### **TEXT DOCUMENT SUMMARIZATION**

### A PROJECT REPORT

***Submitted by***

|  |  |
| --- | --- |
| **HARISH V** | **1805015** |
| **MATHIPRIYA.S** | **1805028** |
| **SOWMIYA S** | **1805056** |
| **ROBERO** | **1805043** |

***in partial fulfilment for the award of the degree of***

BACHELOR OF ENGINEERING

***in***

## COMPUTER SCIENCE AND ENGINEERING



**COIMBATORE INSTITUTE OF TECHNOLOGY**

***(Government Aided Autonomous Institution Affiliated Anna University)***

## COIMBATORE-641014

### ANNA UNIVERSITY - CHENNAI 600 025

**APRIL-2021**

## **BONAFIDE CERTIFICATE**

Certified that this project “TEXT DOCUMENT SUMMARIZATION” is the bonafide work of **HARISH V (1805015), MATHIPRIYA S (1805028), SOWMIYA (1805056), ROBERO (1805043)** under my supervision during the academic year 2020-2021.

|  |
| --- |
| **Dr.G. Kousalya, M.E., Ph.D.**  **PROFESSOR,** HEAD Department of CSE,  Coimbatore Institute of Technology,  Coimbatore - 641014. |

Certified that the candidates were examined by us in the project work viva-voce examination held on ………………

### Internal Examiner                                                                 External Examiner

**PLACE:**

**DATE:**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER No.** | **CHAPTER NAME** | **PAGE No.** |
|  | **ACKNOWLEDGEMENT** | **5** |
|  | **ABSTRACT** | 6 |
| **1** | **INTRODUCTION** |  |
|  | **1.1 Problem Statement** | 7 |
|  | **1.2** **Deep Learning** | 7 |
|  | **1.3 Text Summarization** | 7 |
|  | **1.4** **Transformers** | 8 |
|  | **1.5** **BERT** | 8 |
|  | **1.6 Purpose of the project** | 9 |
|  | **1.7** **Scope of the project** | 9 |
| **2** | **LITERATURE SURVEY** | **10** |

1. **SYSTEM SPECIFICATION 13**
   1. [**Hardware Specification**](http://../../../C:/Users/Pc/Downloads/REPORT%20INTRO1.docx#_TOC_250018)  **13**

**3.2** [**Software Specification**](http://../../../C:/Users/Pc/Downloads/REPORT%20INTRO1.docx#_TOC_250017)  **13**

1. **SYSTEM DESIGN 14**

**4.1** [**Architecture**](http://../../../C:/Users/Pc/Downloads/REPORT%20INTRO1.docx#_TOC_250014)  **14**

**4.1.1. System Architecture 14**

**4.1.2. Model Architecture 14**

**4.1.3 BERT Architecture 15**

**4.1.4 Architecture of Training Model 16**

**4.2 Activity Diagram 16**

1. **IMPLEMENTATION 17**

**5.1 Data Collection and Preprocessing 17**

**5.1.1 Dataset 17**

**5.1.2 Extraction of text from PDF 17**

**5.1.3** [**Data Preprocessing**](http://../../../C:/Users/Pc/Downloads/REPORT%20INTRO1.docx#_TOC_250009)  **17**

**5.1.4 Methodology 18**

**5.2 Extractive Summarization with BERT 18**

**5.3 Fine tuning with summarization layers 19**

**5.4 Summarization 19**

**5.5 Libraries 19**

**5.5 User Interface 20**

**5.6 Back End – Python Flask 20**

1. **EVALUATION METRICES 21**
2. **APPLICATION FOR THE SOCIETY                                   23**
3. **CONCLUSION AND FUTURE WORK 24**
4. **REFERENCES                                                                   25**
5. **APPENDIX                                                           26**

**1.APPENDIX A – SNAPSHOTS OF OUTPUT 26**

**2.APPENDIX B – SOURCE CODE 27**

## **ACKNOWLEDGEMENT**

Our Project, “**TEXT DOCUMENT SUMMARIZATION**” has been the result of motivation and encouragement from many, whom we would like to thank.

We take this opportunity to express our sincere thanks to our Secretary and Professor Emeritus, **Dr.R.Prabhakar, B.Tech, M.S., Ph.D.,** and our Principal **Dr.V.Selladurai, M.E., Ph.D.,** for providing us the necessary facilities and support for successful completion of this project.

We would like to sincerely thank our Head of the Department, as well as our guide **Dr.G.Kousalya, M.E., Ph.D.,** for her sustained support and guidance which helped in the timely completion of the project.

We sincerely thank all our faculty members for the advice and support in various dimensions of the project work. We present our gratefulness to the Lab Assistants and other non teaching staff for their timely support and assistance in the laboratory. Finally, we like to thank our families and friends for their appreciation and guidance.

**ABSTRACT**

In this new era, where a large amount of information is available on the Internet, it becomes really important to provide an improved mechanism to extract the information quickly and most efficiently. The World Wide Web has brought us a vast amount of on-line information. Every time someone searches something on the Internet, the response obtained is lots of different Web pages with many information, which is impossible for a person to read completely. There is a rapid increase in generation of textual data on a daily basis, be it in healthcare sector, academia, government offices or the corporate sector, the task of maintaining the enormous amount of textual data and processing the data to gain correct and valuable information has become more time and resource intensive. It is very difficult for human beings to manually extract the summary of a large documents of text. So, there is a problem of searching for relevant documents from the number of documents available, and absorbing relevant information from it. In order to solve the above two problems, the automatic text summarization is very much necessary. The basic idea behind summarization is finding the subset of the data which contains the information of all the set. There is a great need to reduce unnecessary data. It is very difficult to summarize the document manually so there is the great need of automatic methods. Our goal with this project is to come up with deep learning model (based on extractive summarization). The most daunting task is to come up with an efficient scoring algorithm which would produce a better output for even a wide range of text.

**1.INTRODUCTION**

**1.1 PROBLEM STATEMENT**

In recent years, we are witnessing the amount of textual information is increasing day by day. It becomes more difficult for the user to read the textual information and it may lead to loss of interest. Everyday corona cases are increasing, and it is extremely difficult for the government to keep track of these reports. In case researchers, students, reading an entire research paper is difficult and time consuming. So the problem is to reduce their time and effort involved in reading long text documents but at the same time getting the important information from that document. Summarization can solve this problem.

**1.2 DEEP LEARNING**

Every day the amount of textual information in the internet, news articles, blogs, research papers are increasing. Most of the times the data is unstructured and the best way to go through it is to just skim those text. The best solution to reduce the effort and time taken in reading a long document is summarization, which can be done by deep learning. Deep learning is a subset of machine learning, which itself falls within the field of artificial intelligence. Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. With deep learning, computer model learns to perform classification tasks directly from images, text, or sound and can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.

**1.3 TEXT SUMMARIZATION**

Nowadays, the significance of text summarization accomplishes more attention due to data inundation on the web. Hence this information overwhelms yields in the big requirement for more reliable and capable progressive text summarizers which finds its application in various fields. Text Summarization is the process of fetching the essential information from a text document. Text Summarization techniques are classified into abstractive and extractive summarization. Extractive summaries are created by reusing the portions (words, sentences) of input text document. The system extracts text from the entire collection, without modifying the text document. Abstractive summaries provide own summary over input text without using same word or sentence of input text. It is the short meaning of each element. The ideal of automatic summarization work is to develop techniques by which a machine can generate summarize that successfully imitate summaries generated by human beings.

**1.4 TRANSFORMERS**

Traditionally recurrent neural networks and their variants have been used extensively for Natural Language Processing problems. In recent years, transformers have outperformed most RNN models. The Transformer architecture excels at handling text data which is inherently sequential. They take a text sequence as input and produce another text sequence as output. At its core, it contains a stack of Encoder layers and Decoder layers. The Encoder stack and the Decoder stack each have their corresponding Embedding layers for their respective inputs. Finally, there is an Output layer to generate the final output. The encoder contains all the important self-attention layer that computes the relationship between different words in the sequence, as well as feed forward layer. In addition to this a second Encoder-Decoder attention layer. Main reason for the Transformer’s ground-breaking performance is its use of Attention. Transformers are very versatile and are used for most NLP tasks such as language models and text classification.

**1.5 BERT**

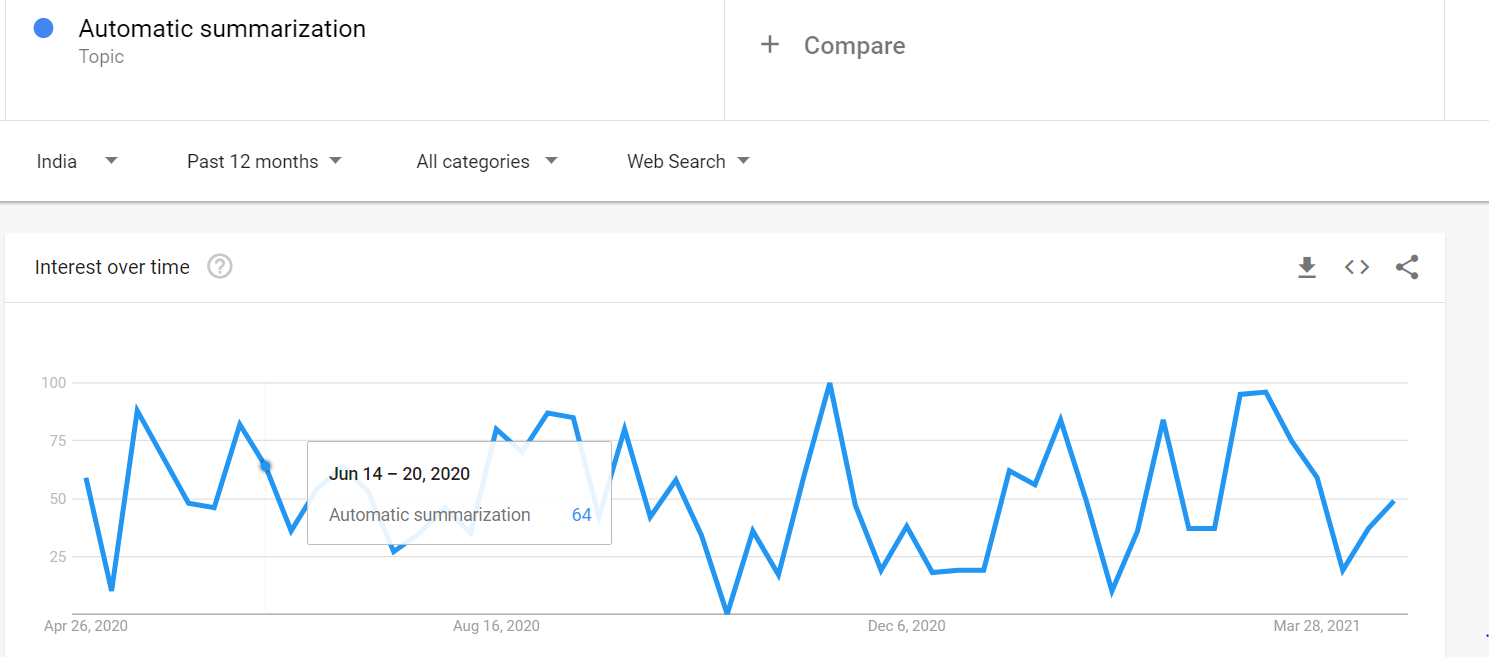
BERT (Bidirectional Encoder Representations from Transformers) is a Natural Language Processing Model proposed by researchers at Google Research in 2018. One of the main reasons for the good performance of BERT on different NLP tasks was the use of **Semi-Supervised Learning**. This means the model is trained for a specific task that enables it to understand the patterns of the language. After training the model (BERT) has language processing capabilities that can be used to empower other models that we build and train using supervised learning.

**1.6 PURPOSE OF THE PROJECT**

Research papers, articles, long documents are often long and descriptive. Analysing these documents manually is really tiresome and make take lots of effort. This is where our system can be applied to generate an automatic summary for long documents. This can definitely help every user to get the important points from the long document at a minimal time and without getting frustrated.

**1.7 SCOPE OF THE PROJECT**

Students, Researchers who needs to refer any research papers, an organisation which needs to have a crisp of their customer’s reviews, people who have to read any news articles are the typical scope of the project. But anyone who wants to read a long document can find this system very effective.



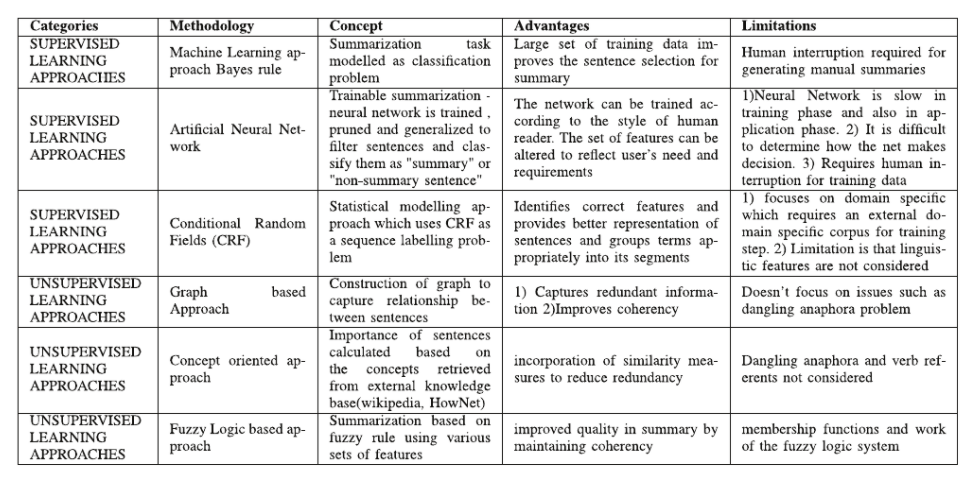
**2.LITERATURE SURVEY**

**2.1** **TITLE: A SURVEY ON EXTRACTIVE TEXT SUMMARIZATION**

AUTHORS: [N. Moratanch](https://ieeexplore.ieee.org/author/37085859698), [S. Chitrakala](https://ieeexplore.ieee.org/author/38190051700)

PUBLISHED IN: [2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)](https://ieeexplore.ieee.org/xpl/conhome/7937566/proceeding)

FEATURES:



**2.2 TITLE: AUTOMATIC TEXT SUMMARIZATION BASED ON SENTENCES CLUSTERING AND EXTRACTION**

AUTHORS: [Pei-ying Zhang](https://ieeexplore.ieee.org/author/37894765100)m, [Cun-he Li](https://ieeexplore.ieee.org/author/37897393500)

FEATURES:

1) It proposes a sentence similarity computing method based on the three features of the sentences, on the base of analyzing of the word form feature, the word order feature and the semantic feature, using the weight to describe the contribution of each feature of the sentence, describes the sentence similarity more preciously.

(2) It gives an approach of text summarization based on the sentences clustering.

EVALUATION METRICES USED:

The first one is by precision (P), recall (R) and F1- measure which are widely used in Information Retrieval.

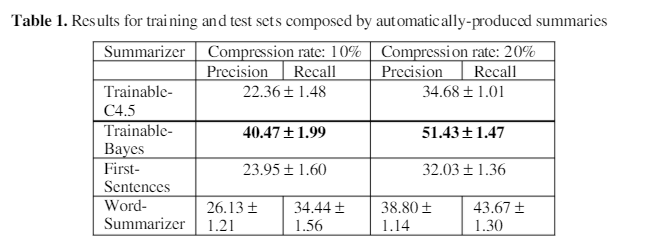
The second measure used is the ROUGE method for evaluation, which was adopted by DUC for automatically summarization evaluation.

# 2.3 TITLE: AUTOMATIC TEXT SUMMARIZATION USING A MACHINE LEARNING APPROACH

AUTHORS: [Joel Larocca Neto](https://www.researchgate.net/profile/Joel_Neto), [Alex Freitas](https://www.researchgate.net/profile/Alex_Freitas2), [Celso A. A. Kaestner](https://www.researchgate.net/profile/Celso_Kaestner)

FEATURES:

The approaches used here are Naive Bayes and C4.5 (a decision-tree algorithm)



Best results were obtained with Naive Bayes classifier for both compression rates, using the same features. But with the C4.5 as classifier, the obtained results were poor: the results are similar to the First-Sentences and Word Summarizer baselines.

# 2.4 TITLE: TEXT SUMMARIZATION USING DEEP NEURAL NETWORKS

AUTHORS: Shivam Duseja

FEATURES:

The approach used here is seq2seq (encoder-decoder architecture) model with a simple dot product attention.

Evaluation Metrices used:

**ROUGE (Recall-Oriented Understanding for Gisting Evaluation):**

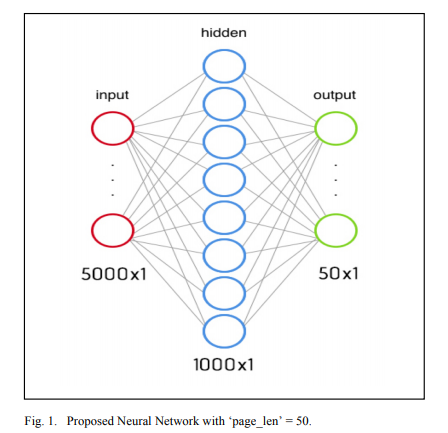
**BLEU (Bilingual Evaluation Understudy)**

**2.5 TITLE: EXTRACTIVE TEXT SUMMARIZATION USING NEURAL NETWORKS**

AUTHORS: Aakash Sinha, Abhishek Yadav, Akshay Gahlot

FEATURES:

The proposed model is based on a neural network which consists of one input layer, one hidden layer, and one output layer. The document is fed to the input layer, computations are carried in the hidden layer and an output is generated at the final layer.



* 1. **SYSTEM SPECIFICATION**

The hardware and software for the system is selected by considering the factors such as CPU processing speed, peripheral channel speed, printer speed, seek time, relational delay of hard disk and communication speed etc. The hardware and software specifications are as follows.

**3.1 HARDWARE SPECIFICATION**

|  |
| --- |
| Processor: i3 or Greater |
| Speed: 175 GHZ |
| RAM: 4 GB |
| Monitor: Quality Product |
| Keyboard: Quality Product |
| Mouse: Quality Product |

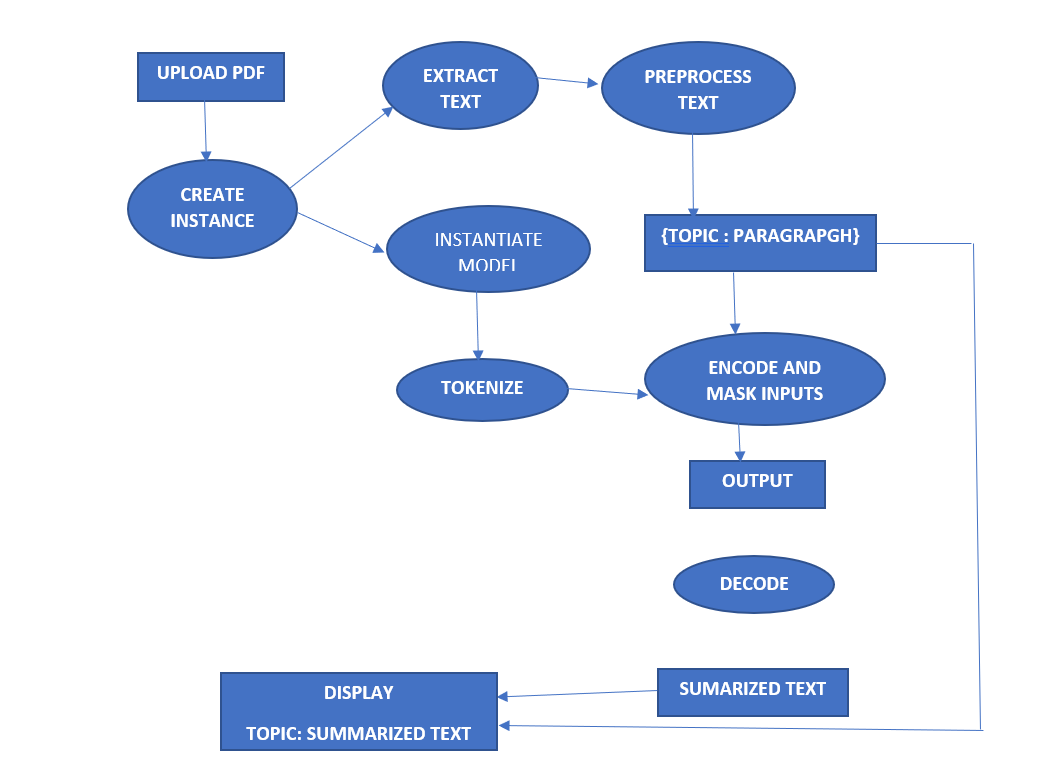
**3.2 SOFTWARE SPECIFICATION**

|  |
| --- |
| Operating system: Windows (7 or higher), Linux or Max OS |
| Language: Python |
| IDE: Jupyter notebook |

* 1. **SYSTEM DESIGN**

**4.1 ARCHITECTURE**

**4.1.1 System Architecture**

****

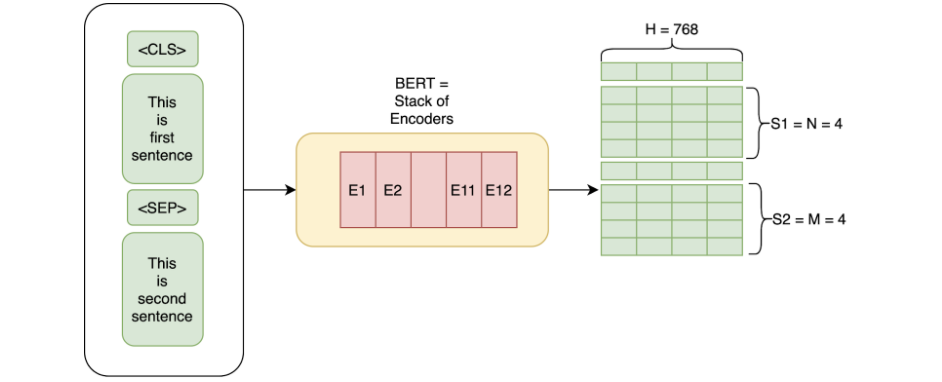
**4.1.2 Model Architecture**

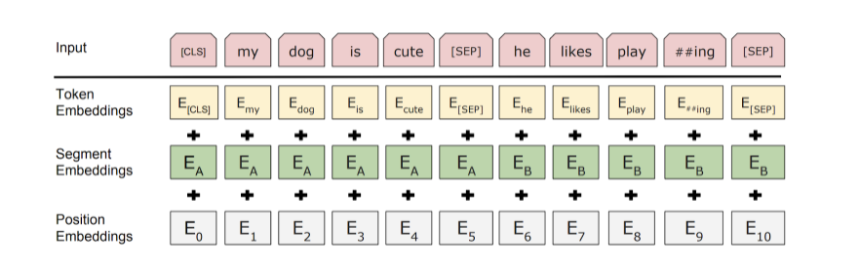
**Diagram

Description automatically generated**

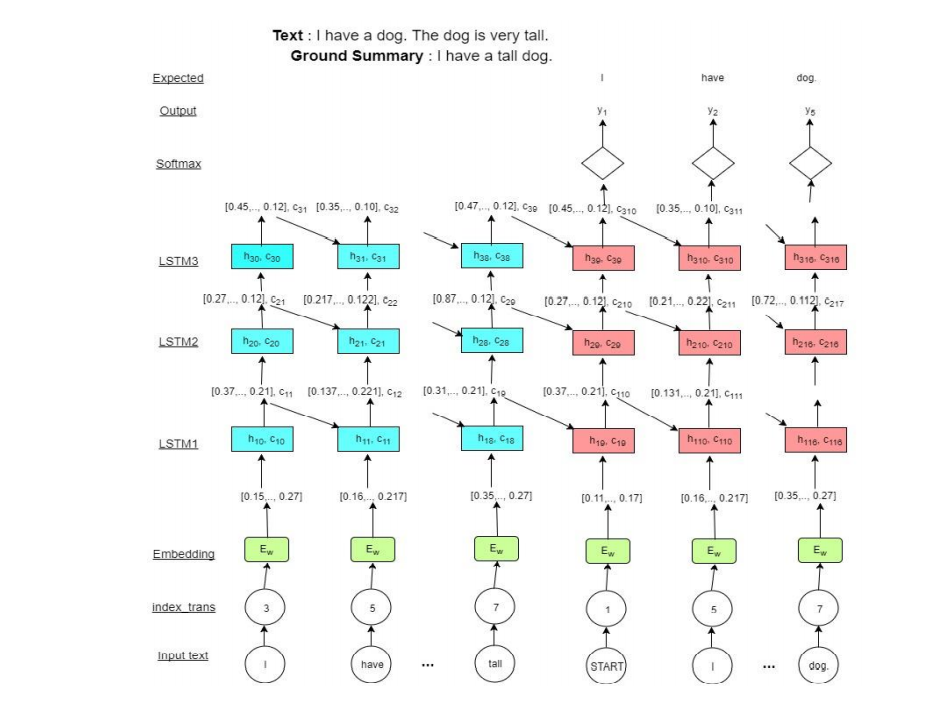
**4.1.3 BERT Architecture**

BERT is essentially just made up of stacked up encoder layers. We give inputs to BERT using the above structure. The input consists of a pair of sentences, called sequences, and two special tokens: [CLS] and [SEP]. We then get the Token embeddings by indexing a Matrix of size 30000x768(H). Here, 30000 is the Vocab length after word piece tokenization. The weights of this matrix would be learned while training. Final the input given to BERT is **Token Embeddings + Segment Embeddings + Position Embeddings.** We try to predict each word of the input sequence using our training data with Cross-Entropy loss. We mask 15% random words in each training input sequence and just predict output for those words. we will replace any word in 20% of those masked tokens by some random word.

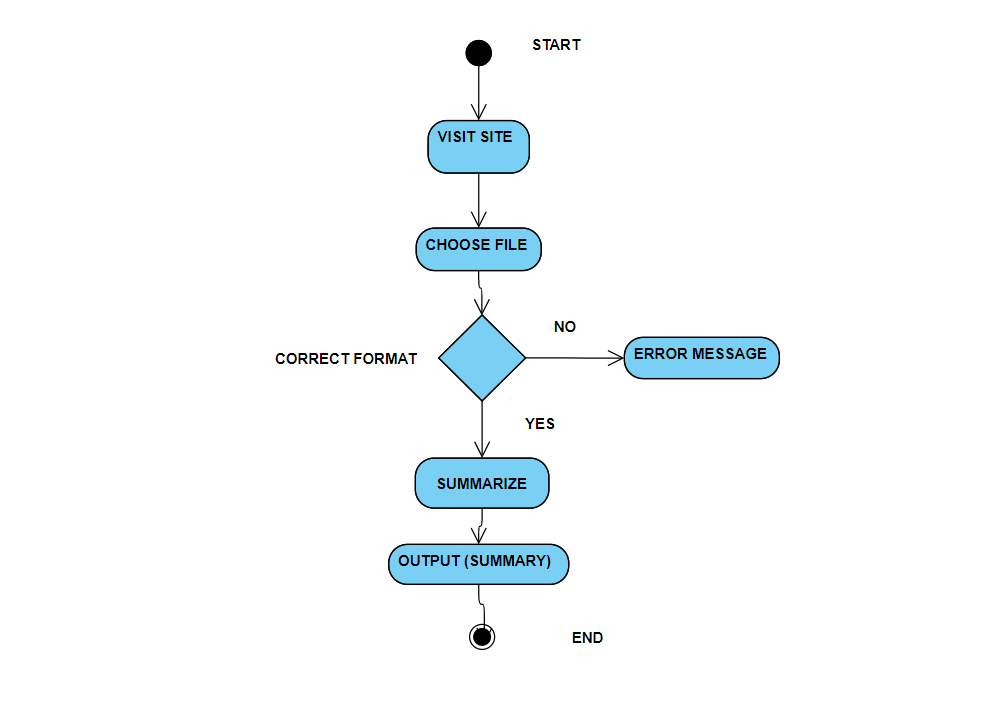




**1.1.4 Architecture of Training model**



**4.2 ACTIVITY DIAGRAM**

****

* 1. **IMPLEMENTATION**

**5.1 DATA COLLECTION AND PREPROCESSING:**

**5.1.1 Dataset**

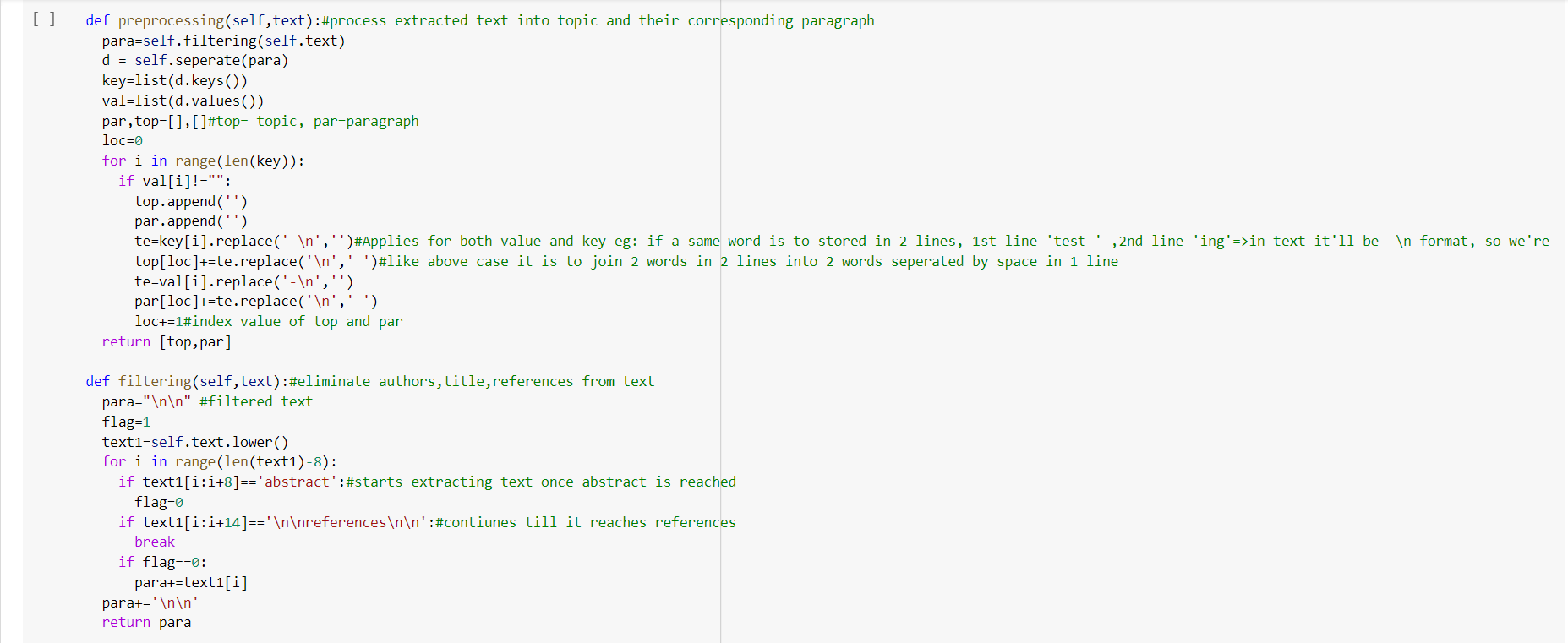
The dataset chosen is COVID – 19 Open Research Dataset Challenge (CORD - 19) from Kaggle. In response to the COVID-19 pandemic, the White House and a coalition of leading research groups have prepared the COVID-19 Open Research Dataset (CORD-19). CORD-19 is a resource of over 400,000 scholarly articles, including over 150,000 with full text, about COVID-19, SARS-CoV-2, and related coronaviruses. This freely available dataset is provided to the global research community to apply recent advances in natural language processing and other AI techniques to generate new insights in support of the ongoing fight against this infectious disease. There is a growing urgency for these approaches because of the rapid acceleration in new coronavirus literature, making it difficult for the medical research community to keep up.

**5.1.2 Extraction of Text from PDF**

This is done using PDFminer package in python. Unlike other PDF-related tools, it focuses entirely on getting and analysing text data. PDFMiner allows one to obtain the exact location of text in a page, as well as other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats (such as HTML).

**5.1.3 Data Preprocessing**

Once the text has been extracted, the following preprocessing steps were done. The entire text is separated as topic and paragraph and stored as key value pairs in python dictionary. Filtering is done to eliminate author name, references, page no from the text, punctuations, any specific marks like IEEE.



**5.1.4 Methodology**

Let d denote a document containing several sentences [sent1, sent2, · · ·, sentm], where senti is the i-th sentence in the document. Extractive summarization can be defined as the task of assigning a label yi ∈ {0, 1} to each senti , indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document.

**5.2 Extractive Summarization with BERT**

To use BERT for extractive summarization, we require it to output the representation for each sentence. However, since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences. Meanwhile, although BERT has segmentation embeddings for indicating different sentences, it only has two labels (sentence A or sentence B), instead of multiple sentences as in extractive summarization. Therefore, we modify the input sequence and embeddings of BERT to make it possible for extracting summaries. We modify the model by using multiple [CLS] symbols to get features for sentences ascending the symbol. Interval Segment Embeddings We use interval segment embeddings to distinguish multiple sentences within a document. For senti we will assign a segment embedding EA or EB conditioned on I is odd or even. For example, for [sent1, sent2, sent3, sent4, sent5] we will assign [EA, EB, EA, EB, EA]. The vector Ti which is the vector of the i-th [CLS] symbol from the top BERT layer will be used as the representation for senti.

**5.3 Fine-tuning with Summarization Layers**

After obtaining the sentence vectors from BERT, we build several summarization-specific layers stacked on top of the BERT outputs, to capture document-level features for extracting summaries. For each sentence senti, we will calculate the final predicted score Yˆ I . The loss of the whole model is the Binary Classification Entropy of Yˆ I against gold label Yi . These summarization layers are jointly fine-tuned with BERT.

**5.4 Summarization**

The model that has been used here is the Vamsi/T5\_Paraphrase\_Paws. It is implemented using TensorFlow,torch,seq2seq,transformers(t5) using transformers attention mechanism and masking function wherein decoder we hide a word and try to predict it.  A Paraphrase-Generator built using transformers which takes an English sentence as an input and produces a set of paraphrased sentences. This is an NLP task of conditional text-generation. The model used here is the T5forConditionalGeneration from huggingface transformer library. This model is trained on the Google’s PAWS dataset and the model is saved in the transformer model hub of hugging face library under the name Vamsi/T5\_Paraphrase\_Paws.

**5.5 LIBRARIES**

**Pandas**

Pandas provides a flexible platform for handling data in a data frame. It contains many open-source data analysis tools written in Python, such as the methods to check missing data, merge data frames, and reshape data structure, etc

**TensorFlow**

Transformer is a Python library for transforming structures by performing atomic substitutions.  It is used for implementing machine learning and deep learning applications.

**Transformer**

Transformers provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet…) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch.

**5.6 USER INTERFACE**

For the UI part we have used HTML and CSS. The user can upload the pdf file in the input area. Summarized text will be generated by the model and the output will be displayed in the webpage.

**5.7 BACKEND – PYTHON FLASK**

To handle the web pages, we have used the flask frame. The main work for the flask is to take the input from the user via a web page (html document) and using the post method we post the file to URL (mini project/main) which is indented to the function main in flask program. After getting the pdf file we save it to the back-end storage and send the file name to the pdf text extract function. After getting the text it sends to the preprocessing code and send it to the model to summarise. The summarised text is returned to the flask application it will render the content (summarised text) into the new webpage.

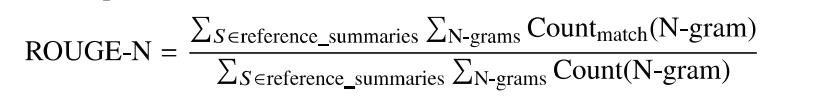
* 1. **EVALUATION METRICES**

**6.1 Human Evaluation**

Human judgment usually has wide variance on what's thought-about a "good" outline, which implies that creating the analysis method automatic is especially tough. Manual analysis is used; however, this can be each time and labour intensive because it needs humans to browse not solely the summaries however conjointly the supply documents. Other issues are those regarding coherence and coverage.

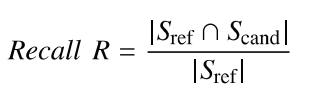
**6.2 Rouge**

Rouge Formally, ROUGE-N is an n-gram recall between a candidate summary and a set of reference summaries. ROUGE-N is computed as follows:

****

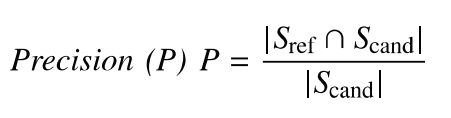
where, n stands for the length of the n-gram Count match (N- gram) is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries. Count(N-gram) is the number of N-grams in the set of reference summaries.

**6.3 Recall**

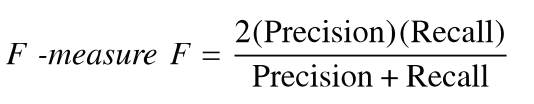


where Sref n Scand indicates the number of sentences that co-occur in both reference and candidate summaries.

**6.4 Precision**

****

**6.5 F – Measure**



**6.6 Compression Ratio**



**7.APPLICATION FOR THE SOCIETY**

With the ever-growing text data, text summarization seems to have the potential for reducing the reading time by showing summaries of the text documents that capture the key points in the original documents. Applying text summarization on each article can potentially improve customer experience and employees’ productivity in case of an organization. With the COVID-19 pandemic, there is a growing urgency for medical community to keep up with the accelerating growth in the new coronavirus-related literature. Everyday lot of articles, reports, papers about COVID-19 – 19 is released. It becomes extremely difficult for Medical community, government to keep track of this. So, it becomes extremely important to have a model which summarizes all these long documents and helps them to save effort and time. Students who are doing any projects need not read the research papers, if they use our model fully to get an essence of what is happening technically. The main aim of the goal of research is to provide the best solution to some of the world problems and also to enhance knowledge. Such researchers may have to read long documents sometimes which can be eliminated by using our system.

1. **CONCLUSION AND FUTURE WORK**

**CONCLUTION**

Extractive summarization process is highly coherent, less redundant and cohesive (summary and information rich). In this paper, we showcased how pretrained BERT can be usefully applied in extractive text summarization. We introduced a novel document-level encoder and proposed a general framework for extractive summarization. Experimental results across three datasets show that our model achieves state-of-the-art results across the board under automatic and human-based evaluation protocols.

**FUTURE WORK**

In future, the system can be implemented much faster. The system can be advanced to take more pdf as input at the same time, summarize and even find some similarities between those papers. As our model works faster with GPU, we have planned to run this cloud in the presence of GPU, So that the computation will be faster.

1. **REFERENCES**

[1] [A survey on extractive text summarization | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/7944061)

[2] [Text Summarization Method Based on Double Attention Pointer Network | IEEE Journals & Magazine | IEEE Xplore](https://ieeexplore.ieee.org/document/8955869)

[3] [An overview of Text Summarization techniques | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/abstract/document/7860024)

[4] [Text Summarization – IJERT](https://www.ijert.org/text-summarization)

[5] [Advancement of Text Summarization Using Machine Learning and Deep Learning: A Review | SpringerLink](https://link.springer.com/chapter/10.1007/978-981-15-3369-3_35)

[6] [An Overview of Automatic Text Summarization Techniques – IJERT](https://www.ijert.org/an-overview-of-automatic-text-summarization-techniques)

[7] [Automatic text summarization: A comprehensive survey - ScienceDirect](https://www.sciencedirect.com/science/article/abs/pii/S0957417420305030)

[8] [(PDF) Review of Automatic Text Summarization Techniques & Methods (researchgate.net)](https://www.researchgate.net/publication/341517588_Review_of_Automatic_Text_Summarization_Techniques_Methods)

[9] [A Quick Introduction to Text Summarization in Machine Learning | by Dr. Michael J. Garbade | Towards Data Science](https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f)

[10] [Automatic text summarization based on sentences clustering and extraction (researchgate.net)](https://www.researchgate.net/publication/224588010_Automatic_text_summarization_based_on_sentences_clustering_and_extraction#:~:text=Automatic%20text%20summarization%20based%20on%20sentence%20clustering%20and,Modelling%20Intelligence%20System%20Based%20On%20Automatic%20Text%20Summarization)

[11] [Extractive Text Summarization Using Neural Networks | by Sahil Chaudhary | Heartbeat (fritz.ai)](https://heartbeat.fritz.ai/extractive-text-summarization-using-neural-networks-5845804c7701)

[12] [Flask Tutorials – Real Python](https://realpython.com/tutorials/flask/)

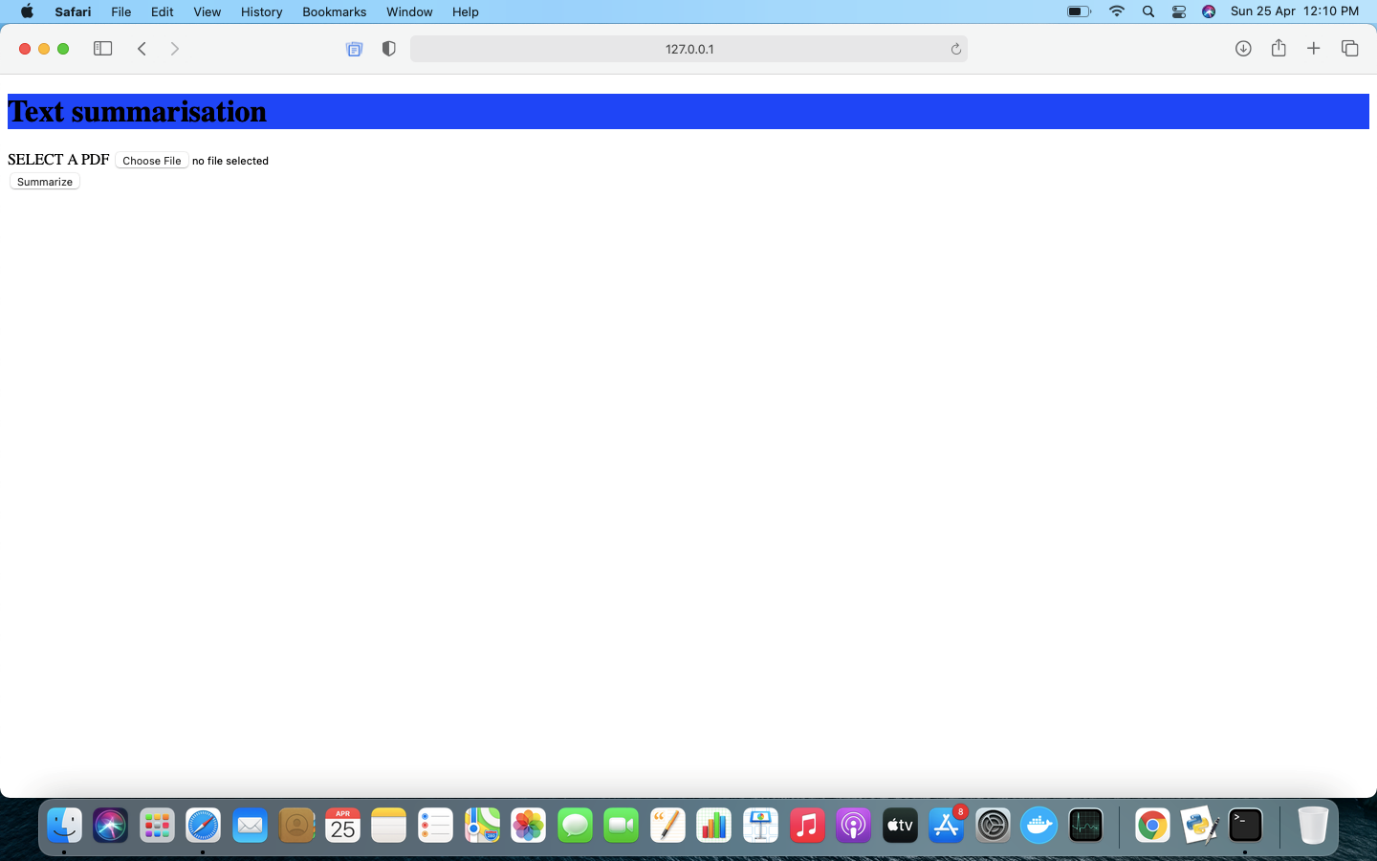
[13] [Understanding BERT Transformer: Attention isn’t all you need | by Damien Sileo | synapse\_dev | Medium](https://medium.com/synapse-dev/understanding-bert-transformer-attention-isnt-all-you-need-5839ebd396db)

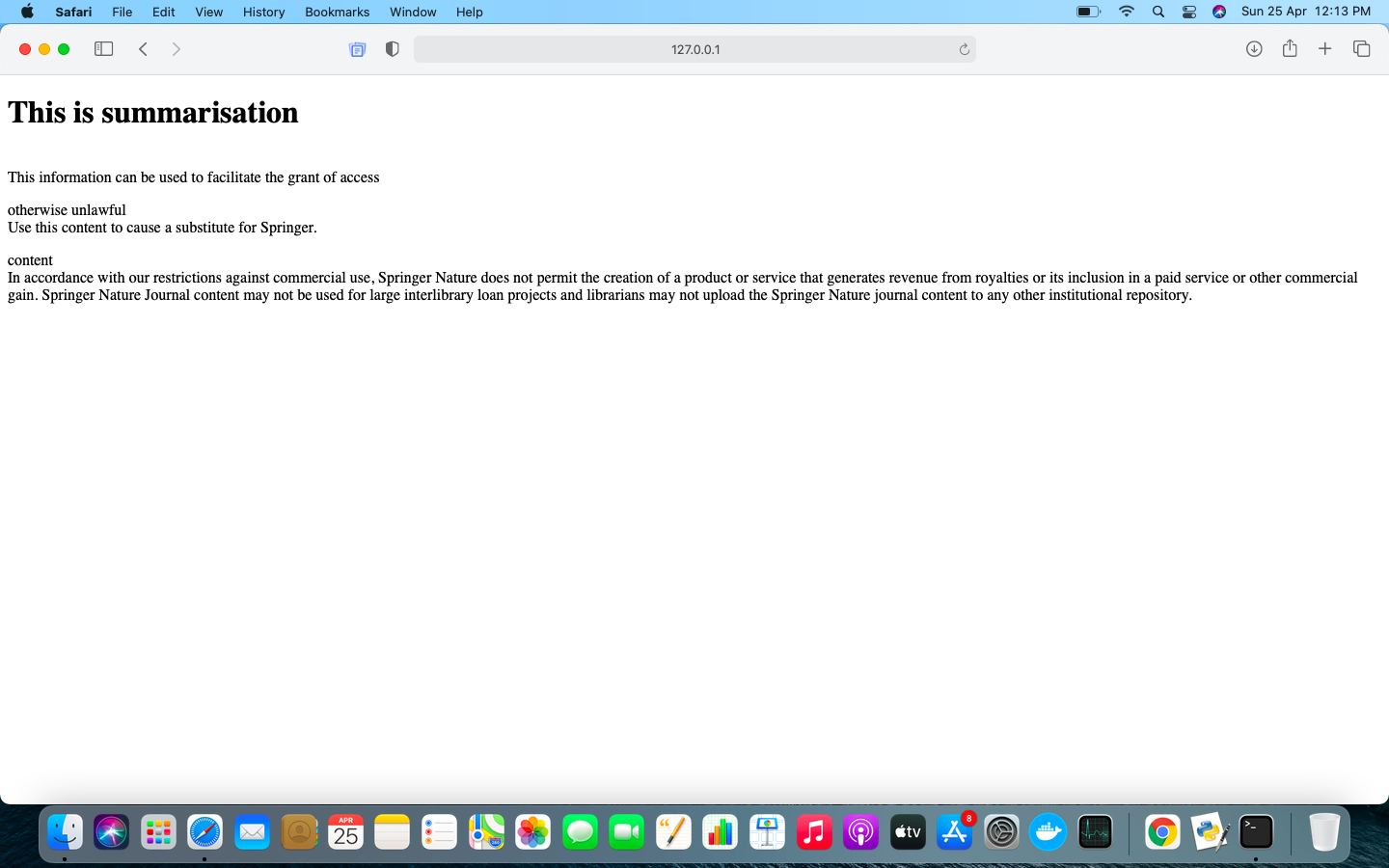
[14] [Automatic Text Summarization Using Fuzzy Extraction | SpringerLink](https://link.springer.com/chapter/10.1007%2F978-981-15-1286-5_33)  
[15] [Automatic text summarization with neural networks | IEEE Conference Publication | IEEE Xplore](https://ieeexplore.ieee.org/document/1344634)

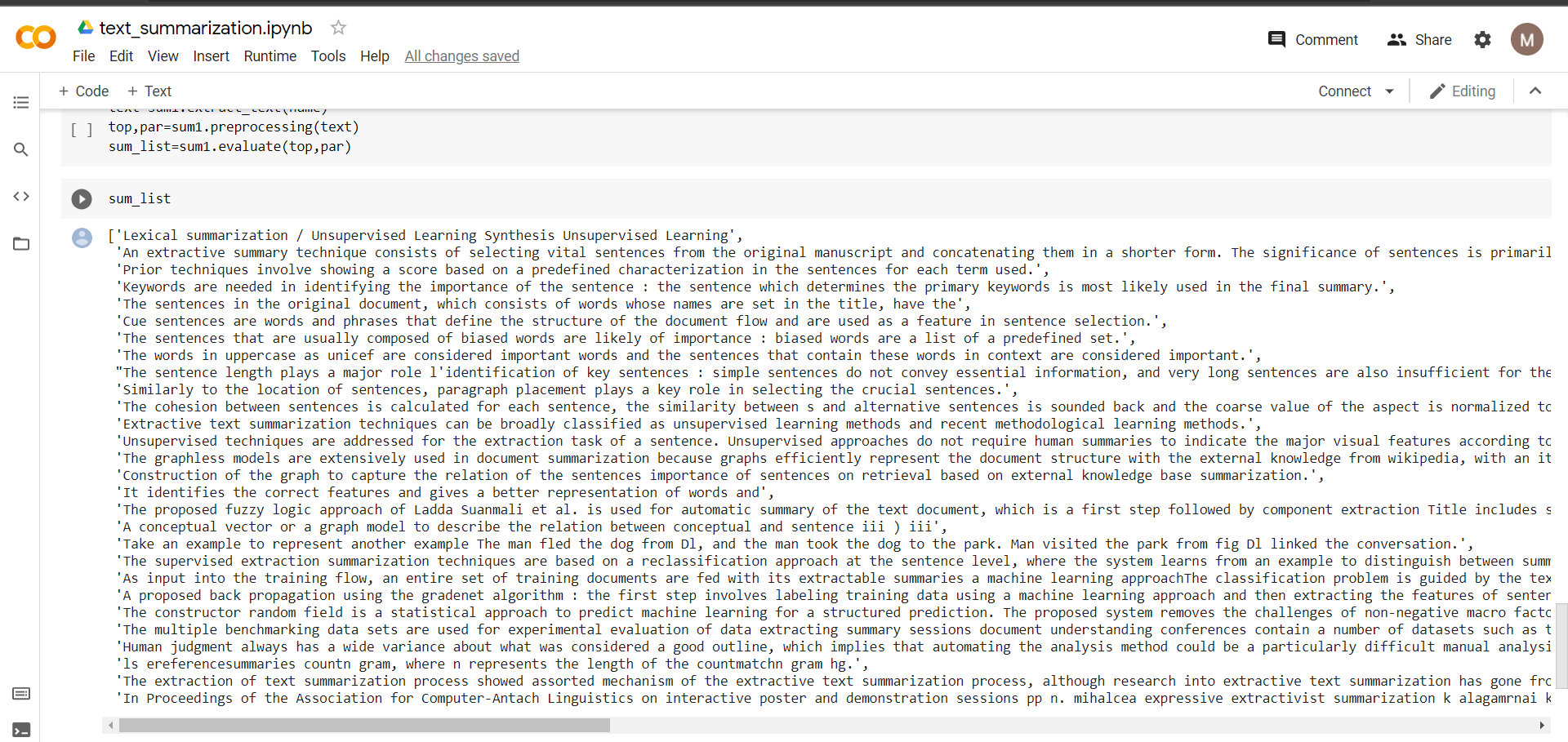
1. **APPENDIX**

**10.1 APPENDIX A**

**SNAPSHOTS OF OUTPUT**

****

****

****

**10.2 APPENDIX B**

**SOURCE CODE:**

**HTML HOME PAGE**

<!DOCTYPE html>

<html>

<head>

<title>

mini project

</title>

</head>

<body>

<h1 style="background-color:blue">

Text summarisation

</h1>

<form id="pd" action="/main" method="post" enctype="multipart/form-data">

<label for="pdf">

SELECT A COVID-19 PDF

</label>

<input type="file" id="pdf" name="pdf" accept=".pdf"><br>

<input type="submit" value="Summarize" formaction="/main" formmethod="post">

</form>

</body>

</html>

**OUTPUT PAGE:**

<!DOCTYPE html>

<html>

<head>

<title>

summarisation

</title>

</head>

<body>

<h1>

This is summarisation

</h1>

<p>

{% for key,value in end\_dict.items() %}

<p>{{key}} <br> {{value}}</p>

{% endfor %}

</p>

</body>

</html>

**DATA PREPROCESSING AND SUMMARIZATION:**

import numpy as np

import re

import pandas as pd

import tensorflow as tf

from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

from pdfminer.high\_level import extract\_text

from collections import defaultdict

class summary:

def \_\_init\_\_(self):#declare model

self.tokenizer = AutoTokenizer.from\_pretrained("Vamsi/T5\_Paraphrase\_Paws")

self.model = AutoModelForSeq2SeqLM.from\_pretrained("Vamsi/T5\_Paraphrase\_Paws")

device = "cpu" #GPU access

self.model = self.model.to(device)

self.fin\_sum=[]

def extract\_text(self,filename): #extract text from pdf file assuming the pdf is already in device

self.text=extract\_text(filename)

return self.text

def preprocessing(self,text):#process extracted text into topic and their corresponding paragraph

para=self.filtering(self.text)

d = self.seperate(para)

key=list(d.keys())

val=list(d.values())

par,top=[],[]#top= topic, par=paragraph

loc=0

for i in range(len(key)):

if val[i]!="":

top.append('')

par.append('')

te=key[i].replace('-\n','')#Applies for both value and key eg: if a same word is to stored in 2 lines, 1st line 'test-' ,2nd line 'ing'=>in text it'll be -\n format, so we're joining the word

top[loc]+=te.replace('\n',' ')#like above case it is to join 2 words in 2 lines into 2 words seperated by space in 1 line

te=val[i].replace('-\n','')

par[loc]+=te.replace('\n',' ')

loc+=1#index value of top and par

return [top,par]

def filtering(self,text):#eliminate authors,title,references from text

para="\n\n" #filtered text

flag=1

text1=self.text.lower()

for i in range(len(text1)-8):

if text1[i:i+8]=='abstract':#starts extracting text once abstract is reached

flag=0

if text1[i:i+14]=='\n\nreferences\n\n':#contiunes till it reaches references

break

if flag==0:

para+=text1[i]

para+='\n\n'

return para

def seperate(self,para):#eliminate punctuations and split into topic and paragraph

d=defaultdict(list)

i=0

k,val="",""

while(i<len(para)):

a=list(re.search('\n(?=\n)',para[i:]).span())

a[0],a[1]=a[0]+i,a[1]+i

if((a[1]+1)>=len(para)-3):

break

b=list(re.search('\n(?=\n)',para[a[1]+1:]).span())

b[0],b[1]=b[0]+a[1]+1,b[1]+a[1]+1

t=para[a[1]+1:b[0]] # store string between \n\n and \n\n ie:string between 2 enter

i=b[0]-1

lowtext=t

tex=""

val=""

pattern = r'\[.\*?\]'#remove referenced papers [5,6,7]

reg\_text=re.sub(pattern, '', lowtext)

pattern = r'\(.\*?\)'

reg\_text=re.sub(pattern, '', reg\_text)

punc= '''!()-[]{};:'"\,<>/?@#$%^&\*\_~0123456789.''' #removes punctuation

for j in reg\_text:

if j not in punc:

tex=tex+j

if tex.find('ieee')==-1: #to eliminate IEEE mark

if len(tex)<=50 and len(tex)>4:#if it is a topic

k=tex

elif len(tex)>=100:

val=tex

d[k] = val

return d #returns a dictionary in{topic:paragraph} format

def evaluate(self,top,par):#summarizing model

for ii in range(len(top)):#iterate for each topic specified in top

text1=par[ii]

oo=text1

for i in range(2,7):

n=int(len(oo)/10)

tex= "paraphrase: " + text1 + " </s>"

encoding = self.tokenizer.encode\_plus(tex,pad\_to\_max\_length=False, return\_tensors="pt")

input\_ids, attention\_masks = encoding["input\_ids"].to("cpu"), encoding["attention\_mask"].to("cpu")

outputs = self.model.generate(

input\_ids=input\_ids, attention\_mask=attention\_masks,

max\_length=n,

do\_sample=True,

top\_k=120,

top\_p=1,

early\_stopping=True,

num\_return\_sequences=5

)

line=[]

for output in outputs:

line.append(self.tokenizer.decode(output, skip\_special\_tokens=True,clean\_up\_tokenization\_spaces=True))

text1=str(line[0])

self.fin\_sum.append(str(line[0]))

return self.fin\_sum#list of summarized paragraphs

def summ(name):

end\_dict={}

sum1=summary()

text=sum1.extract\_text(name)

top,par=sum1.preprocessing(text)

sum\_list=sum1.evaluate(top,par)

for i in range(len(sum\_list)):

end\_dict[top[i]]=sum\_list[i]

return end\_dict

**CONNECTING MODEL TO WEBPAGE:**

from flask import \*

import summ

app=Flask(\_\_name\_\_)

@app.route('/')

def first\_page():

return render\_template("webpage.html")

@app.route('/main',methods=['POST'])

def main():

if request.method=='POST':

file=request.files['pdf']

file.save(file.filename)

end\_dict=summ.summ(file.filename)

return render\_template("main.html",end\_dict=end\_dict)

else:

return render\_template("webpage.html")

if \_\_name\_\_=='\_\_main\_\_':

app.run(debug=True)