Project Synopsis On

COMPUTERIZED SYNOPSIS GENERATOR

By

Sweta Gupta - 28

Yash Jobalia - 30

Isheet Shetty - 55

Guided by

Prof. Anagha Patil

(Asst.Prof., Department of Information Technology)

Department of Information Technology

Vidyavardhini's College of Engineering and Technology
University of Mumbai

2021-22

Vidyavardhini's College of Engineering and Technology

Department of Information Technology

CERTIFICATE

Sweta Gupta - 28
Yash Jobalia - 30
Isheet Shetty - 55

COMPUTERIZED SYNOPSIS BUILDER

Internal Guide:	____________________	( )
Internal Examiner:	____________________	( )
External Examiner:	____________________	( )

Dr. Ashish Vanmali	**Dr. Harish Vankudre**
(HOD, IT)	**(Principal, VCET)**

Acknowledgement

It gives us immense pleasure to thank Dr. Harish Vankudre, our Principal for extending his support to carry out this project. We wish to express our gratitude and thanks to our HOD, Dr. Ashish Vanmali and our guide, Prof. Anagha Patil for helping us and giving us support because of which we are able to finish our Project on time. We would also like to thank some other faculty members of the Information Technology Department, Prof. Maryam Jawadwala and Prof. Vaishali Shirsath for their valuable ideas and timely assistance in this project.

Sweta Gupta - 28
Yash Jobalia - 30
Isheet Shetty - 55

Abstract

Nowadays, the world is plummeting by the escalation of the amount of data. With such a bombardment of data wandering aimlessly in a high-tech real-time virtual universe, it appears necessary to synopsize overlong text and serve on a target compendium that can cogently deliver the intended messages. Thereupon, synopsis generation is the charge of generating incisive summaries unescorted by human assistance while conserving the genuine sense of the overlong document. Summarization creation is critical in the goal of saving time.

Contents

1 Introduction **1**
1.1 Problem Statement - to generate a concise summary 1
1.2 Motivation 2

2 Report on the present investigation **3**
2.1 Overview 4

3 Working **7**
3.1 Speech To Text Phase 7
3.2 Text Summarization 8
3.3 Text To Speech 12

4 Results and Discussions **14**

5 Summary and Conclusions **17**
5.1 Future Work 18

6 Appendix **20**
6.1 Python 20
6.2 Natural Language Processing 21
6.3 FFmpeg 21

List of Figures

2.1 Flowchart of the project 5

3.1 Comparison of Techniques 13

4.1 Speech To Text 14
4.2 Abstractive Text Summarization 15
4.3 Extractive Text Summarization 15
4.4 Extractive Text Summarization 16

Chapter 1

Introduction

The human attention span is less than 8 seconds so, if you need to capture someone's attention or highlight an important topic, you need to have a strong headline or summary. A prime example of this phenomenon is that whenever you open a newspaper you glance through the headlines and read through only the ones which have captivating headlines or an interest which aligns with yours. Hence, in this ever growing and vastly expanding world with abundance of data, you need to make sure that the important topics gain your priority attention and you get precise information and knowledge from the vast abyss of data. Information is knowledge and knowledge is power. With the huge increase in the amount of data available the fine distinction between the knowledge information and data is thinning. The motive is to use the huge amount of data provided, abstract the required information and try to highlight emphasize on the necessary knowledge that it contains.

1.1 Problem Statement - to generate a concise summary

In the modern day , time is not only considered as valuable but is also sometimes regarded as the most important aspect because once spent it cannot be taken back . Moreover no one has the luxury of spending their precious time on reading reports which are more than a few pages long just of the regular day to day meeting or a general conversation. Hence in order to save your valuable time the meeting summariser can summarise the entire meeting into a few paragraphs and also highlight the entire gist of the content in a few lines. The project's purpose is to learn about natural language processing concepts and construct a text summary machine learning tool that only incorporates the most important information from the material. The process takes place as follows . We first record the conversation or the meeting from a voice recorder on a device like phone or a mic then we use highly sophisticated STT (speech To Text) algorithms which recognise speech and convert them to text as our first input . These algorithms have the capacity to convert huge audio files with large number of recorded minutes into smaller chunks of audio files slashed at regular intervals of time in order to reduce discrepancy and for better time complexity and management with better results. This converted text file will be then encoded and decoded by algorithms by either extractive summarisation or abstractive summarisation based on the users requirements in order to convert to convert the lengthy conversation or meetings into small summaries of the entire topic containing of the most important topics and covering the gist of the text.

1.2 Motivation

The amount of data these days is exponentially increasing through the internet and various other sources. To avoid browsing through these overutilized and long-drawn-out documents and converting them to succinct summary, we stand in need of a tool that can help withdraw summary by clipping of the data in these documents and giving the foremost sentences with pivotal meaning from the prolix document or from a cluster of documents. It is difficult for the human mind to reminisce all this data, so a synopsis generator plays a pivotal role to save the human effort and time. Our work aims to build a synopsis generator that provides the user the liberty to make a selection from n number of summarization methods in accordance with their needs. Furthermore humans are highly intellectual species who can communicate through sophisticated measures and in order to make a futuristic tool which can utilise these highly efficient communicative methods the tool must have additional features like converting human dictated or vocalised speech to a written readable and legible texts and hence giving them a boon of summarising verbose and elongated speeches and conversations to minimalistic, concise and to the point summaries containing the heart of the entire passage all the while keeping them limited upto a certain precise point

Chapter 2

Report on the present investigation

This segment elucidates about the techniques that have been used for text summarization.It is one of the branches in natural processing language. Text summarization is bifurcated into categories as follows :

- Extractive Summarization
- Abstractive Summarization.

Extractive text summarization is to handpick must-have and imperative sentence from the text. This necessitous and important paragraph can be picked out by using linguistic and statistical features of paragraphs. Abstractive summarization learns the main concept of the long-drawn-out document and meaning of the same document or text. It discovers the new concept from the document by using linguistic method by interpreting the text. In preliminary researches, text summarization which is a part of natural processing language was carried out on scientific documents focused on the proposed features like sentence ranking.
J.N.Madhuri and Ganesh Kumar.R have done analysis on extractive summarization by extracting upmost weighted frequency sentences. In this paper we see that after finishing the pre-processing step they calculate the frequency of each keyword like how habitually that keyword has arised, from that greatest frequency of the keyword is taken. Then weighted frequency of the word is calculated by dividing frequency of the keywords by maximum frequency of the keywords. In this step, they calculate the sum of weighted frequencies. Finally, the summarizer extracted the high weighted frequency sentences. Similarly, Aakanksha Sharaff, Amit Siddharth Khaire and Dimple Sharma analysed rule based logic for extractive text summarization in which

first the dataset is pre-processed which is used to calculate frequency and position. This is done by using hash map and formulae after which the weight of sentences is calculated which is done by using frequency and position of words formula. The last step is analysis where the calculated values are used to find the mean for that sentence and are fed to the triangular membership function which gives values between 0 and 1 to each sentence and the fuzzy rule is applied. These values are arranged in descending order and they are picked according to the percentage of the original text needed. Finally, the ROUGE score is given to each and every summary to be compared efficiently. In the next paper, Siya Sadashiv Naik, Manisha Naik Gaonkar [3] research about extractive text summarization by feature based sentence extraction using rule based. The main focus of this paper is to summarize a single document and create its extractive summary. After pre-processing step, each sentence of document is represented as attribute vector of features. Seven features are calculated for each sentence and each feature is given a value from 0 to 1 after normalization. Features considered are Sentence Position, Title Feature, Numerical Value, Keyword Weight, Proper Noun, Sentence-To-Sentence Similarity and Sentence Length. Based on their ratings, all of the sentences are arranged in ascending order. Finally, the extractive summary of the document will be generated and displayed. In the following paper, Kaiz Merchant and Yash Pande [4] use latent semantic analysis approach for creating short summaries on basis of similar words. They use 2 approaches depending on the type of case if it is a criminal case, they used single document untrained approach and for civil case they used multi-document trained approach. They first pre processed the data which is cleaning it, lemmatisation and removing stop words then they pass it through the model and depending on the type of case (civil case or a criminal case) it is decided which process is used. Then using the appropriate model, they generate a summary and finally add sentence selection, in which the final line is always added because it is the judgement passed and hence the final summary output is generated.

2.1 Overview

In this section, we represent our methodology for making an effective text summarizer for english to english text document. The project uses a top-down approach in which each segment is divided into smaller components and each component performs a particular role. The objective of the project is to understand the concept of natural processing language and creating a tool for text summarization containing only main points described in the

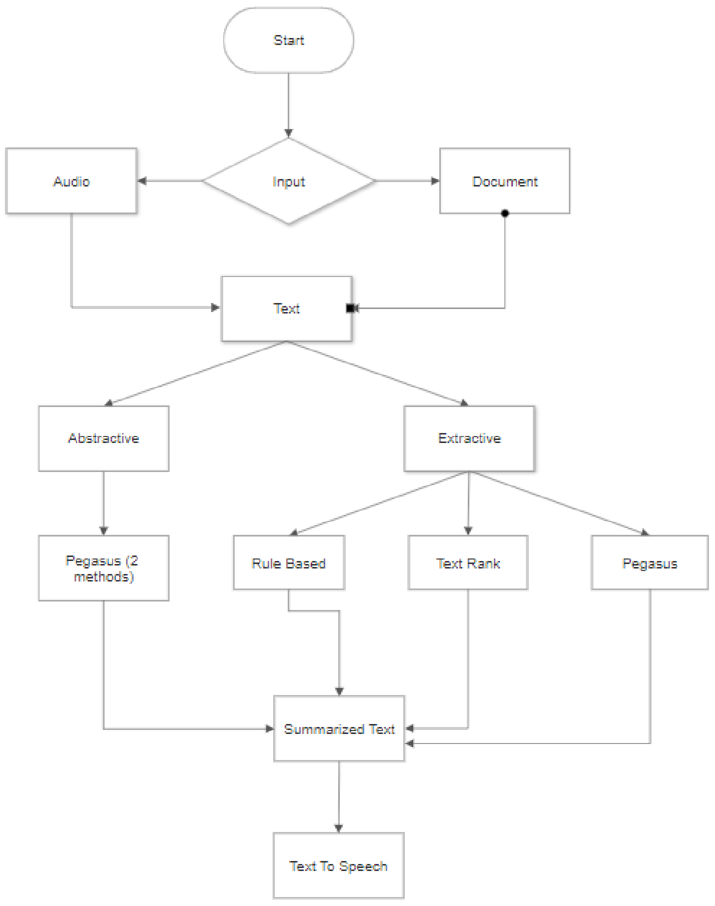


Figure 2.1: Flowchart of the project

document.
For example, the entire project is divided into three phases:

- Speech to text phase
- Summarization phase
- Text to speech phase

The first phase aims at converting the audio files into texts. The output generated from the first phase is further used as an input for the succeeding phase which consists of summarizing these verbose texts into short excerpts along with exterminating discrepancies which may have been introduced in the previous phase due to disturbances in the audio file. Finally, the third phase is an optional feature provided to the users especially with special needs which helps them convert the short summary into an audio output. The users can use the model in various ways depending on their specific requirements and the numerous options provided in each and every phase. Furthermore, each phase also has various options provided to the users which have been discussed in detail below.

2.1.1 Speech To Text Phase

The first phase is the speech to text phase. In this phase, the given audio input file is converted into text. This process is done through passing the audio file through an algorithm which uses google speech recognition in order to identify the audio speech and convert it into text.

2.1.2 Text Summarization Phase

The second phase is the core of the project. It deals with the summarization of the text which were the output from the previous phase. This segment gives the user the ability to select several choices depending on his/her needs of summarization. The first set of options from which the user can select the type of summarization is either abstractive or extractive. In abstractive method, the user gets futher options to either summarize the gist of the text in limited lines or generate an abstractive headline. These methods try to understand the context of the input and generate the desirable output in complete new words either using pegasus xsum or pegasus reddit tifu model. On the counter part, the extractive method gives the user another set of choices consisting of different algorithms based on extractive summarization in which key pivotal lines are extracted from the input text and used as it is for the output. There are several extractive methods listed below which are explained in great detail in the working chapter. The extractive methods used are as follows: Rule based, gensim, text rank and pegasus.

2.1.3 Text To Speech Phase

The final phase consists of the optional functionality which the user can use depending on his/her needs. This phase has the main purpose of converting the summarised text outputs into audio form. This is done by using various libraries like gTTs and pyttsx3 of python language. These additional features help the user in getting the output in any format he/she desires. This can be useful for users with special needs.

Chapter 3

Working

The working of the project takes place in three different phases each with its own unique objective, input and output. Each phase performs a unique task which is a key necessity for the overall output. Furthermore, each phase has its own set of unique options which it provides to its users for a wide range of combinations such as type of input, summarization type, summarization methods, ratio of summarization and output method.

3.1 Speech To Text Phase

Speech recognition, a multidisciplinary topic of computational linguistics, assists in the development of technology that enable speech recognition and translation into text. Long speeches can be difficult to follow at times owing to differences in pronunciation, tempo, and other factors. The speech is transformed to the appropriate text, which is then summarised. Text summarization is the process of extracting the most significant information from a text source and providing an appropriate summary of it. This has a variety of uses, including creating lecture notes, summarising catalogues for long texts, and so on. The speech to text phase deals with the first problem of extracting the speech from the audio file and converting it into text. The speech to text phase uses various python libraries such as speech recognition, os and pydub. Each library serving a unique purpose in converting a overall speech to text. The speech recogntion text consists of recognizer function. This function is crucial to the entire working and conversion of the audio file. The bonus of using these libraries is that we can convert various files with large audio files in an efficient way. The method in which we achieve such efficiency is by converting the large audio file into smaller chunks slashed at regular intervals. The next step is, we use the os library to create a folder which consists of various audio chunks which are a product of slashing the large audio file. These chunks are made based on the silence feature which is a part of split on silence library which is a part of pydub silence library. The speech recognition is applied to each and every chunk stored in the directory. Once each chunk is converted, its text is appended to give us a final text output containing the entire audio file converted into text.

3.2 Text Summarization

3.2.1 Extractive Summarization

The extractive strategy includes extracting the most pertinent words and lines from the documents. After then, the summary is made by merging all of the key lines. As a consequence, each line and word in the summary belongs to the original text that is being summarised in this case.

Gensim

Gensim is one of the most used free open source python libraries for converting documents efficiently. It is a useful module for conducting natural language processing tasks. It is a mild variation of the TextRank algorithm, The TextRank Algorithm is used to summarise text using the gensim package. TextRank is a technique for extracting information from documents. It is based on the idea that words that appear more frequently are more important. As a result, sentences with a high frequency of words are crucial. The system then assigns ratings to each sentence in the text based on this. The top-scoring sentences are included in the summary. The first step after importing the gensim package is to import summarise from gensim summarization. TextRank is implemented using an in-built function. Then, as an input, provide the text corpus to the summary function.The default parameters of the summary function can be changed to suit your needs. The parameters are as follows:

- It has a range of values from 0 to 1.
- It reflects the percentage of the summary that is different from the original text.
- The number of words in the summary is determined by word count.

Text Rank

TextRank is a text summation approach used to construct Document Summaries in Natural Language Processing. TextRank is an unsupervised graph-based text summarising tool that uses an extractive method. PageRank is a search engine ranking algorithm that Google and other search engines use to determine the importance of webpages. TextRank is built on top of the PageRank Algorithm. This algorithm was inspired by Larry Page, one of Google's co-founders. A page's rank and worth are determined by the number and quality of links it has. It does numerous iterations on the pages to arrive at a final value. The basic steps involved in TextRank algorithm are as follows -

- Step 1 : Use any approach you like to extract all of the sentences from the text document, such as separating at whitespaces or full stops.
- Step 2 : The phrases retrieved in Step 1 are used to generate a graph. The weight on the edges between two nodes is computed using a similarity function such as Cosine or Jaccard Similarity, and the nodes represent texts. For this purpose, you can create your own similarity metric.
- Step 3 : This stage entails iterating the algorithm until convergence, i.e. until consistent scores are acquired, in order to determine the significance (scores) of each node.
- Step 4 : The sentences are sorted in descending order based on their scores. To be included in the summary, the first k sentences of the text are picked.

Rule Based

The Rule-Based Summarizer's suggested design is detailed below.

- Pre-processing: The most important stage in any summarising approach is pre-processing. Pre-processing is used to sanitise the document, eliminate noisy data, and correct grammatical problems. Tokenization, stop word elimination, and stemming are some of the pre-processing techniques used.
- Tokenization: Tokenization is the process of breaking down paragraphs into sentences and then breaking down each phrase into individual words or tokens.

- Stop Word Removal: After tokenization, the data is evaluated, and frequent terms or stop words such as a, an, and the are deleted from the document. All words are reduced to their root format in stemming.
- Keyword Extraction: The frequency count of each word or phrase in a text is determined in the keyword extraction phase to determine its relevance. This is accomplished by computing the document's (tf-idf) scores.
- Threshold: A threshold is defined during this stage. This number is derived by adding the terms with the lowest and highest frequencies and taking the mean of the two. All phrases with tf less than the threshold value are trimmed from the document after the threshold is computed.
- Feature Extraction: Each phrase of the document is represented as an attribute vector of features after the pre-processing stage. Each phrase has seven characteristics, each of which is assigned a value between 0 and 1 after normalisation.
- Rule Generation: For each of the seven characteristics, low and high values are determined. Following that, a single rule is constructed with all feature values set to high, with the exception of sentence to sentence similarity, which is set to low because the summary requires fewer comparable sentences.All sentences are run via this rule once it has been written. This rule's feature values are mapped to the sentence's feature values. If there is a match, the result is 0; otherwise, the output is 1. Finally, with reference to the rule, all 1's are totaled, giving us the number of mismatching characteristics for that statement. A single score value has been assigned to each sentence.
- Sentence Completion: Based on their ratings, all sentences are arranged in ascending order. To retrieve sentences, the extent of summarization is determined. Sentences are extracted to a 20 percent extent; it has been proven that extracting sentences to this percentage yields a more informative summary than extracting sentences from the entire manuscript.
- Extractive Summary: The document's final extractive summary will be shown.

3.2.2 Abstractive Summarization

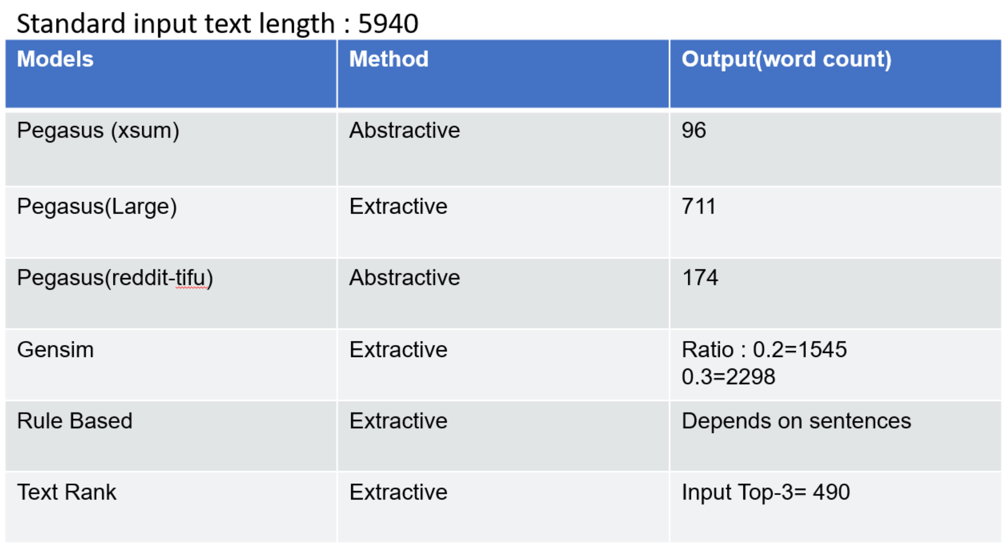
The technique of constructing a quick and succinct summary of a source text that captures the major ideas is known as abstractive text summarization. Additional phrases and sentences not present in the original text may be included in the summaries.

Pegasus

Abstractive text summarization is one of the most difficult jobs in natural language processing since it requires reading long passages, compression of information, and language development. In 2020, Google AI released PEGASUS, which stands for Pre-training with Extracted Gap-Sentences for Abstractive Summarization. They suggest that massive Transformer-based encoder-decoder models be pre-trained using a single self-supervised objective on big text corpora. PEGASUS, an encoder-decoder-based language model, is used by the abstractive summarizer to create a semantically sound summary. PEGASUS is a gap sentence masking and summary generating pre-training approach. The PEGASUS model's architecture typically includes 15 layers of encoders and 15 layers of decoders, all of which consider text documents as input after masking. The closer the pre-training self-supervised target is to the final down-stream task, the better the fine-tuning performance, according to their premise. Several sentences are deleted from documents and the model is charged with recovering them in the proposed technique, pre-training. A document with missing sentences can be used as an example input for pre-training, with the output consisting of the missing phrases concatenated together. PEGASUS masks many complete sentences from a manuscript. These sentences have been taught to PEGASUS to anticipate. PEGASUS will recover missing sentences from an input document, and the outcome will be a concatenated list of missing sentences. The task at hand is Gap Sentence Generation. Although it features an encoder and a decoder in its fundamental design, PEGASUS main addition is Gap Sentence Generation. PEGASUS uses a pre-trained encoder as a masked language model. In PEGASUS, complete sentences are removed from a text and the model is trained to predict these sentences, as seen in the image. The authors concede that this endeavour appears to be challenging even for humans. However, such training enhances comprehension for the development of words with a relationship to the original content, so confirming their theory. Gap Sentence Generation is the name given to this endeavour (GSG). Furthermore, the authors recommend that hiding the document's most significant phrases is the most effective technique. This is accomplished by utilising a statistic called ROUGE to identify phrases that are the most similar to the entire text (which is usually used to evaluate the quality of a summary in summarization tasks). Masked Language Model (MLM) Although the GSG (explained in the preceding section) is PEGASUS's most important contribution, its basic architecture consists of an encoder and a decoder; hence, pre-training the encoder as a masked language model is a good idea. We mask words from sequences at random and guess the masked words using other words from the sequence in this challenge. The GSG job, which is based on this paradigm, might be considered a document-level MLM. The model has been fine-tuned using 12 accessible abstractive summarization datasets. It beat the prior state of the art on six of these datasets and it did so with a little number of samples.

3.3 Text To Speech

This is the final phase of the project. We sometimes prefer to listen to information rather than read it. While listening to the crucial file data, we can multitask. Python has a number of APIs for converting text to voice. The Google Text to Speech API, or gTTS API, is a well-known and widely used API. The programme is simple to use and comes with a number of built-in features, including the ability to save a text file as an mp3 file. We don't need to employ a neural network or train a model to convert the file to voice because that is likewise difficult. Instead, we'll finish a task using these APIs. We will use another offline package called pyttsx3 to convert text files into. The gTTS function is a three-argument function. The first parameter is a text value to be converted to a speech. The second parameter is a language to use. It supports a wide range of languages. We can make an audio file out of the text. The pace of the discourse is represented by the third argument. We set the slow value to false, which implies that the speech will be delivered at usual pace. As a result, we obtain speech as a result.



Standard input text length : 5940

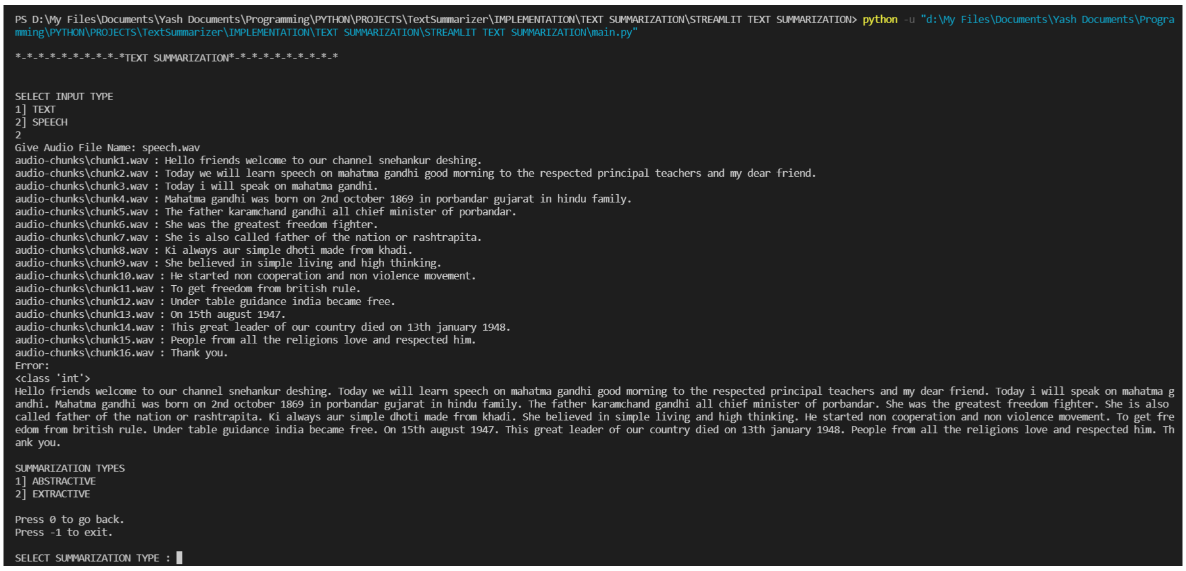
Models	Method	Output(word count)
Pegasus (xsum)	Abstractive	96
Pegasus(Large)	Extractive	711
Pegasus(reddit-tifu)	Abstractive	174
Gensim	Extractive	Ratio : 0.2=1545 0.3=2298
Rule Based	Extractive	Depends on sentences
Text Rank	Extractive	Input Top-3= 490

Figure 3.1: Comparison of Techniques

Chapter 4

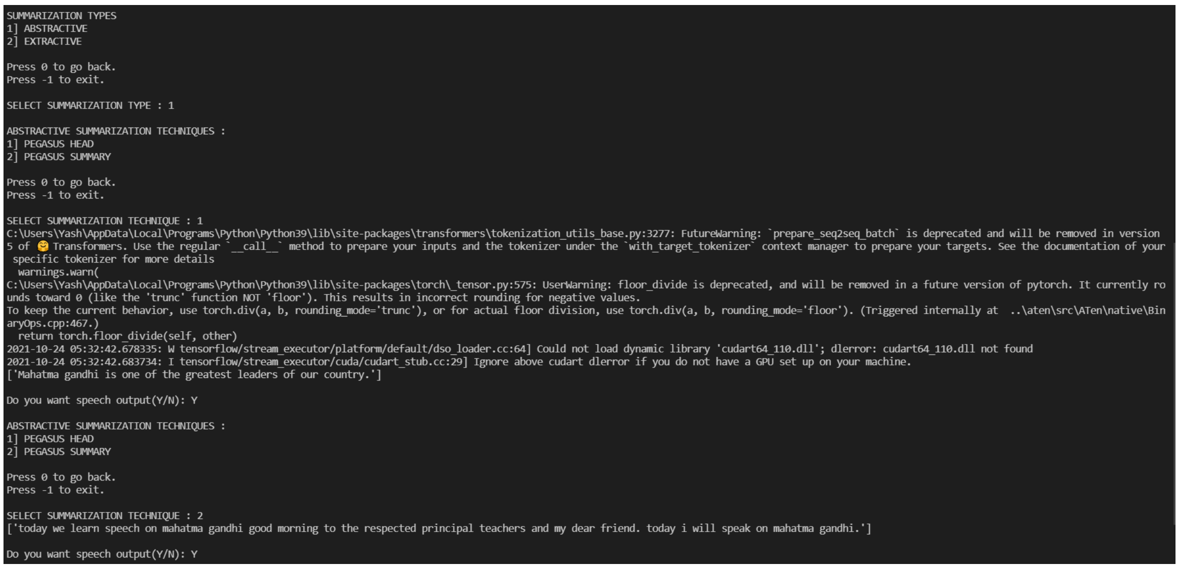
Results and Discussions

As a result of implementing various extractive and abstractive methods of text summarization, each having its own perks and swindles we can conclude in this chapter the efficiencies and the conclusions of enacting various techniques. Following are the implementation results, each compared to the output to the same input.



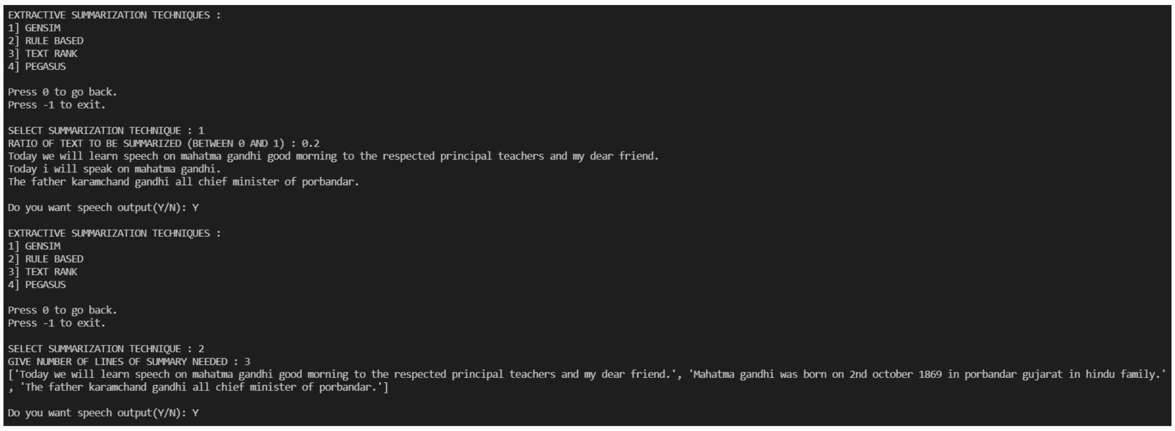
```
PS D:\My Files\Documents\Yash Documents\Programming\PYTHON\PROJECTS\TextSummarizer\IMPLEMENTATION\TEXT SUMMARIZATION\STREAMLIT TEXT SUMMARIZATION> python -u "d:\My Files\Documents\Yash Documents\Progra
mming\PYTHON\PROJECTS\TextSummarizer\IMPLEMENTATION\TEXT SUMMARIZATION\STREAMLIT TEXT SUMMARIZATION\main.py"

*-*-*-*-*-*-*-*-*-*TEXT SUMMARIZATION*-*-*-*-*-*-*-*-*-*


SELECT INPUT TYPE
1] TEXT
2] SPEECH
2
Give Audio File Name: speech.wav
audio-chunks\chunk1.wav : Hello friends welcome to our channel snehankur deshing.
audio-chunks\chunk2.wav : Today we will learn speech on mahatma gandhi good morning to the respected principal teachers and my dear friend.
audio-chunks\chunk3.wav : Today i will speak on mahatma gandhi.
audio-chunks\chunk4.wav : Mahatma gandhi was born on 2nd october 1869 in porbandar gujarat in hindu family.
audio-chunks\chunk5.wav : The father karamchand gandhi all chief minister of porbandar.
audio-chunks\chunk6.wav : She was the greatest freedom fighter.
audio-chunks\chunk7.wav : She is also called father of the nation or rashtrapita.
audio-chunks\chunk8.wav : Ki always aur simple dhoti made from khadi.
audio-chunks\chunk9.wav : She believed in simple living and high thinking.
audio-chunks\chunk10.wav : He started non cooperation and non violence movement.
audio-chunks\chunk11.wav : To get freedom from british rule.
audio-chunks\chunk12.wav : Under table guidance india became free.
audio-chunks\chunk13.wav : On 15th august 1947.
audio-chunks\chunk14.wav : This great leader of our country died on 13th january 1948.
audio-chunks\chunk15.wav : People from all the religions love and respected him.
audio-chunks\chunk16.wav : Thank you.
Error:
<class 'int'>
Hello friends welcome to our channel snehankur deshing. Today we will learn speech on mahatma gandhi good morning to the respected principal teachers and my dear friend. Today i will speak on mahatma g
andhi. Mahatma gandhi was born on 2nd october 1869 in porbandar gujarat in hindu family. The father karamchand gandhi all chief minister of porbandar. She was the greatest freedom fighter. She is also
called father of the nation or rashtrapita. Ki always aur simple dhoti made from khadi. She believed in simple living and high thinking. He started non cooperation and non violence movement. To get fre
edom from british rule. Under table guidance india became free. On 15th august 1947. This great leader of our country died on 13th january 1948. People from all the religions love and respected him. Th
ank you.

SUMMARIZATION TYPES
1] ABSTRACTIVE
2] EXTRACTIVE

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TYPE : █
```

Figure 4.1: Speech To Text



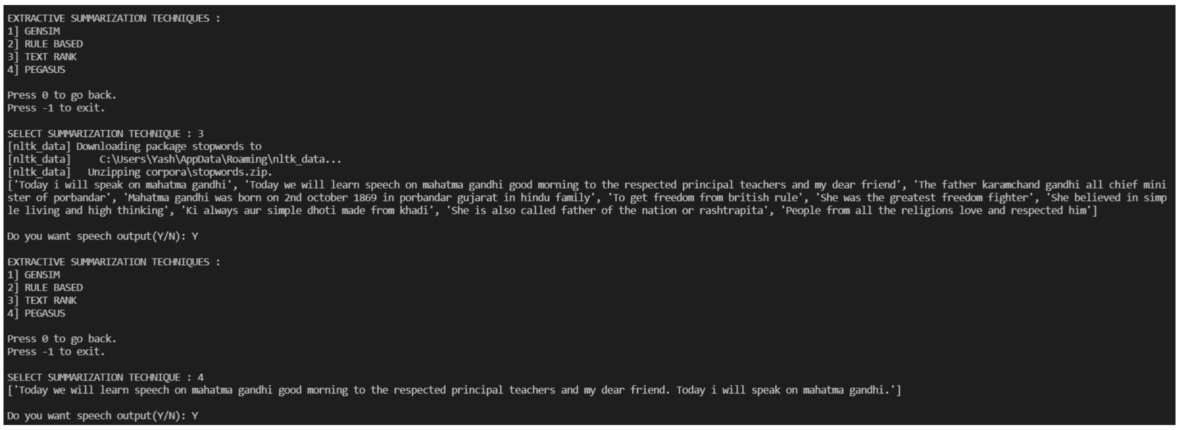
```
SUMMARIZATION TYPES
1] ABSTRACTIVE
2] EXTRACTIVE

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TYPE : 1

ABSTRACTIVE SUMMARIZATION TECHNIQUES :
1] PEGASUS HEAD
2] PEGASUS SUMMARY

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TECHNIQUE : 1
C:\Users\Yash\AppData\Local\Programs\Python\Python39\lib\site-packages\transformers\tokenization_utils_base.py:3277: FutureWarning: `prepare_seq2seq_batch` is deprecated and will be removed in version
5 of 🤗 Transformers. Use the regular `__call__` method to prepare your inputs and the tokenizer under the `with_target_tokenizer` context manager to prepare your targets. See the documentation of your
 specific tokenizer for more details
  warnings.warn(
C:\Users\Yash\AppData\Local\Programs\Python\Python39\lib\site-packages\torch\_tensor.py:575: UserWarning: floor_divide is deprecated, and will be removed in a future version of pytorch. It currently ro
unds toward 0 (like the 'trunc' function NOT 'floor'). This results in incorrect rounding for negative values.
To keep the current behavior, use torch.div(a, b, rounding_mode='trunc'), or for actual floor division, use torch.div(a, b, rounding_mode='floor'). (Triggered internally at  ..\aten\src\ATen\native\Bin
aryOps.cpp:467.)
  return torch.floor_divide(self, other)
2021-10-24 05:32:42.678335: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2021-10-24 05:32:42.683734: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
['Mahatma gandhi is one of the greatest leaders of our country.']

Do you want speech output(Y/N): Y

ABSTRACTIVE SUMMARIZATION TECHNIQUES :
1] PEGASUS HEAD
2] PEGASUS SUMMARY

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TECHNIQUE : 2
['today we learn speech on mahatma gandhi good morning to the respected principal teachers and my dear friend. today i will speak on mahatma gandhi.']

Do you want speech output(Y/N): Y
```

Figure 4.2: Abstractive Text Summarization



```
EXTRACTIVE SUMMARIZATION TECHNIQUES :
1] GENSIM
2] RULE BASED
3] TEXT RANK
4] PEGASUS

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TECHNIQUE : 1
RATIO OF TEXT TO BE SUMMARIZED (BETWEEN 0 AND 1) : 0.2
Today we will learn speech on mahatma gandhi good morning to the respected principal teachers and my dear friend.
Today i will speak on mahatma gandhi.
The father karamchand gandhi all chief minister of porbandar.

Do you want speech output(Y/N): Y

EXTRACTIVE SUMMARIZATION TECHNIQUES :
1] GENSIM
2] RULE BASED
3] TEXT RANK
4] PEGASUS

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TECHNIQUE : 2
GIVE NUMBER OF LINES OF SUMMARY NEEDED : 3
['Today we will learn speech on mahatma gandhi good morning to the respected principal teachers and my dear friend.', 'Mahatma gandhi was born on 2nd october 1869 in porbandar gujarat in hindu family.'
, 'The father karamchand gandhi all chief minister of porbandar.']

Do you want speech output(Y/N): Y
```

Figure 4.3: Extractive Text Summarization



```
EXTRACTIVE SUMMARIZATION TECHNIQUES :
1] GENSIM
2] RULE BASED
3] TEXT RANK
4] PEGASUS

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TECHNIQUE : 3
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Yash\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
['Today i will speak on mahatma gandhi', 'Today we will learn speech on mahatma gandhi good morning to the respected principal teachers and my dear friend', 'The father karamchand gandhi all chief mini
ster of porbandar', 'Mahatma gandhi was born on 2nd october 1869 in porbandar gujarat in hindu family', 'To get freedom from british rule', 'She was the greatest freedom fighter', 'She believed in simp
le living and high thinking', 'Ki always aur simple dhoti made from khadi', 'She is also called father of the nation or rashtrapita', 'People from all the religions love and respected him']

Do you want speech output(Y/N): Y

EXTRACTIVE SUMMARIZATION TECHNIQUES :
1] GENSIM
2] RULE BASED
3] TEXT RANK
4] PEGASUS

Press 0 to go back.
Press -1 to exit.

SELECT SUMMARIZATION TECHNIQUE : 4
['Today we will learn speech on mahatma gandhi good morning to the respected principal teachers and my dear friend. Today i will speak on mahatma gandhi.']

Do you want speech output(Y/N): Y
```

Figure 4.4: Extractive Text Summarization

Chapter 5

Summary and Conclusions

This is the final chapter of the thesis. The first portion of the Chapter will be a concise summary of the work that has been completed. The results of the logical analysis provided in the Results and Discussions Chapter must be presented and stated in full, with each argument articulated separately. The scope of future work should be explicitly indicated in the chapter's final section.
In recent days, several studies on the creation of summaries from numerous documents have been done. Text summarising generates a summary comprising essential phrases and all pertinent important information from the source material automatically. According to the summary data, extractive and abstractive techniques are among the most common. Text summary has a number of key advantages, including the following: They make reading easier. It saves you time. It makes it easier to remember information. It improves work rate efficiency.The basic objectives of text summarization are as follows:-Optimal topic coverage Maximum readability. There are a few evaluation standards to assure these two aspects. One of these is salience, or the retention of the most significant component. To capture the most significant information from the original document, a summarizer must be programmed. The final summary must be exactly the right length. It should be neither too long nor too short. The structure must be user-friendly. The sentences must be logical and understandable.It should not include any unusual pronouns. The summary as a whole must be balanced. This implies that it must include all of the main components of the paper and, of course, be grammatically perfect throughout. Finally, the phrases should not be repetitive. If a summarizer meets all of these characteristics, it will be able to generate reader-friendly summaries that can benefit us in a variety of ways. Text summary has become an important component of the daily lives of academics, students, and anyone who work with large amounts of text. With the

advancement of technology and AIs, it is likely that one day, automatic text summarising may be as excellent and clear as manual text summarization.

5.1 Future Work

For future works, the current prokect works only and only for English language but in future we can expand this in such a way that it can support not only various languages but also inter language summarization.
Apart from the single document summarization, we can enhance this project to summarize multiple documents at the same time.
The time complexity of converting speech to text can be modified to achieve a faster and more accurate level.
The input type for speech to text currently only accepts wav files. This drawback needs to be amended such that it can support various file types.
A wide array of input types such as pdf, docx, jpg, etc can also be used as a supplement to the current input types.

References

[1] J.N.Madhuri , Ganesh Kumar.R “Extractive Text Summarization Using Sentence Ranking”, Institute of Electrical and Electronics Engineers (IEEE), 2019,

[2] Aakanksha Sharaff, Amit Siddharth Khaire, Dimple Sharma,” Analysing fuzzy based approach for extractive text summarization”, International Conference on Intelligent Computing and Control Systems (ICICCS 2019),

[3] Siya Sadashiv Naik, Manisha Naik Gaonkar,” Extractive Text Summarization by Feature based sentence extraction using rule based”, IEEE International Conference On Recent Trends in Electronics Information Communication Technology (RTEICT),2017,

[4] Kaiz Merchant, Yash Pande ,” NLP Based Latent Semantic Analysis for Legal Text Summarization”,IEEE,2018.

Chapter 6

Appendix

Detailed information, lengthy derivations, raw experimental observations etc. are to be presented in the separate appendices, which shall be numbered in Roman Capitals (e.g. "Appendix IV"). Since reference can be drawn to published/unpublished literature in the appendices these should precede the " Lit. . . .erature Cited" section.

Sample file is shown below...

6.1 Python

Python is a widely used high-level programming language for general-purpose programming that was initially introduced in 1991 by Guido van Rossum. Python is an interpreted language with a design philosophy that prioritises code readability (notice the use of whitespace indentation to delimit code blocks rather than curly brackets or keywords) and a syntax that allows programmers to express concepts in fewer lines of code than languages like C++ or Java. Python supports numerous programming paradigms, including object-oriented, imperative, functional programming, and procedural approaches, and has a dynamic type system and automated memory management. It contains a sizable and well-rounded standard library. Python interpreters for a number of operating systems are available, allowing Python code to execute on a wide range of platforms. The standard version of Python, CPython, as well as virtually all of its variant implementations, is open source software with a community-based development strategy. The Python Softwar eFoundation, a non-profit organisation, is in charge of CPython.

6.2 Natural Language Processing

Natural Language Processing (NLP) is a branch of computer science that includes aspects of human language and artificial intelligence. Machines utilise this technology to comprehend, analyse, manipulate, and interpret human languages. It aids developers in organising their information in order to execute tasks like translation, automated summarization, Named Entity Recognition (NER), audio recognition, relationship extraction, and topic segmentation.

6.3 FFmpeg

FFmpeg is a full audio and video recording, conversion, editing, and streaming solution. It's a command-line video programme that runs on Windows, Mac OS X, and Linux. It can convert between a wide range of video and audio formats.