# TEXT SUMMARISER

MINOR PROJECT REPORT

By

**SANJEEV SITARAMAN (RA2211026010355)**

**GAURI GUPTA (RA2211026010359)**

**MRINALINI VAISH (RA2211026010365)**

Under the guidance of   
**Dr. Maivizhi R***In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

in Department of Computational Intelligence



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203P** **ADVANCED PROGRAMMING PRACTICE** entitled in "**Text Summarizer** " is the bonafide work of **SANJEEV SITARAMAN (RA2211026010355), GAURI GUPTA (RA2211026010359), and MRINALINI VAISH (RA2211026010365)** who carried out the work under my supervision.

# PROJECT GUIDE HEAD OF THE DEPARTMENT

Dr Maivizhi R Dr Annie Uthra

# Assistant Professor Professor & Head

**Department of Computational Intelligence Department of Computational Intelligence**

SRM Institute of Science and Technology SRM Institute of Science and Technology

Kattankulathur Kattankulathur

# ABSTRACT

A text summarizer in Python is a project that aims to automatically generate concise and coherent summaries from longer pieces of text, such as articles, documents, or web pages. The primary purpose of this project is to save time and effort for individuals who need to quickly understand the main points of a lengthy text without reading the entire content. The goal of this project is to craft a text summarizer that aids in navigating the information overload, catering to diverse user needs, and revolutionizing the way people consume and interact with textual content.

# ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr. Maivizhi R , Assistant Professor , Department of Computational Intelligence,** for herassistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. Annie Luthra, Department of Computational Intelligence** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHAPTER NO** | | **CONTENTS** | | **PAGE NO** |
| **1** | **INTRODUCTION** | | | **1** |
|  | 1.1 Motivation | | |  |
|  | 1.2 Objective | | |  |
|  | 1.3 Problem Statement | | |  |
|  | 1.4 Challenges | | |  |
| **2** | **LITERATURE SURVEY** | | | **2** |
| **3** | **REQUIREMENT ANALYSIS** | | | **3** |
| **4** | **ARCHITECTURE & DESIGN** | | | **4** |
| **5** | **IMPLEMENTATION** | | | **5 – 7** |
| **6** | **EXPERIMENT RESULTS & ANALYSIS** | | | **8 – 9** |
| **7** | **CONCLUSION** | | | **10** |
|  |  | |

**INTRODUCTION**

1. Motivation

In an age inundated with vast amounts of textual data, the need for a streamlined and efficient method to condense information is crucial. The motivation behind the development of a text summarizer stems from the necessity to enable quicker comprehension and accessibility to pertinent content, saving time for users inundated with excessive information.

1. Objective

The primary objective of the text summarizer is to provide a tool that can automatically condense lengthy texts into concise and coherent summaries. By employing natural language processing techniques, the aim is to extract the most critical information, allowing users to grasp the main points without having to sift through entire documents or articles.

1. Problem Statement

The challenge lies in creating an algorithm that can discern the essential elements of a text, extract the main concepts, and represent them in a shorter, coherent form while preserving the original meaning. The summarizer must tackle nuances, context, and variations in writing styles to generate accurate and informative summaries.

1. Challenges

Developing a robust text summarizer involves overcoming several challenges. These include handling diverse text structures, preserving the original intent and context, addressing language intricacies, and ensuring the generated summary remains informative and coherent without losing the essence of the source material. Additionally, adapting to various domains and sources of information poses a challenge in creating a universally effective text summarization tool.

**LITERATURE SURVEY**

A literature survey for a text summarizer involves reviewing existing research papers, articles, and publications related to text summarization techniques. Here's a structured literature survey outline to get you started:

1. Extractive vs. Abstractive Summarization
2. Python Libraries for Text Summarization:
3. Gensim
4. NLTK (Natural Language Toolkit)
5. spaCy
6. sumy
7. TextRank Algorithm
8. Evaluation Metrics
9. Example Datasets

**REQUIREMENTS**

**Requirement Analysis**

1. Input Data in txt format
2. Text Preprocessing
3. Extractive Summarization:
4. Sentence tokenization
5. TF-IDF (Term Frequency-Inverse Document Frequency) for sentence importance
6. Sentence scoring
7. Sentence selection criteria
8. Summary generation
9. Evaluation Metrics
10. Integration with the Java android studio app through Gdrive authentication
11. Data Storage solutions

**Software Requirement**

1. pycharm ide
2. Operating server (here Linux)

**ARCHITECTURE AND DESIGN**

**A diagram of a company

Description automatically generated**

**IMPLEMENTATION**

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.cluster.util import cosine\_distance

import numpy as np

import networkx as nx

import scipy as sp

from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive

def read\_article(file\_name):

file = open(file\_name, "r")

filedata = file.readlines()

article = filedata[0].split(". ")

sentences = []

for sentence in article:

sentences.append(sentence.replace("[^a-zA-Z]", " ").split(" "))

sentences.pop()

return sentences

def sentence\_similarity(sent1, sent2, stopwords=None):

if stopwords is None:

stopwords = []

sent1 = [w.lower() for w in sent1]

sent2 = [w.lower() for w in sent2]

all\_words = list(set(sent1 + sent2))

vector1 = [0] \* len(all\_words)

vector2 = [0] \* len(all\_words)

for w in sent1:

if w in stopwords:

continue

vector1[all\_words.index(w)] += 1

for w in sent2:

if w in stopwords:

continue

vector2[all\_words.index(w)] += 1

return 1 - cosine\_distance(vector1, vector2)

def gen\_similarity\_matrix(sentences, stop\_words):

similarity\_matrix = np.zeros((len(sentences), len(sentences)))

for idx1 in range(len(sentences)):

for idx2 in range(len(sentences)):

if idx1 == idx2:

continue

similarity\_matrix[idx1][idx2] = sentence\_similarity(sentences[idx1], sentences[idx2], stop\_words)

return similarity\_matrix

def find\_summary\_file\_id(drive, folder\_id):

# Search for the 'summary.txt' file in the folder and return its ID if found

file\_list = drive.ListFile({'q': f"'{folder\_id}' in parents and title = 'summary.txt' and trashed=false"}).GetList()

if file\_list:

return file\_list[0]['id']

return None

def generate\_summary(file\_name, top\_n=5, folder\_id='14cj35NPauP9qni7vVXkY68Ohl9j-h0G\_'):

# Authenticate with Google Drive using OAuth 2.0

gauth = GoogleAuth()

gauth.LocalWebserverAuth()

drive = GoogleDrive(gauth)

# Find the existing 'summary.txt' file ID in Google Drive

existing\_summary\_file\_id = find\_summary\_file\_id(drive, folder\_id)

stop\_words = nltk.corpus.stopwords.words('english')

summarize\_text = []

sentences = read\_article(file\_name)

sentence\_similarity\_matrix = gen\_similarity\_matrix(sentences, stop\_words)

sentence\_similarity\_graph = nx.from\_numpy\_array(sentence\_similarity\_matrix)

scores = nx.pagerank(sentence\_similarity\_graph)

ranked\_sentence = sorted(((scores[i], s) for i, s in enumerate(sentences)), reverse=True)

top\_n = min(top\_n, len(ranked\_sentence))

summary\_text = ". ".join([" ".join(ranked\_sentence[i][1]) for i in range(top\_n)])

if existing\_summary\_file\_id:

# Update the content of the existing file

file\_drive = drive.CreateFile({'id': existing\_summary\_file\_id})

file\_drive.SetContentString(summary\_text)

file\_drive.Upload()

print("Summary has been updated in Google Drive.")

else:

# Create a new 'summary.txt' file if it doesn't exist

file\_drive = drive.CreateFile({'title': 'summary.txt', 'parents': [{'id': folder\_id}]})

file\_drive.SetContentString(summary\_text)

file\_drive.Upload()

print("Summary has been saved to Google Drive.")

generate\_summary("msft.txt", 2, '14cj35NPauP9qni7vVXkY68Ohl9j-h0G\_')

**RESULTS**

* 1. **Implemented through Android Studio using Java.**

A screenshot of a application

Description automatically generated

* 1. **After access allowed to overwrite the summary.txt doc in google drive.**

A screenshot of a computer

Description automatically generated

* 1. **After successful updation.**

A line of lines on a black background

Description automatically generated

* 1. **Updated summary in gdrive.**

A screenshot of a computer

Description automatically generated

**CONCLUSION**

In conclusion, the development of a text summarizer stands as a pivotal step towards revolutionizing information accessibility. The endeavor to condense vast amounts of textual data into concise, informative summaries addresses the pressing need for efficient information processing in an age of overwhelming content. Despite the challenges posed by varied text structures, linguistic nuances, and context preservation, the pursuit of creating an effective summarization tool remains crucial.

The potential impact of a successful text summarizer is vast, offering users a time-saving solution while enabling quicker comprehension of essential content. By navigating the complexities of natural language and diverse writing styles, the summarizer aims to enhance accessibility to information across various domains, making it a valuable resource for professionals, researchers, students, and anyone seeking distilled, yet comprehensive knowledge. As technology advances, the evolution of text summarization holds promise in reshaping how we interact with, consume, and derive value from textual information.

Python