract is used by many organizations as the basis for selecting research that is proposed for presentation in the form of a poster, platform/oral presentation or workshop presentation at an academic conference. Most bibilography databases only index abstracts rather than providing the entire text of the paper. Full texts of scientific papers must often be purchased because of copyright and/or publisher fees and therefore the abstract is a significant selling point for the reprint or electronic form of the full text. must often be purchased because of copyright and/or publisher fees and therefore the abstract is a significant selling point for the reprint or electronic form of the full text.The abstract can convey the main results and conclusions of a scientific article but the full text article must be consulted for details of the methodology, the full experimental results, and a critical discussion of the interpretations and conclusions.An abstract allows one to sift through copious numbers of papers for ones in which the researcher can have more confidence that they will be relevant to his or her research. Once papers are chosen based on the abstract, they must be read carefully to be evaluated for relevance. It is generally agreed that one must not base reference citations on the abstract alone, but the content of an entire paper.

1.2 Extractive based summarization

Here, content is extracted from the original data, but the extracted content is not modified in any way. Examples of extracted content include key-phrases that can be used to "tag" or index a text document, or key sentences (including headings) that collectively comprise an abstract, and representative images or video segments, as stated above. For text, extraction is analogous to the process of skimming, where the summary (if available), headings and subheadings, figures, the first and last paragraphs of a section, and optionally the first and last sentences in a paragraph are read before one chooses to read the entire document in detail.

Other examples of extraction that include key sequences of text in terms of clinical relevance (including patient/problem, intervention, and outcome)

There are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query.

Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.

An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic). This problem is called multi-document summarization. A related application is summarizing news articles. Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary.

1.3. Tokenizer:

Tokenizer is a compact pure-Python (>= 3.6) executable program and module for tokenizing Icelandic text. It converts input text to streams of tokens, where each token is a separate word, punctuation sign, number/amount, date, e-mail, URL/URI, etc. It also segments the token stream into sentences, considering corner cases such as abbreviations and dates in the middle of sentences.

To run the below python program, (NLTK) natural language toolkit has to be installed in your system.

The NLTK module is a massive tool kit, aimed at helping you with the entire Natural Language Processing (NLP) methodology.

In order to install NLTK run the following commands in your terminal.

- sudo pip install nltk
- Then, enter the python shell in your terminal by simply typing python
- Type import nltk
- nltk.download('all')

The above installation will take quite some time due to the massive amount of tokenizers, chunkers, other algorithms, and all of the corpora to be downloaded.

Some terms that will be frequently used are:

- Corpus – Body of text, singular. Corpora is the plural of this.
- Lexicon – Words and their meanings.
- Token – Each "entity" that is a part of whatever was split up based on rules. For examples, each word is a token when a sentence is "tokenized" into words. Each sentence can also be a token, if you tokenized the sentences out of a paragraph. So basically tokenizing involves splitting sentences and words from the body of the text.

1.3.1Page Rank:

PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of "measuring" its relative importance within the set. A PageRank results from a mathematical algorithm based on the webgraph, created by all World Wide Web pages as nodes and hyperlinks as edges, taking into consideration authority hubs such as cnn.com or mayoclinic.org. The rank value indicates an importance of a particular page. A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages that link to it ("incoming links"). A page that is linked to by many pages with high PageRank receives a high rank itself.Numerous academic papers concerning

PageRank have been published since Page and Brin's original paper. In practice, the PageRank concept may be vulnerable to manipulation. Research has been conducted into identifying falsely influenced PageRank rankings. The goal is to find an effective means of ignoring links from documents with falsely influenced PageRank.

1.3.2 Stop Words:

stop words are words which are filtered out before or after processing of natural language data (text). Though "stop words" usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even use such a list. Some tools specifically avoid removing these stop words to support phrase search.

Any group of words can be chosen as the stop words for a given purpose. For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on. In this case, stop words can cause problems when searching for phrases that include them, particularly in names such as "The Who", "The The", or "Take That". Other search engines remove some of the most common words—including lexical words, such as "want"—from a query in order to improve performance.

1.4 Objectives

The main objective of the text summarization is the reduction of a given text to a smaller number of sentences without leaving out the main ideas of the original text.

1.5 Methodology

Steps to Text Summarization:

1) Text Cleaning: Removing stop words, punctuation marks and making the words in lower case.
2) Work Tokenization: Tokenize each word from sentences.
3) Word Frequency table: Count the frequency of each word and then divide the maximum frequency with each frequency to get the normalized word frequency count.
4) Sentence Tokenization: As per frequency of sentence then
5) Summarization

1.5.1 Dataset

Dataset collection: Text you want to summarize

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering

2.1.1 Software Requirements

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:

• A condition or capability needed by a user to solve a problem or achieve an objective.

• A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.

• A documented representation of a condition or capability as in 1 or 2.

The activities related to working with software requirements can broadly be broken down into elicitation, analysis, specification, and management.

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

A software requirement can be of 3 types:

• Functional requirements: These are the requirements that the end user specifically Demands as basic facilities that the system should offer.

• Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract.

• Domain requirements: These are the requirements which are characteristic of a particular category or domain of projects

This project requirements are:

• Domain	: Machine Learning
• Programming Language	: Python 3.6,HTML,CSS
• Dataset	: Text you want to summarize.
• Packages	: Numpy, Pandas, NLTK, Genism, SpaCy and Sumy
• Tool	: PyCharm,Anaconda- Jupyter Notebook

2.1.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements. The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- Operating System : Windows 7 Ultimate 32 bit / Windows XP/ Windows 10
- Processor : Intel Processor
- CPU Speed : 2.30 GHz
- Memory : 4 GB (RAM)

2.2 Technologies Description

Machine Learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data in order to look for patterns in data and make better decisions in the future based on the examples that are provided. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms are often categorized as supervised or unsupervised.

- Supervised machine learning algorithms can apply what has been learned in the past to new data using labelled examples to predict future events.
- Unsupervised machine learning algorithms are used when the information used to train is neither classified nor labelled.

Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

- Python is Interpreted: Python is processed at runtime by the interpreter. Compilation is not needed before executing the program. This is similar to PERL and PHP.
- Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python's features include:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintain.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter.These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- Scalable: Python provides a better structure and support for large programs than shell scripting.

Jupyter Notebook

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

- Web application
- Notebook documents

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.

Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

Numpy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases. It also discusses the various array functions, types of indexing, etc. In order to import Numpy the following command is used:

import numpy as np

Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The Data Frame is one of these structures. In order to import pandas the following command is used:

import pandas as pd

PyCharm

PyCharm is the most popular IDE for Python, and includes great features such as excellent code completion and inspection with advanced debugger and support for web programming and various frameworks. PyCharm is created by Czech company, Jet brains which focusses on creating integrated development environment for various web development languages like JavaScript and PHP.

Algorithms

Algorithms used in our project are:

- SpaCy
- Gensim
- NLTK
- Sumy LexRank

Text Summarization Method

SpaCy Summarizer

SpaCy is a free, open-source advanced natural language processing library, written in the programming languages Python and Cython. SpaCy mainly used in the development of production software and also supports deep learning workflow via statistical models of PyTorch and TensorFlow.

SpaCy provides a fast and accurate syntactic analysis, named entity recognition and ready access to word vectors. We can use the default word vectors or replace them with any you have. SpaCy also offers tokenization, sentence boundary detection, POS tagging, syntactic parsing, integrated word vectors, and alignment into the original string with high accuracy**.**

Gensim Summarizer

Summarization is a useful tool for varied textual applications that aims to highlight important information within a large corpus. With the outburst of information on the web, Python provides some handy tools to help summarize a text. This provides an overview of the two major categories of approaches followed – extractive and abstractive. In this article, we shall look at a working example of extractive summarization**.**

NLTK

There are two NLTK libraries that will be necessary for building an efficient feedback summarizer.

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize,sent_tokenize

Sumy LexRank

LexRank summarizer can be implemented using an object called from Sumy package. After creating an object of LexRank Summarizer,pass the text document and number of lines to return in the summary.

Flask

Flask is microweb frame work written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask is part of the categories of the micro-framework. Micro-framework are normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

Create an environment

Create a project folder and a venv folder within:

```
$ mkdir myproject
$ cd myproject
$ python3 -m venv venv
```

Within the activated environment, use the following command to install Flask:

```
$ pip install Flask
```

Flask is now installed. Check out the Quickstart or go to the Documentation Overview.

Living on the edge

If you want to work with the latest Flask code before it’s released, install or update the code from the master branch:

```
$ pip install -U https://github.com/pallets/flask/archive/master.tar
```

3. DESIGN

3.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Once system requirements have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed, reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design consists of four activities – architectural design, data structure design, interface design and procedural design.

3.2 Architecture Diagram

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through a network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people. Like desktop applications, web applications are made up of many parts and often contain mini programs and some of which have user interfaces. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or JavaScript programming language. Also, many applications use only the Python programming language, which is ideal because of its versatility.

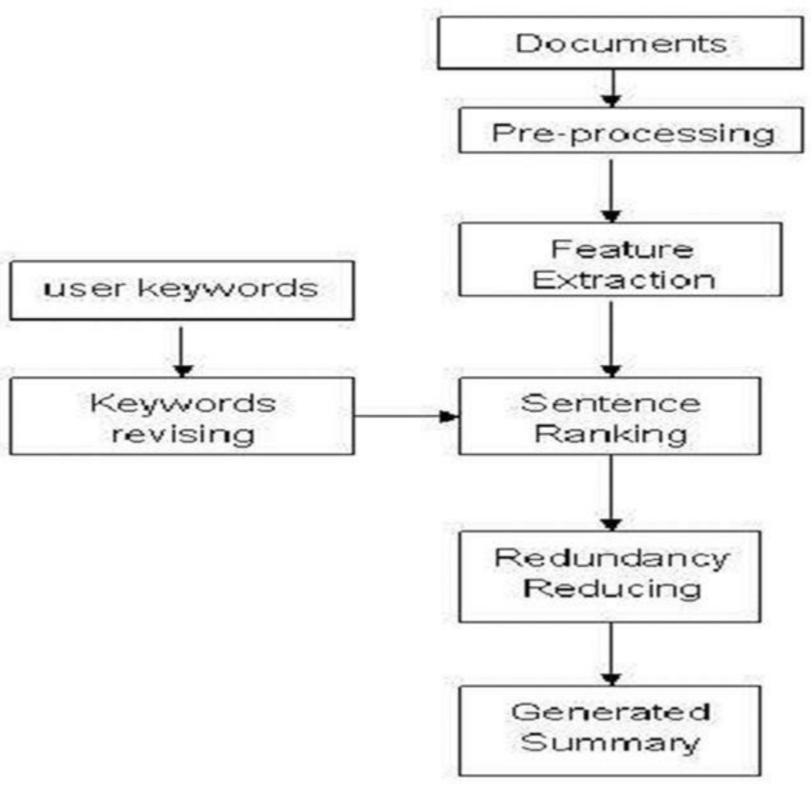


Figure 3.2: Architecture Diagram

3.3 UML Diagrams

3.3.1Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic Behavior means the behavior of the system when it is running/operating. Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagrams are one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

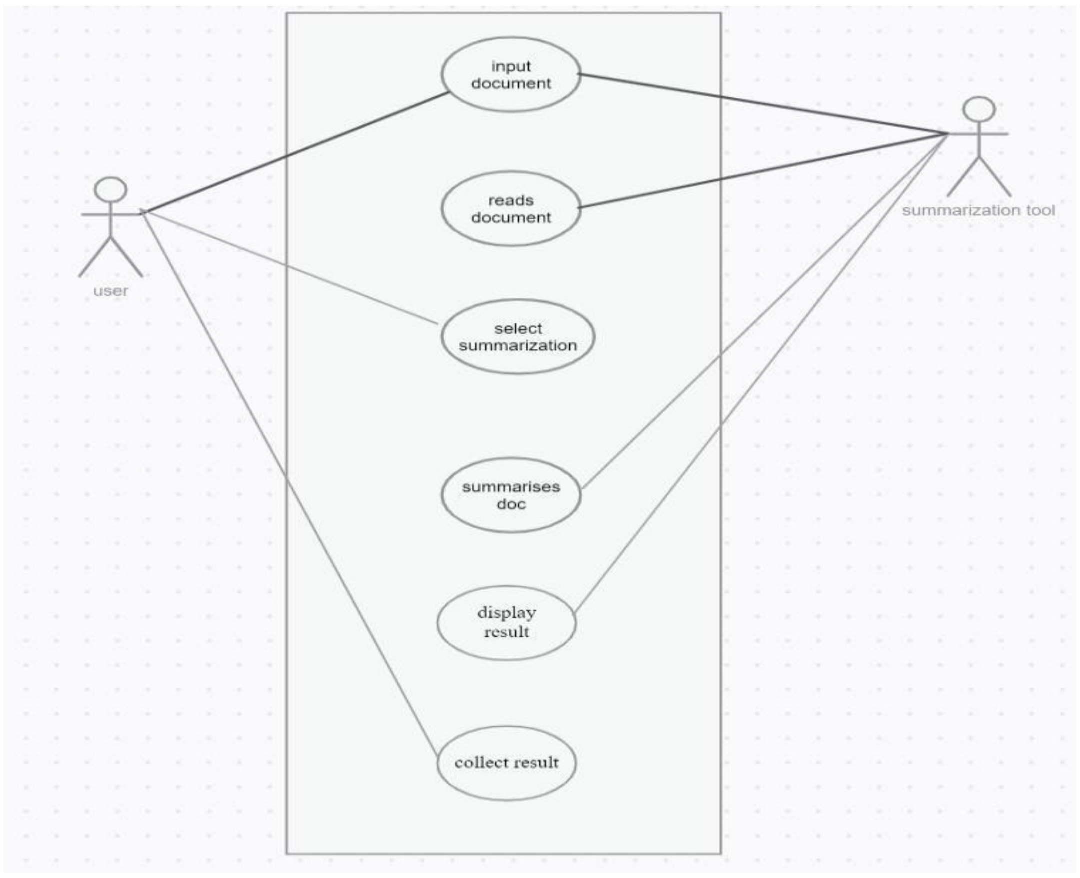


Figure 3.3.1 Use Case Diagram

3.3.2SEQUENCE DIAGRAM

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioural classifier that represents a declaration of an 21 Department of Computer Science and Engineering offered behaviour. Each use case specifies some behaviour, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behaviour of the subject without reference to its internal structure. These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behaviour, including exceptional behaviour and error handling.

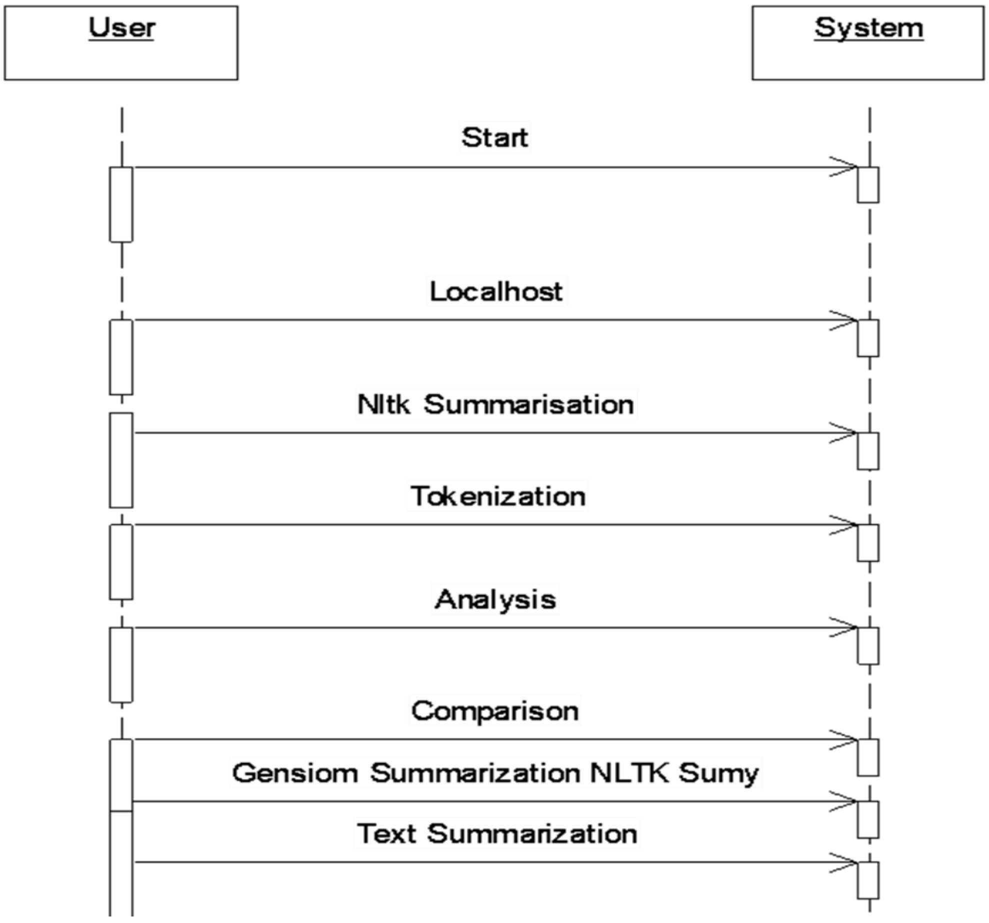


Figure 3.3.2 Sequence diagram

3.3.3Class Diagram

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

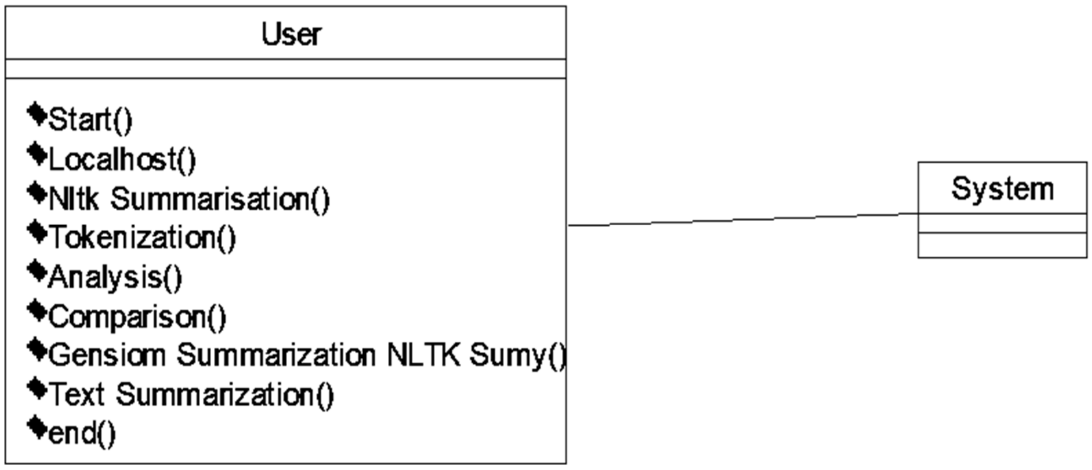


Figure. 3.3.3 Class diagram

3.3.1Flow diagram:

Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

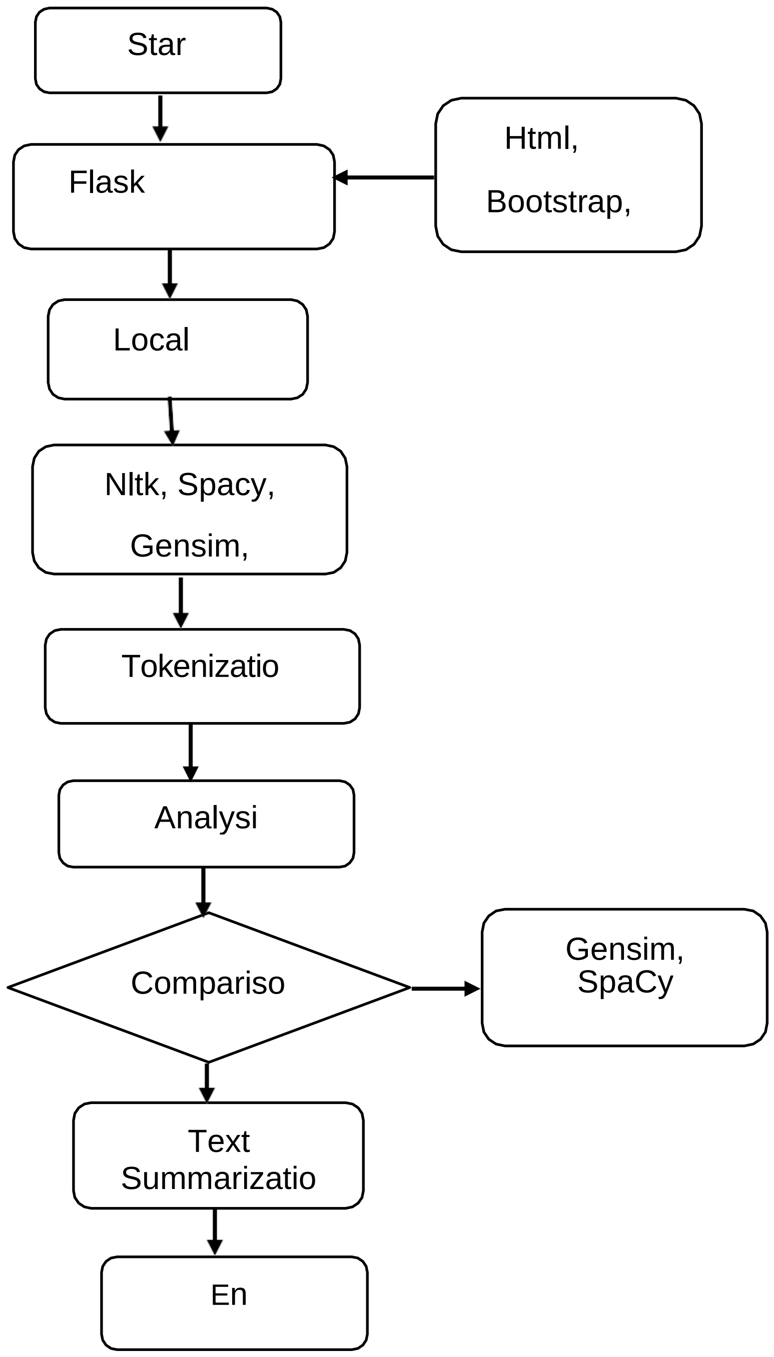


Figure. 3.3.4 Flow diagram

4. IMPLEMENTATION

4.1 Coding

#app.py

```
from___future___import unicode_literals
from flask import Flask, render_template, url_for, request
from spacy_summarization import text_summarizer
from gensim.summarization import summarize
from nltk_summarization import nltk_summarizer
import time
import spacy
nlp = spacy.load('en_core_web_sm')
app = Flask(_name_)
```

# Web Scraping Pkg

```
from bs4 import BeautifulSoup
# from urllib.request import urlopen
from urllib.request import  urlopen
```

# Sumy Pkg

```
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.lex_rank import LexRankSummarizer
```

# Sumy

```
def sumy_summary(docx):
parser = PlaintextParser.from_string(docx,Tokenizer("english"))
lex_summarizer = LexRankSummarizer()
summary = lex_summarizer(parser.document,3)
summary_list = [str(sentence) for sentence in summary]
result = ' '.join(summary_list)
return result
```

# Reading Time

```
def readingTime(mytext):
total_words = len([ token.text for token in nlp(mytext)])
estimatedTime = total_words/200.0
return estimatedTime
```

Text Summarization Method

```
# Fetch Text From Url
def get_text(url):
page = urlopen(url)
soup = BeautifulSoup(page)
fetched_text = ' '.join(map(lambda p:p.text,soup.find_all('p')))
return fetched_text
@app.route('/')
def index():
return render_template('index.html')
@app.route('/analyze',methods=['GET','POST'])
def analyze():
start = time.time()
if request.method == 'POST':
rawtext = request.form['rawtext']
final_summary = text_summarizer(rawtext)
final_summary_gensim = summarize(rawtext)
# NLTK
final_summary_nltk = nltk_summarizer(rawtext)
# Sumy
final_summary_sumy = sumy_summary(rawtext)
end = time.time()
final_time = end-start
return
render_template('index.html',ctext=rawtext,final_summary=final_summary,final_summary_s
pacy=final_summary,final_summary_gensim=final_summary_gensim,final_summary_nltk=f
inal_summary_nltk,final_summary_sumy=final_summary_sumy)
@app.route('/compare_summary')
def compare_summary():
return render_template('index.html')
@app.route('/comparer',methods=['GET','POST'])
def comparer():
start = time.time()
if request.method == 'POST':
rawtext = request.form['rawtext']
```

```
final_reading_time = readingTime(rawtext)
final_summary_spacy = text_summarizer(rawtext)
summary_reading_time = readingTime(final_summary_spacy)
# Gensim Summarizer
final_summary_gensim = summarize(rawtext)
summary_reading_time_gensim = readingTime(final_summary_gensim)
# NLTK
final_summary_nltk = nltk_summarizer(rawtext)
summary_reading_time_nltk = readingTime(final_summary_nltk)
# Sumy
final_summary_sumy = sumy_summary(rawtext)
summary_reading_time_sumy = readingTime(final_summary_sumy)
end = time.time()
final_time = end-start
return
render_template('index.html',ctext=rawtext,final_summary_spacy=final_summary_spacy,fina
l_summary_gensim=final_summary_gensim,final_summary_nltk=final_summary_nltk,final_
time=final_time,final_reading_time=final_reading_time,summary_reading_time=summary_r
eading_time,summary_reading_time_gensim=summary_reading_time_gensim,final_summar
y_sumy=final_summary_sumy,summary_reading_time_sumy=summary_reading_time_sumy
,summary_reading_time_nltk=summary_reading_time_nltk)
@app.route('/about')
def about():
return render_template('index.html')
if__name__== '_main_':
app.run(debug=True)
#attention.py
import tensorflow as tf
import os
from tensorflow.python.keras.layers import Layer
from tensorflow.python.keras import backend as K
class AttentionLayer(Layer):
    """
    This class implements Bahdanau attention (https://arxiv.org/pdf/1409.0473.pdf).
```

```
There are three sets of weights introduced W_a, U_a, and V_a
 """
 def_init_(self, **kwargs):
 super(AttentionLayer, self)._init_(**kwargs)
 def build(self, input_shape):
 assert isinstance(input_shape, list)
# Create a trainable weight variable for this layer.
self.W_a = self.add_weight(name='W_a',
shape=tf.TensorShape((input_shape[0][2], input_shape[0][2])),
initializer='uniform',
trainable=True)
self.U_a = self.add_weight(name='U_a',
shape=tf.TensorShape((input_shape[1][2], input_shape[0][2])),
initializer='uniform',
trainable=True)
self.V_a = self.add_weight(name='V_a',
shape=tf.TensorShape((input_shape[0][2], 1)),
initializer='uniform',
trainable=True)
super(AttentionLayer, self).build(input_shape) # Be sure to call this at the end
def call(self, inputs, verbose=False):
"""
inputs: [encoder_output_sequence, decoder_output_sequence]
"""
assert type(inputs) == list
encoder_out_seq, decoder_out_seq = inputs
if verbose:
print('encoder_out_seq>', encoder_out_seq.shape)
print('decoder_out_seq>', decoder_out_seq.shape)
def energy_step(inputs, states):
 """ Step function for computing energy for a single decoder state
inputs: (batchsize * 1 * de_in_dim)
 states: (batchsize * 1 * de_latent_dim)
"""
```

```
assert_msg = "States must be an iterable. Got {} of type {}".format(states, type(states))
assert isinstance(states, list) or isinstance(states, tuple), assert_msg
""" Some parameters required for shaping tensors"""
en_seq_len, en_hidden = encoder_out_seq.shape[1], encoder_out_seq.shape[2]
 de_hidden = inputs.shape[-1]
""" Computing S.Wa where S=[s0, s1, ..., si]"""
# <= batch size * en_seq_len * latent_dim
W_a_dot_s = K.dot(encoder_out_seq, self.W_a)
""" Computing hj.Ua """
U_a_dot_h = K.expand_dims(K.dot(inputs, self.U_a), 1) # <= batch_size, 1, latent_dim
if verbose:
print('Ua.h>', U_a_dot_h.shape)
""" tanh(S.Wa + hj.Ua) """
# <= batch_size*en_seq_len, latent_dim
Ws_plus_Uh = K.tanh(W_a_dot_s + U_a_dot_h)
if verbose:
print('Ws+Uh>', Ws_plus_Uh.shape)
""" softmax(va.tanh(S.Wa + hj.Ua)) """
# <= batch_size, en_seq_len
 e_i = K.squeeze(K.dot(Ws_plus_Uh, self.V_a), axis=-1)
 # <= batch_size, en_seq_len
e_i = K.softmax(e_i)
if verbose:
print('ei>', e_i.shape)
return e_i, [e_i]
def context_step(inputs, states):
""" Step function for computing ci using ei """
assert_msg = "States must be an iterable. Got {} of type {}".format(states, type(states))
assert isinstance(states, list) or isinstance(states, tuple), assert_msg
 # <= batch_size, hidden_size
c_i = K.sum(encoder_out_seq * K.expand_dims(inputs, -1), axis=1)
if verbose:
print('ci>', c_i.shape)
return c_i, [c_i]
```

```
fake_state_c = K.sum(encoder_out_seq, axis=1)
fake_state_e = K.sum(encoder_out_seq, axis=2) # <= (batch_size, enc_seq_len, latent_dim
""" Computing energy outputs """
# e_outputs => (batch_size, de_seq_len, en_seq_len)
   last_out, e_outputs, _ = K.rnn(
    energy_step, decoder_out_seq, [fake_state_e],)
""" Computing context vectors """
last_out, c_outputs, _ = K.rnn(
context_step, e_outputs, [fake_state_c],
)
return c_outputs, e_outputs
def compute_output_shape(self, input_shape):
 """ Outputs produced by the layer """
return [
tf.TensorShape((input_shape[1][0], input_shape[1][1], input_shape[1][2])),
tf.TensorShape((input_shape[1][0], input_shape[1][1], input_shape[0][1]))
 ]
```

NLTK Summarization:

We need to tokenize all the sentences to get all the words that exist in the sentences. After tokenizing the sentences, we get list of words. Next we need to find the weighted frequency of occurrences of all the words. We can find the weighted frequency of each word by dividing its frequency by the frequency of the most occurring word. The final step is to plug the weighted frequency in place of the corresponding words in original sentences and finding their sum. It is important to mention that weighted frequency for the words removed during preprocessing (stop words, punctuation, digits etc.) will be zero and therefore is not required to be added. The final step is to sort the sentences in inverse order of their sum. The sentences with highest frequencies summarize the text.

NLTK Summarization.py

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
import heapq
def nltk_summarizer(raw_text):
//calculating word frequency
```

```
stopWords = set(stopwords.words("english"))
word_frequencies = {}
for word in nltk.word_tokenize(raw_text):
if word not in stopWords:
if word not in word_frequencies.keys():
word_frequencies[word] = 1
else:
word_frequencies[word] += 1
maximum_frequncy = max(word_frequencies.values())
for word in word_frequencies.keys():
word_frequencies[word] = (word_frequencies[word]/maximum_frequncy)
sentence_list = nltk.senttokenize(raw_text)
```

In the script above, we first store all the English stop words from the nltk library into a stopwords variable. Next, we loop through all the sentences and then corresponding words to first check if they are stop words. If not, we proceed to check whether the words exist in word_frequency dictionary i.e. word_frequencies, or not. If the word is encountered for the first time, it is added to the dictionary as a key and its value is set to 1. Otherwise, if the word previously exists in the dictionary, its value is simply updated by 1.

Finally, to find the weighted frequency, we can simply divide the number of occurances of all the words by the frequency of the most occurring word

We have now calculated the weighted frequencies for all the words. Now is the time to calculate the scores for each sentence by adding weighted frequencies of the words that occur in that particular sentence. The following script calculates sentence scores:

```
//sentence scores
sentence_scores = {}
for sent in sentence_list:
for word in nltk.word_tokenize(sent.lower()):
 if word in word_frequencies.keys():
if len(sent.split(' ')) < 30:
 if sent not in sentence_scores.keys():
sentence_scores[sent] = word_frequencies[word]
else:
 sentence_scores[sent] += word_frequencies[word]
summary_sentences = heapq.nlargest(7, sentence_scores, key=sentence_scores.get)
```

```
summary = ' '.join(summary_sentences)
return summary
```

In the script above, we first create an empty sentence_scores dictionary. The keys of this dictionary will be the sentences themselves and the values will be the corresponding scores of the sentences. Next, we loop through each sentence in the sentence_list and tokenize the sentence into words.We do not want very long sentences in the summary, therefore, we calculate the score for only sentences with less than 30 words (although you can tweak this parameter for your own use-case). Next, we check whether the sentence exists in the sentence_scores dictionary or not. If the sentence doesn't exist, we add it to the sentence_scores dictionary as a key and assign it the weighted frequency of the first word in the sentence, as its value. On the contrary, if the sentence exists in the dictionary, we simply add the weighted frequency of the word to the existing value.

In the script above, we use the heapq library and call its nlargest function to retrieve the top 7 sentences with the highest scores.

#Spacy Summarization.py

```
# NLP Pkgs
import spacy
nlp = spacy.load('en_core_web_sm')
# Pkgs for Normalizing Text
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
# Import Heapq for Finding the Top N Sentences
from heapq import nlargest
def text_summarizer(raw_docx):
raw_text = raw_docx
docx = nlp(raw_text)
stopwords = list(STOP_WORDS)
# Build Word Frequency # word.text is tokenization in spacy
word_frequencies = {}
for word in docx:
if word.text not in stopwords:
if word.text not in word frequencies.keys():
word_frequencies[word.text] = 1
```

```
else:
word_frequencies[word.text] += 1
maximum_frequncy = max(word_frequencies.values())
for word in word_frequencies.keys():
word_frequencies[word] = (word_frequencies[word]/maximum_frequncy)
 # Sentence Tokens
sentence_list = [ sentence for sentence in docx.sents ]
# Sentence Scores
sentence_scores = {}
for sent in sentence_list:
for word in sent:
if word.text.lower() in word_frequencies.keys():
if len(sent.text.split(' ')) < 30:
if sent not in sentence_scores.keys():
sentence_scores[sent] = word_frequencies[word.text.lower()]
else:
sentence_scores[sent] += word_frequencies[word.text.lower()]
summarized_sentences = nlargest(7, sentence_scores, key=sentence_scores.get)
final_sentences = [ w.text for w in summarized_sentences ]
summary = ' '.join(final_sentences)
return summary
#Spacy Summarizer.py
# NLP Pkgs
import spacy
nlp = spacy.load('en')
# Pkgs for Normalizing Text
From spacy.lang.en.stop_words import STOP_WORDS
From string import punctuation
# Import Heapq for Finding the Top N Sentences
from heapq import nlargest
def text_summarizer(raw_docx):
raw_text = raw_docx
docx = nlp(raw_text)
topwords = list(STOP_WORDS
```

Text Summarization Method

```
# Build Word Frequency
# word.text is tokenization in spacy
word_frequencies = {}
for word in docx:
if word.text not in stopwords:
if word.text not in word_frequencies.keys():
word_frequencies[word.text] = 1
else:
word_frequencies[word.text] += 1
maximum_frequncy = max(word_frequencies.values())
for word in word_frequencies.keys():
word_frequencies[word] = (word_frequencies[word]/maximum_frequncy)
# Sentence Tokens
sentence_list = [ sentence for sentence in docx.sents ]
# Sentence Scores
sentence_scores = {}
for sent in sentence_list:
for word in sent:
if word.text.lower() in word_frequencies.keys():
if len(sent.text.split(' ')) < 30:
if sent not in sentence_scores.keys():
sentence_scores[sent] = word_frequencies[word.text.lower()]
else:
sentence_scores[sent] += word_frequencies[word.text.lower()]
summarized_sentences = nlargest(7, sentence_scores, key=sentence_scores.get)
final_sentences = [ w.text for w in summarized_sentences ]
summary = ' '.join(final_sentences)
print("Original Document\n")
print(raw_docx)
print("Total Length:",len(raw_docx))
print('\n\nSummarized  Document\n')
print(summary)
print("Total Length:",len(summary))
```

SUMY LEX SUMMARIZATION:

```
# Sumy Pkg
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.lex_rank import LexRankSummarizer
# Sumy
def sumy_summary(docx):
parser = PlaintextParser.from_string(docx,Tokenizer("english"))
lex_summarizer = LexRankSummarizer()
summary = lex_summarizer(parser.document,3)
summary_list = [str(sentence) for sentence in summary]
result = ' '.join(summary_list)
return result
```

Gensim Summarization:

```
# Gensim Summarizer
from gensim.summarization import summarize
final_summary_gensim = summarize(rawtext)
```

From the above four Summarization techniques spaCy provides a fast and accurate syntactic analysis, named entity recognition and ready access to word vectors. spaCy is a free, open-source advanced natural language processing library, written in the programming languages Python and Cython. spaCy mainly used in the development of production software and also supports deep learning workflow via statistical models of PyTorch and TensorFlow.

4.2 TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.There are basically two types of testing approaches.One is Black-Box testing – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.

The other is White-Box testing – knowing the internal workings of the product,tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised. White box and Black box testing methods have been used to test this package. The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

4.2.1 TESTING STRATEGIES

Testing is a set of activities that can be planned in advanced and conducted systematically.

A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function.

Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened. Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase.

This is an activity to be performed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

UNIT TESTING:

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

SYSTEM TESTING:

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths throughout the system to make sure that no path is making its way into chaos.

INTEGRATED TESTING:

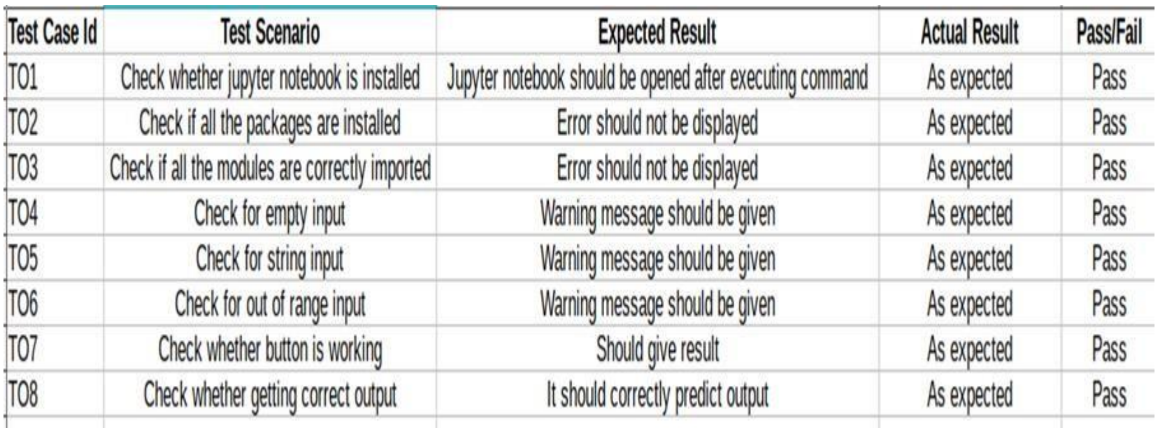
Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an endless loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.Different incremental integration strategies are top – down integration, bottom – up integration, regression testing.

REGRESSION TESTING:

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors. Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases. A representative sample to tests that will exercise all software functions. Additional tests that focus on software functions that are likely to be affected by the change.

4.3 TEST CASES

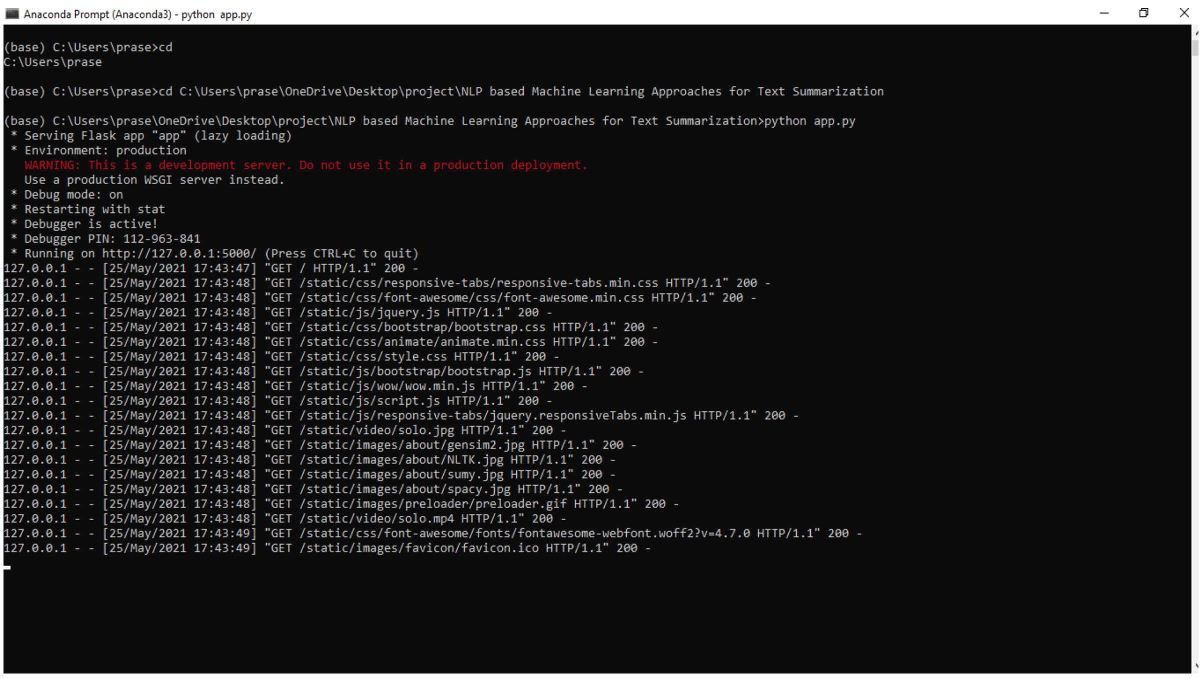
Integrated and regression testing strategies are used in this application for testing.



Test Case Id	Test Scenario	Expected Result	Actual Result	Pass/Fail
TO1	Check whether jupyter notebook is installed	Jupyter notebook should be opened after executing command	As expected	Pass
TO2	Check if all the packages are installed	Error should not be displayed	As expected	Pass
TO3	Check if all the modules are correctly imported	Error should not be displayed	As expected	Pass
TO4	Check for empty input	Warning message should be given	As expected	Pass
TO5	Check for string input	Warning message should be given	As expected	Pass
TO6	Check for out of range input	Warning message should be given	As expected	Pass
TO7	Check whether button is working	Should give result	As expected	Pass
TO8	Check whether getting correct output	It should correctly predict output	As expected	Pass

Figure 4.3 Test Cases

4.4 EXECUTION SCREENSHOTS



Anaconda Prompt (Anaconda3) - python app.py

```
(base) C:\Users\prase>cd
C:\Users\prase

(base) C:\Users\prase>cd C:\Users\prase\OneDrive\Desktop\project\NLP based Machine Learning Approaches for Text Summarization

(base) C:\Users\prase\OneDrive\Desktop\project\NLP based Machine Learning Approaches for Text Summarization>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 112-963-841
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [25/May/2021 17:43:47] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/css/responsive-tabs/responsive-tabs.min.css HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/css/font-awesome/css/font-awesome.min.css HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/js/jquery.js HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/css/bootstrap/bootstrap.css HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/css/animate/animate.min.css HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/css/style.css HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/js/bootstrap/bootstrap.js HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/js/wow/wow.min.js HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/js/script.js HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/js/responsive-tabs/jquery.responsiveTabs.min.js HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/video/solo.jpg HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/images/about/gensim2.jpg HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/images/about/NLTK.jpg HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/images/about/sumy.jpg HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/images/about/spacy.jpg HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/images/preloader/preloader.gif HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:48] "GET /static/video/solo.mp4 HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:49] "GET /static/css/font-awesome/fonts/fontawesome-webfont.woff2?v=4.7.0 HTTP/1.1" 200 -
127.0.0.1 - - [25/May/2021 17:43:49] "GET /static/images/favicon/favicon.ico HTTP/1.1" 200 -
```

Figure 4.4.1 Execution page

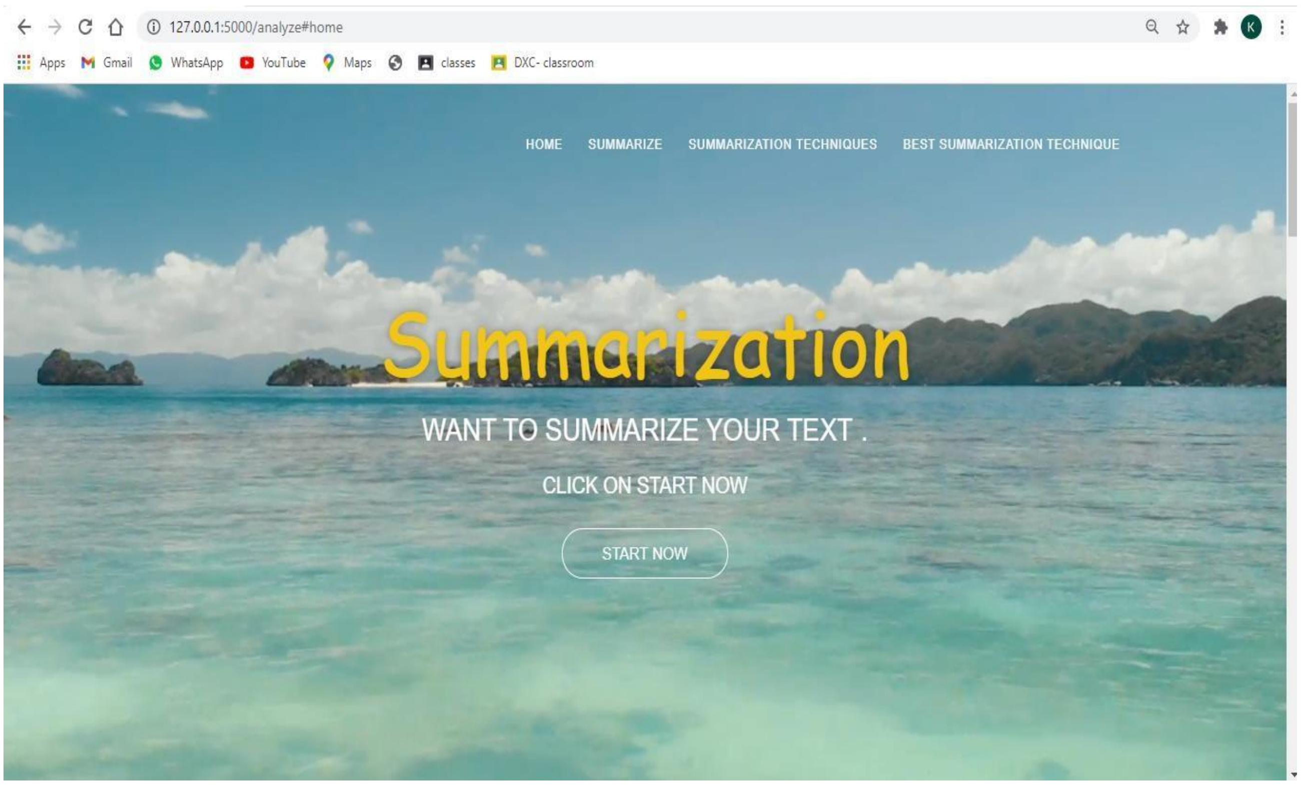


Figure 4.4.2 Dashboard

Text Summarization Method

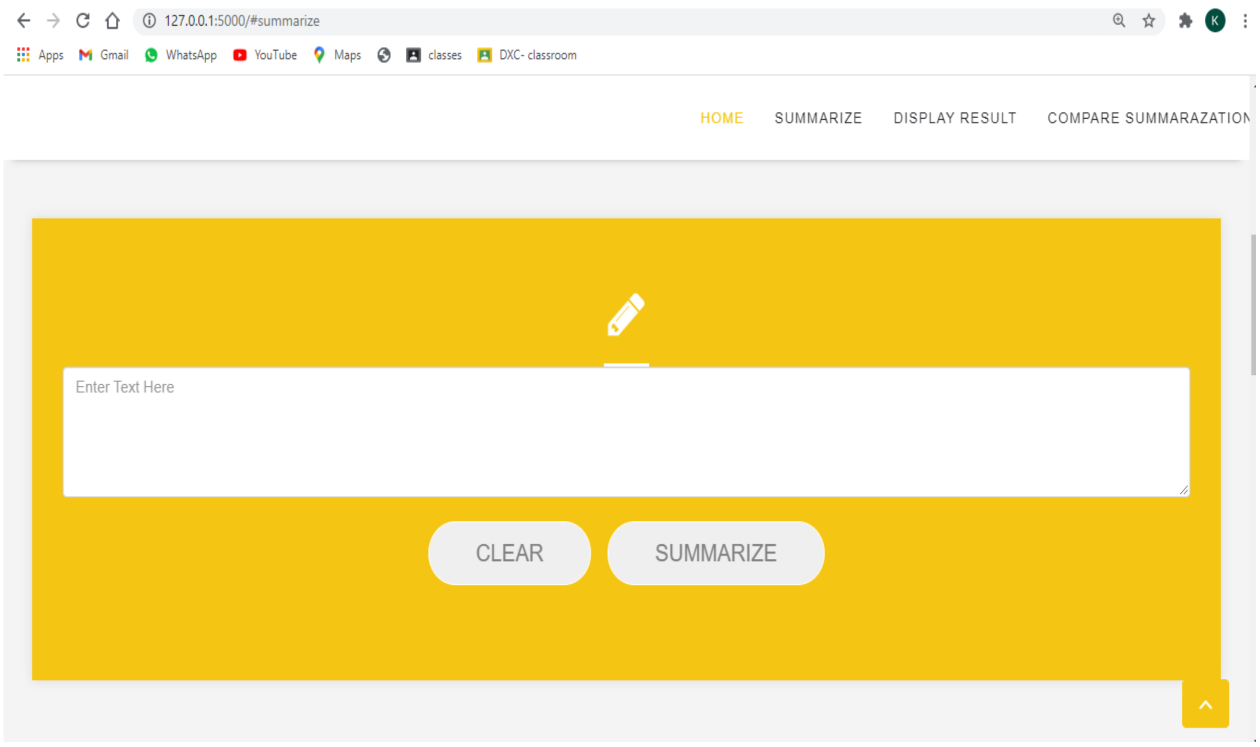
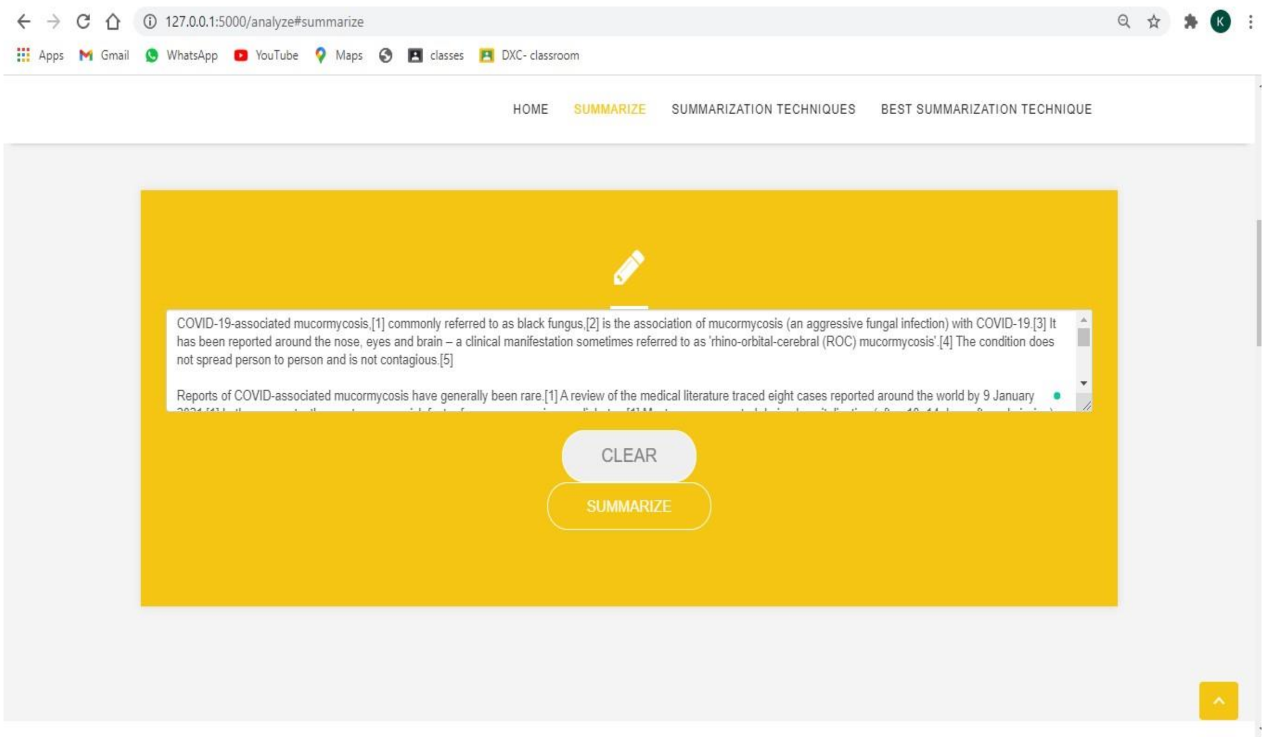


Figure 4.4.3 Home Page

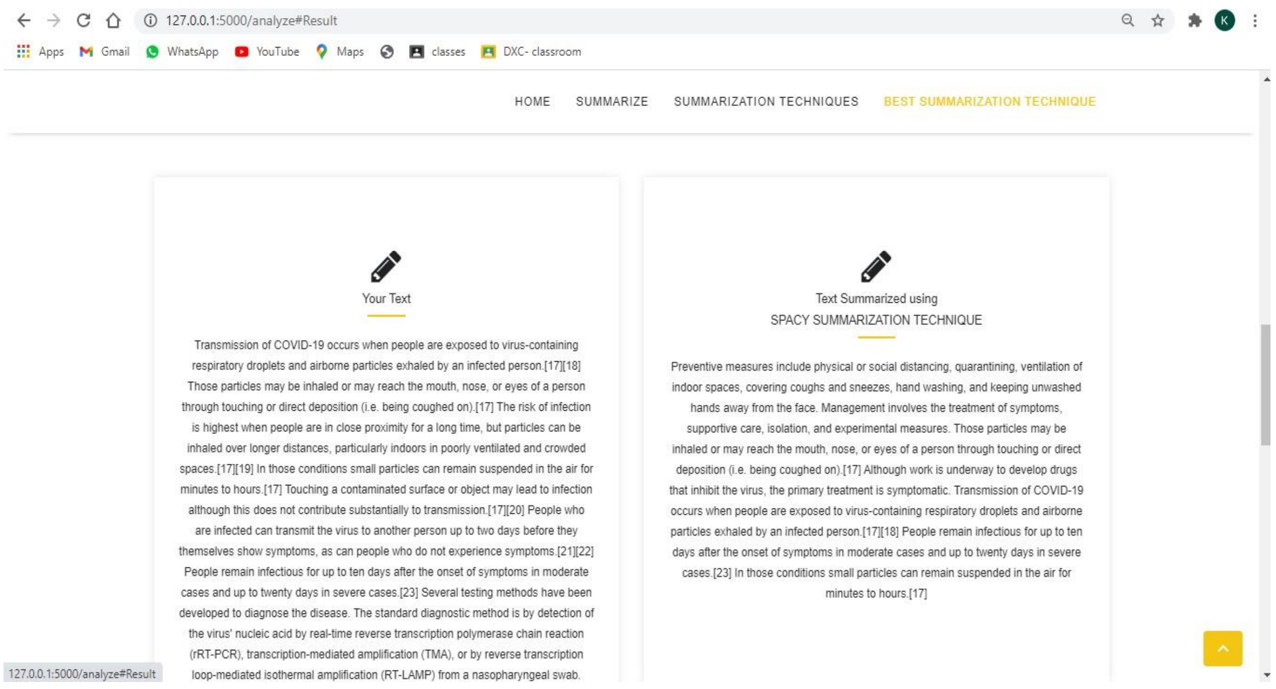


COVID-19-associated mucormycosis,[1] commonly referred to as black fungus,[2] is the association of mucormycosis (an aggressive fungal infection) with COVID-19.[3] It has been reported around the nose, eyes and brain – a clinical manifestation sometimes referred to as 'rhino-orbital-cerebral (ROC) mucormycosis'.[4] The condition does not spread person to person and is not contagious.[5]

Reports of COVID-associated mucormycosis have generally been rare.[1] A review of the medical literature traced eight cases reported around the world by 9 January

Figure 4.4.4 Text Page

Text Summarization Method


127.0.0.1:5000/analyze#Result
Apps
Gmail
WhatsApp
YouTube
Maps
classes
DXC- classroom
HOME
SUMMARIZE
SUMMARIZATION TECHNIQUES
BEST SUMMARIZATION TECHNIQUE
127.0.0.1:5000/analyze#Result

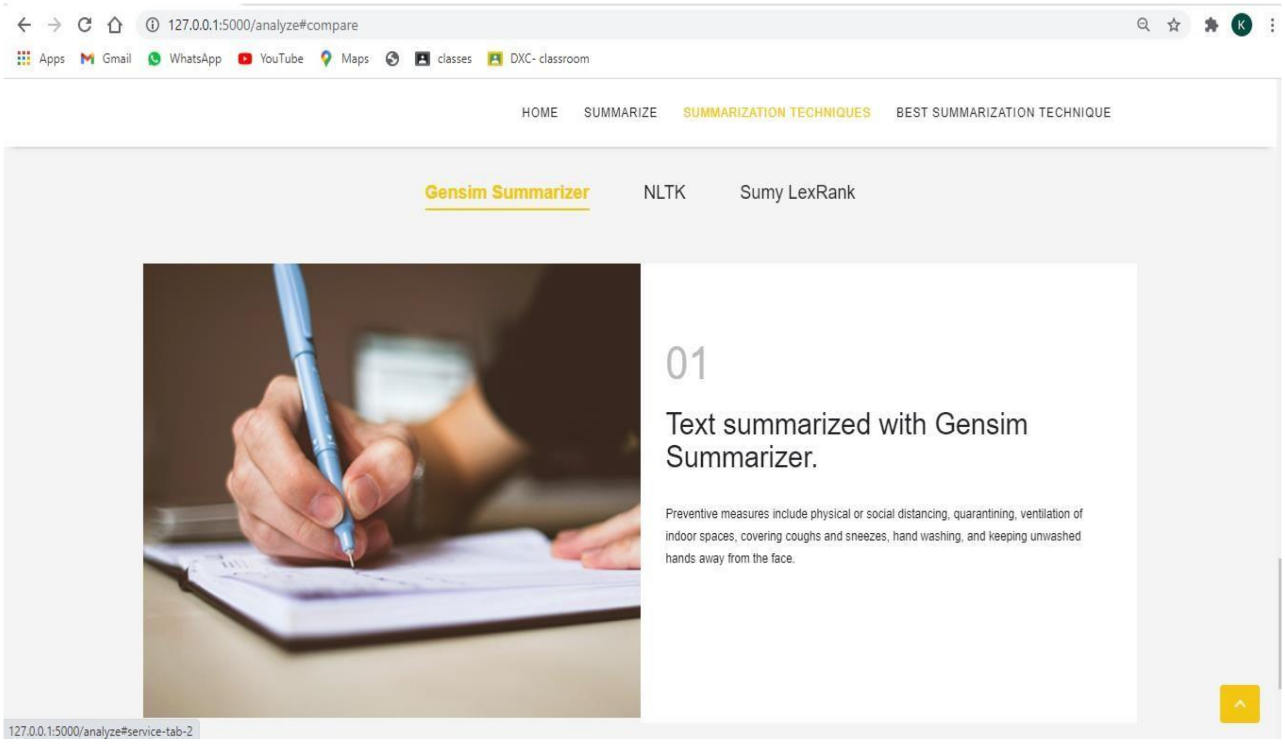
Your Text

Transmission of COVID-19 occurs when people are exposed to virus-containing respiratory droplets and airborne particles exhaled by an infected person.[17][18] Those particles may be inhaled or may reach the mouth, nose, or eyes of a person through touching or direct deposition (i.e. being coughed on).[17] The risk of infection is highest when people are in close proximity for a long time, but particles can be inhaled over longer distances, particularly indoors in poorly ventilated and crowded spaces.[17][19] In those conditions small particles can remain suspended in the air for minutes to hours.[17] Touching a contaminated surface or object may lead to infection although this does not contribute substantially to transmission.[17][20] People who are infected can transmit the virus to another person up to two days before they themselves show symptoms, as can people who do not experience symptoms.[21][22] People remain infectious for up to ten days after the onset of symptoms in moderate cases and up to twenty days in severe cases.[23] Several testing methods have been developed to diagnose the disease. The standard diagnostic method is by detection of the virus' nucleic acid by real-time reverse transcription polymerase chain reaction (rRT-PCR), transcription-mediated amplification (TMA), or by reverse transcription loop-mediated isothermal amplification (RT-LAMP) from a nasopharyngeal swab.

Text Summarized using
SPACY SUMMARIZATION TECHNIQUE

Preventive measures include physical or social distancing, quarantining, ventilation of indoor spaces, covering coughs and sneezes, hand washing, and keeping unwashed hands away from the face. Management involves the treatment of symptoms, supportive care, isolation, and experimental measures. Those particles may be inhaled or may reach the mouth, nose, or eyes of a person through touching or direct deposition (i.e. being coughed on).[17] Although work is underway to develop drugs that inhibit the virus, the primary treatment is symptomatic. Transmission of COVID-19 occurs when people are exposed to virus-containing respiratory droplets and airborne particles exhaled by an infected person.[17][18] People remain infectious for up to ten days after the onset of symptoms in moderate cases and up to twenty days in severe cases.[23] In those conditions small particles can remain suspended in the air for minutes to hours.[17]

Figure 4.4.5 Summarized Text

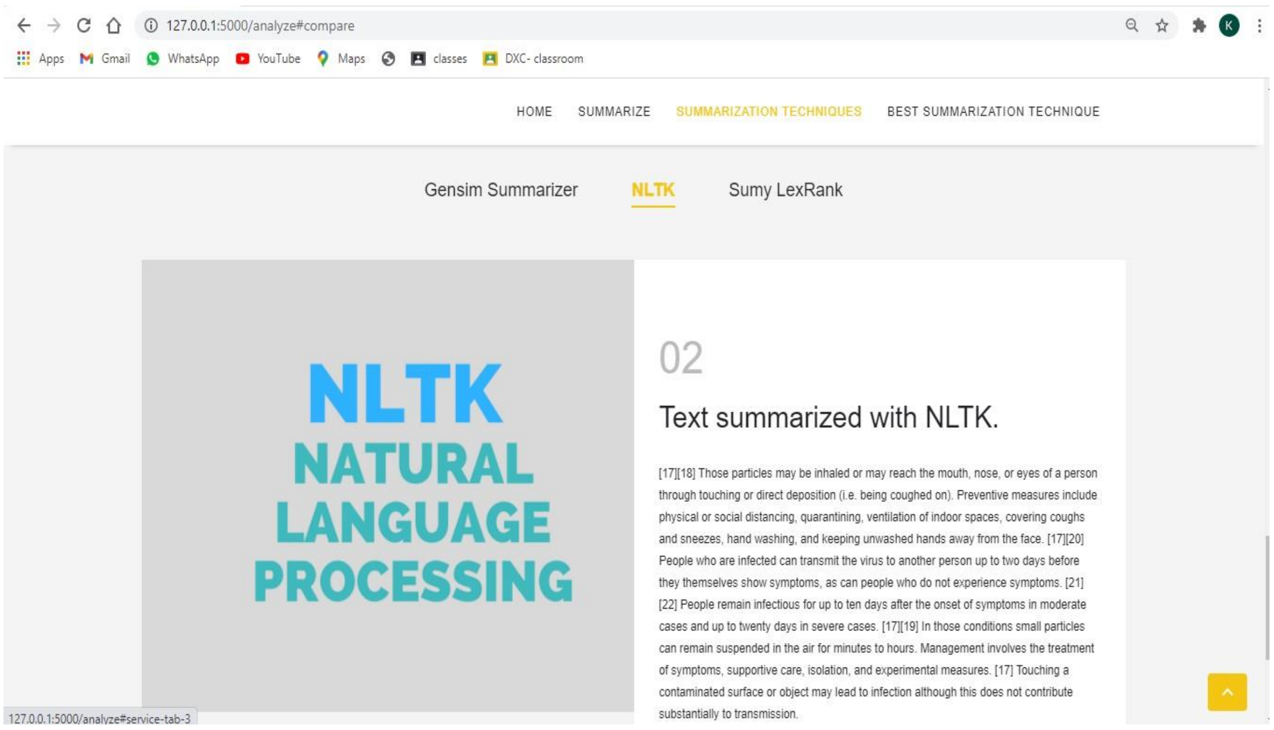


01

Text summarized with Gensim Summarizer.

Preventive measures include physical or social distancing, quarantining, ventilation of indoor spaces, covering coughs and sneezes, hand washing, and keeping unwashed hands away from the face.

Figure 4.4.6 Gensim Summarizer

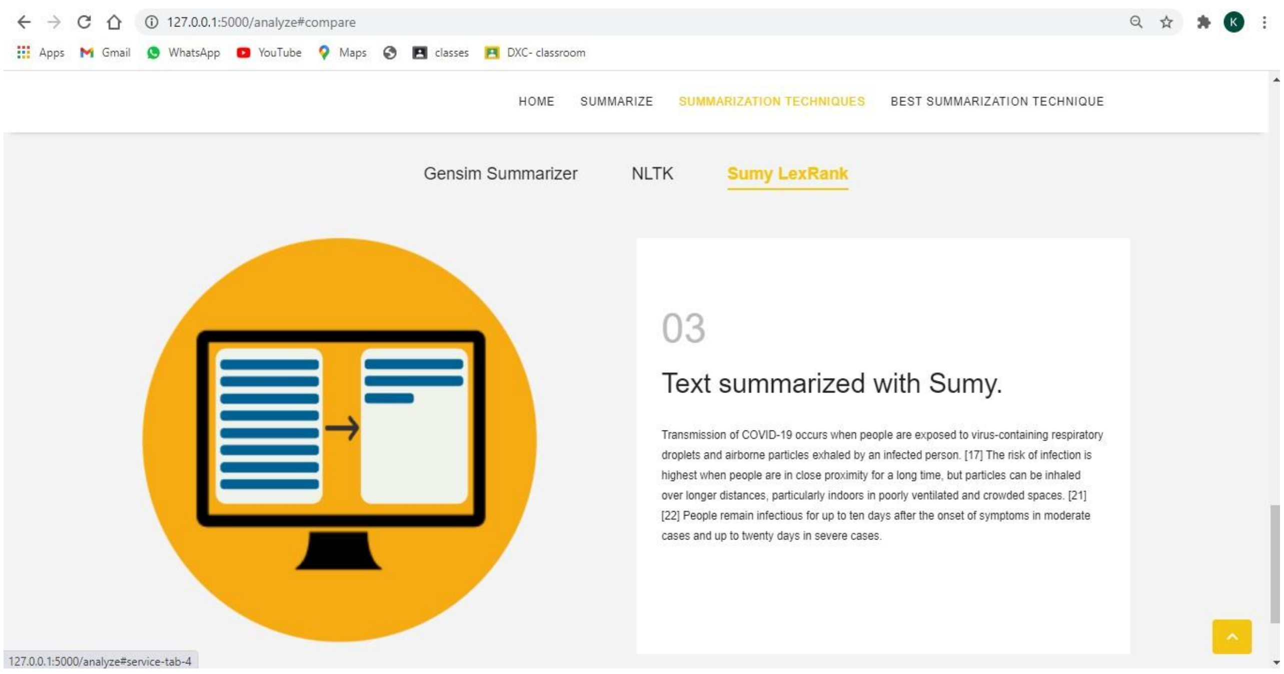

127.0.0.1:5000/analyze#compare
Apps Gmail WhatsApp YouTube Maps classes DXC- classroom
HOME SUMMARIZE SUMMARIZATION TECHNIQUES BEST SUMMARIZATION TECHNIQUE
Gensim Summarizer NLTK Sumy LexRank
NLTK NATURAL LANGUAGE PROCESSING
127.0.0.1:5000/analyze#service-tab-3

02

Text summarized with NLTK.

[17][18] Those particles may be inhaled or may reach the mouth, nose, or eyes of a person through touching or direct deposition (i.e. being coughed on). Preventive measures include physical or social distancing, quarantining, ventilation of indoor spaces, covering coughs and sneezes, hand washing, and keeping unwashed hands away from the face. [17][20] People who are infected can transmit the virus to another person up to two days before they themselves show symptoms, as can people who do not experience symptoms. [21][22] People remain infectious for up to ten days after the onset of symptoms in moderate cases and up to twenty days in severe cases. [17][19] In those conditions small particles can remain suspended in the air for minutes to hours. Management involves the treatment of symptoms, supportive care, isolation, and experimental measures. [17] Touching a contaminated surface or object may lead to infection although this does not contribute substantially to transmission.

Figure 4.4.7 NLTK Summarizer


127.0.0.1:5000/analyze#compare
Apps Gmail WhatsApp YouTube Maps classes DXC- classroom
HOME SUMMARIZE SUMMARIZATION TECHNIQUES BEST SUMMARIZATION TECHNIQUE
Gensim Summarizer NLTK Sumy LexRank
127.0.0.1:5000/analyze#service-tab-4

03

Text summarized with Sumy.

Transmission of COVID-19 occurs when people are exposed to virus-containing respiratory droplets and airborne particles exhaled by an infected person. [17] The risk of infection is highest when people are in close proximity for a long time, but particles can be inhaled over longer distances, particularly indoors in poorly ventilated and crowded spaces. [21][22] People remain infectious for up to ten days after the onset of symptoms in moderate cases and up to twenty days in severe cases.

Figure 4.4.8 Sumy LexRank

5. CONCLUSION AND FUTURE SCOPE

We have seen that due to abundant availability of data, text summarization has a very vital role in saving user's time, as well as resources. Text summarization is indeed an important tool for today. We have seen the use of various algorithms and methods for this purpose. These methods, in individual and together give different types of summaries. Their accuracy score can be compared to find the better and more concise summaries. The summaries generated using these methods are not always up to the mark. Sometimes, it's also irrelevant to the original document. Therefore, this topic is ongoing and people have done various works on this. There isn't any specific model that generates best summaries.

In future work, we plan to investigate machine learning techniques to incorporate additional features for the improvement of generic text summarization quality.

As part of the large-scale video content summarization project, we also plan to investigate how image and audio acoustic features extracted from video programs can help to improve the text summarization quality.

6. REFERENCES

- https://www.irjet.net/archives/V7/i4/IRJET-V7I4647.pdf
- https://www.researchgate.net/publication/317420253_A_survey_on_extractive_text_summarization
- https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f
- https://www.aclweb.org/anthology/2020.acl-main.120.pdf