

# **YouTube/Podcast Summarizer using AI**

A major project report submitted in partial fulfilment of the requirement  
for the award of a degree of

**Bachelor of Technology**

in

**Computer Science & Engineering/Information Technology**

*Submitted by*

**Mitushi Kohli (201174)**

**Ansh Choudhary (201302)**

*Under the guidance & and supervision of*

**Dr. Deepak Gupta**



**Department of Computer Science & Engineering and  
Information Technology**

**Jaypee University of Information Technology,**

**Waknaghat, Solan - 173234 (India)**

# CERTIFICATE

This is to certify that the work which is being presented in the project report titled “**YouTube/Podcast Summarizer using AI**” in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Ansh Choudhary(201302) & Mitushi Kohli(201174).” during the period from August 2023 to May 2024 under the supervision of Dr. Deepak Gupta, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.



Mitushi Kohli  
(201174)



Ansh Choudhary  
(201302)

The above statement made is correct to the best of my knowledge.



18/05/24

Dr. Deepak Gupta

Assistant Professor (SG)

Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat

# CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled '**YouTube/Podcast Summarizer using AI**' in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Wagnaghat is an authentic record of my work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Deepak Gupta** Assistant Professor (SG) Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Mitushi  
18/05/24

(Student Signature with Date)

Student Name: Mitushi Kohli

Roll No.: 201174

Ansh  
18/05/24

(Student Signature with Date)

Student Name: Ansh Choudhary

Roll No.: 201302

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Deepak  
18/05/24

(Supervisor Signature with Date)

Supervisor Name: Dr. Deepak Gupta

Designation: Assistant Professor (SG)

Department: CSE & IT

Dated:

# ACKNOWLEDGEMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully.

We are grateful and wish our profound indebtedness to Supervisor Dr. Deepak Gupta, Assistant Professor in the Department of CSE Jaypee University of Information Technology, Wagnaghat. Deep Knowledge & and keen interest of our supervisor in the field of “Artificial Intelligence” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts, and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr. Deepak Gupta, Department of CSE, for their kind help to finish this project.

We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

# TABLE OF CONTENTS

<b>LIST OF ABBREVIATIONS</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>LIST OF TABLES</b>	<b>ix</b>
<b>ABSTRACT</b>	<b>x</b>
<b>1 INTRODUCTION .....</b>	<b>1-7</b>
1.1 Introduction.....	1
1.2 Problem Statement.....	3
1.3 Objectives .....	4
1.4 Significance and motivation of the project report .....	5
1.5 Organization of project report.....	7
<b>2 LITERATURE SURVEY .....</b>	<b>8-15</b>
2.1 Overview of relevant literature .....	9
2.1.1 Introduction.....	9
2.2 Key gaps in the literature .....	14
<b>3 System Development.....</b>	<b>16-31</b>
<b>4 Requirements and Analysis.....</b>	<b>17</b>
4.1 Project Design and Architecture.....	19
4.1.1 Methodology.....	19
4.1.2 Data Preparation... ..	22
4.2 Implementation .....	22
4.2.1 Implementation using Google Gemini Pro.....	22
4.2.2 Implementation using Hugging Face.....	28
4.2.3 Implementation of two Parallel Videos.....	28

4.3 Key Challenges.....	29
<b>5 Testing.....</b>	<b>32- 39</b>
5.1 Tools used in project.....	34
5.2 Test Cases and Outcomes.....	38
<b>6 Results and Evaluation.....</b>	<b>40-44</b>
6.1 Results.....	41
<b>7 Conclusions and Future Scope .....</b>	<b>45-47</b>
7.1 Conclusion.....	46
7.2 Future Scope.....	48
<b>REFERENCES.....</b>	<b>49-50</b>

# LIST OF ABBREVIATIONS

<b>Abbreviations</b>	<b>Meaning</b>
LLM	Large Language Model
AI	Artificial Intelligence
GPT	General Pre-trained Transformers
NLP	Natural Language Model
URL	Uniform Resource Locator
API	Application Programming interface
Standlibs	Standard Library
BERT	Bidirectional Encoder Representations from Transformers

# LIST OF FIGURES

<b>S. No.</b>	<b>Title</b>	<b>Page No.</b>
1	Figure 3.1: Flow Graph of The Project	20
2	Figure 3.2: Flow Graph with LLM	20
3	Figure 3.3: Flow Graph with YouTube Transcript and LLM	21
4	Figure 3.4: Flow Graph for Long YouTube Video/Podcast	21
5	Figure 3.5: The download_transcript function extract its transcript.	22
6	Figure 3.6: Transcript is extracted using the YoutubeTranscript API.	25
7	Figure 3.7: The Gemini API is called, and it starts summarization	25
8	Figure 3.8: Cosine Similarity function is used to calculate the similarity.	25
9	Figure 3.9: Heatmap is plotted between the transcript and summary	26
10	Figure 3.10: Fetches the transcript of a YouTube video from a user-provided URL	26
11	Figure 3.11: Utilizes a Hugging Face summarization pipeline to summarize the transcript	27
12	Figure 3.12: Generates summaries for each chunk	27
13	Figure 3.13: Calculates the cosine similarity between the transcript and the final summary	28
14	Figure 3.14: Creates a heatmap to visualize the similarity score between the transcript and summary.	28
15	Figure 3.15: The important libraries were installed	29
16	Figure 3.16: A function is made to get the transcript of the videos	29
17	Figure 3.17: It summarizes the data using the model being used	29
18	Figure 3.18: ThreadPoolExecutor to fetch transcripts in parallel for the provided YouTube video URLs.	30
19	Figure 4.1: Transcript is downloaded.	39
20	Figure 4.2: Output when incorrect URL is input.	39



21	Figure 4.3: Output when no URL is input.	40
22	Figure 5.1: User must enter the link of video which is to be summarised.	41
23	Figure 5.2: Generated summary of the YouTube video.	42
24	Figure 5.3: Similarity score between Transcript and generated Summary.	42
25	Figure 5.4: Similarity score between Transcript and generated Summary	43
26	Figure 5.5: Similarity score between Transcript and generated Summary	44
27	Figure 5.6: Result of multiple video summarization	45

# LIST OF TABLES

2.1 Summary of the Relevant Literature.....	10
---	----

# ABSTRACT

The boom of video content in today's age of information overload places increasing demands on end-users to digest more and better-quality information than ever before. However, this is a challenge that calls for a new tactic—using Artificial Intelligence (AI) to automate video summarization in this project. YouTube video summarizer utilizes Natural Language Processing (NLP) and deep learning approaches to comprehend, condense, and identify the main aspects of a video.

This is a project of a YouTube video summary leveraging the Large Language Model (LLM) for language comprehension and content abstraction. The summarizer uses the abilities of LLMs like Google Gemini Pro and HuggingFace models. It extracts cognitive insights from text-to-text transcribing of videos by integrating linguistic subtleties into environmental context.

The methodology involves input retrieval, processing, and incorporation of LLMs in the summary generator. LLM is enhanced language understanding models that generate summaries for videos in a holistic manner.

The evaluation demonstrates that the proposed YouTube video summarizer is quite successful and an effective tool for improving accessibility in video-sharing systems and user experiences. The project assesses the advantages and disadvantages of the model, which will lay the foundation for future studies and improvement.

This provides another input into the discussion of automated video summaries assisting content providers, viewers, and software manufacturers. With the changing face of video technology, the development of artificial intelligence within video summarization plays an important role in negotiating the vast ocean of internet video data.

# CHAPTER 1

## INTRODUCTION

### 1.1 INTRODUCTION

One of the most prominent digital worlds is YouTube where we can get educational, entertaining storylines or other fascinating video clips. Though that may constitute success in democracy by YouTube itself, it also presents a further problem for the audience who find it difficult to separate beneficial information by wading through seas of videos.

Nonetheless currently, in the realm of the internet flooded with videos, YouTube is a treasure chest with information, entertainment, and wisdom. The massive archive of videos created through the democratization of content generation has been possible due to millions of people uploading their stories, skills, and ideas onto the web. However, this comes with another hurdle to users, how will they find usable and useful items in such a big pool of content?

Recognizing this challenge, our project endeavors to introduce a transformative solution: Weaving the sophistication of a YouTube video summarizer together with the sophistication of AI fabric. The key objective here is to distill it to the essence, in summary form, which are the highest hopes for improvement of the user experience. Further, it helps solve the issue of overload with information, as well as facilitating the decision-making process of users regarding what should be checked out.

As the new paradigm of video media becomes the main means, there is a demand for innovative solutions that enable new ways of watching videos. This project addresses this need by introducing a cutting-edge solution: A Smart YouTube video summarizer with AI. The summary generated tries to alter viewers' habits of watching, finding, or using videos online.

The motivation for this project lies in the recognition of the increasing information overload faced by users at websites like YouTube. They are also limited by lack of time and information overload in trying to go through many lengths of video clips to get some important parts that they want to digest. The issue under consideration concerns the

design of a YouTube Video Summarizer that aims to improve the ease of viewing and interacting with online videos. This is important for the successful operation of this project. It could change the way we communicate with users for YouTube and related media. To make it simpler for end-users, a summarizer has applied the newest NLP advances such as advanced natural language models and artificial intelligence to produce a summary of short but relevant videos. On the other hand, apart from facilitating fast content intake, it provides power for people to choose if such videos can be viewed briefly or in detail. LLM models and AI will therefore be whittler in making the Summarizer. Essentially, whenever we want to tell what exactly the scene entails, it is necessary to observe image and audio effects together with major happenings taking place when the characters talk. This problem is being solved, thanks to the development of LLM which makes it possible for computers to decipher these phrases as if humans themselves were saying them.

Similarly, large language models can be viewed as a group of superheroes for reading text. Consequently, they excel in understanding what an individual word means and its coherence to other words. To this end, we will be able to understand what they are speaking about when viewing their video. However, YouTube and other media have thousands of channels through which we can play virtually any video imaginable – from informal to formal educational videos. Thus, since the range is quite wide, the LLMs are very useful due to their flexibility. This kind of video summarizer would suit any content, which would employ the casual voice of a vlogger or the more formal tone of an instructional video.

Video Summarizer does not choose any words and phrases randomly, in this case it uses LLMs because they understand the language of this specific video. For our summaries, our application is going to extract the transcript of the video and summarize it. It is like there's a computer that understands what is being presented in the video. Therefore, in short, LLMs function like language brains making our initiative. This helps ensure that our summarizer does not only look at the contents of videos, but it also understands what each video means. Provides computers with a type of superpower to identify the most important parts of a video out of several in just a few seconds. The main principle of this project involves the fusion of large language models, such as Google Gemini Pro and HuggingFace, enabling detailed analysis of videos' visuals and speech. Some degree of revolution is involved when using LLMs for video summarization as such algorithms

understand the language well and consider context and content. It has been more than mere summarization; rather building a versatile web tool with today's fluidity involved in clips. The goal was to develop a productive method for developing substantial synopses on emerging domains of User-Generated Content that can learn by itself.

This project has targeted revolutionizing internet video consumption offering easy-to-use interfaces allowing users to navigate through an ocean of information they find online, and finally, opening new avenues for future technology.

## **1.2 PROBLEM STATEMENT**

In the vast world of online videos, especially on platforms like YouTube, there's a big problem: insufficiency of time due to an overload of content. People are always interested in seeing entertaining items, something unusual, or just having some amusement. However, after a while of watching all these videos may become tiresome. This is the reason why we developed YouTube Video Summarizer.

However, the biggest issue is that users encounter numerous information problems. Thousands of video clips can be found that address practically every imaginable topic. Therefore, one cannot just figure out which one among a million videos is worth watching the first minute he watches it. Some individuals may be looking for certain things in a long video clip / some people cannot watch the whole movie. The conventional techniques of using manual tags or classes. Nevertheless, the massive volume of freshly generated information today has rendered these methods obsolete beyond the point of reparation. This means users are not able to find things they want by simple routes.

However, the plethora of available choices overburdens the end-users while the conventional means of content filtering, such as tagging, classification, and search engines fail to tackle the emergence of all these videos covering assorted themes. Information explosion means, in some respects, cases when a person does not manage to find necessary and useful content, they seek it.

Users also have time constraints that make it impossible for them to watch every video just to know whether it will be useful to them. This has created a demand for a tool that can compress a video into a condensed and useful summary such that the end user can choose those videos worth exploring.

Our objective in the course is to find a solution to the challenge by creating a self-generating summarizer for YouTube videos. Briefly, we hope to help users decide in the first couple of seconds or minutes whether this clip is useful for them. This procedure does not necessarily cater only to the audience but also gives way for television programs to adapt to the changing video arena.

As technology gets advanced, people's expectations increase as well. People demand more intelligent instruments that can be ahead of them and provide decisions quickly. The goal our YouTube Video Summarizer project aspires to meet is to narrow the gap that continues to widen between tons of videos available, while viewers are craving more engaging experiences. Now, our challenge is to summarize online videos that will make online videos more accessible and fun.

### **1.3 OBJECTIVES**

With this YouTube Video Summarizer project, we hope to take advantage of the potential that AI gives and improve the user's experience as well as the time spent watching videos on YouTube. The project seeks to address the following key objectives:

#### **1. Efficient Content Consumption:**

Provide a quick and simple method of user consumption of YouTube videos. Offer meaningful, concise, and summaries with the main goal of addressing "overload" information, which viewers will use to choose what videos to watch.

#### **2. Adaptability to Diverse Content:**

Create a strategy that handles videos found on YouTube such as educational materials, entertainment, tutorials, and so forth. Ensure that the YouTube video summarizer is adaptable and can comprehend and summarize material in various disciplines and dialects.

#### **3. User Empowerment:**

Providing users with an effective tool that allows them to find out its applicability within a short period before they watch the entire video. Simplified movement through YouTube's vast video database to provide an enhanced holistic use experience.

#### **4. Integration of Advanced AI Models:**

Utilize up-to-date AI models, particularly LLMs, for advanced video summarization practices. What are some ways that such models could understand the context and come up with appropriate summaries given the dynamism of user-driven material?

#### **5. Continuous Learning and Improvement:**

Create a guide for iterating and enhancing the summarization model as time passes by. Devise ways through which the system can respond to emerging trends, changes in users' interests, and the evolving character of YouTube content.

#### **6. Contribution to the AI-Driven Content Landscape:**

Make a valuable contribution to the wider discussion around AI-generated curation and summary. Consider how new AI technologies can improve access and usefulness of web-based videos. Primarily the YouTube video summarizer project investigates reworking how a person interacts with YouTube Videos. The use of a summary enables easy accessibility to the core information in videos.

### **1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK**

This YouTube Video Summarizer project is significant in that it aims to change the way users view and consume content on the famous YouTube platform. The influence of this tool is multidimensional. It can affect user experience as well as the issue regarding the integration of AI in content curation.

The project involves an effort geared towards improving content utilization through the improvement of consumption efficiency. The old system that involves manually listing videos as well as relying on search engines fails to grant viewers an at-a-glance preview of all available material. The project will use state-of-the-art AI models such as LLMs to analyze not only the visual but also language aspects of videos, enabling detailed perspective. It makes summarization easier and provides a versatile tool that adjusts to the varied subject matter in the platform.

In addition, the YouTube Video Summarizer highlights the transformative power of LLMs for video summarization. These models can be integrated into the project to enable the system to understand the spoken or written word in a video, moving past the usual



visual signs. Such in-depth comprehension allows the creation of outlines with not only pictures' meaning but also linguistic details transmitted by speech. This way the project helps advance AI applications for multimedia content analysis and description.

Finally, the success of the YouTube Video Summarizer project is based on its potential to reshape YouTube usage by introducing a tool that simplifies both consumerism and entertainment. The project aims to deal with issues such as information overload, advanced artificial intelligence modeling, and contribution to the wider discussions concerning AI and content curation.

This realization that users have a shared problem with copious amounts of web material, especially on websites such as YouTube motivated the YouTube Video Summarizer project. With the overwhelming growth of the digital world with thousands of videos on a wide range of topics, audiences are typically faced with one big challenge: how to search for videos that suit their interests and requirements in a fast way?

Indeed, the main purpose behind that is to respond to the term "information overload". Millions of videos are uploaded every day, and it is hard to find from all these movies those that may be useful, meaningful, or interesting. It is difficult to decide what to watch due to such huge amounts of choices that users must scroll through and become frustrated by, leading to a waste of their time.

Additionally, the motivation goes further into making YouTube better for viewers. To this end, an interactive platform will be created whereby users can view snippets of the videos and get previews to help them establish if the videos contain topics they are interested in. Hence, this initiative seeks to provide a more user-friendly experience for use in the digital arena.

The other reason is the drive to be at the cutting edge of technology by adopting state-of-the-art artificial intelligence, specifically LLMs. This model is aimed at enhancing the ability of such models to interpret not only visual clips but also audio, text, and images included in the video. The entire approach is consistent with a change in user expectations and the prospective power of AI in content analysis.

Overall, the reason for developing the YouTube Video Summarizer project is to make people interact favorably with online videos. It seeks to make content discovery easier for the users, save them time, as well as boost their pleasure by proposing a device that complies with users' changing tastes and ever-changing online materials. The motivation

for the project lies in the view that technology should not only be about providing people with a digital experience but also one that is easily accessible, effective, and joyful.

## **1.5 ORGANIZATION OF PROJECT REPORT**

### **Chapter 1: Introduction**

In this chapter, we get to know about the need to summarize lengthy YouTube videos/Podcasts as content consumption is growing with time. With the advancement of technology and the boom of the use of ChatGPT, we need effective tools for summarizing our content as we do not have much time to spare. The aim is to save time and get the most relevant information from the video/podcast.

### **Chapter 2: Literature Review**

In this chapter, we discuss the various summarization techniques used in the past like NLP techniques. We learn about the different types of summarizations the Machine Learning models can perform, one being abstractive and the other being extractive. We also discussed the growth of LLM and how various applications can be built using it.

### **Chapter 3: System Development**

In this chapter, we discuss the various stages of the project and the necessary libraries required for the development. We also discussed the methodology and the problems faced during the development.

### **Chapter 4: Testing**

This chapter includes the different tests performed and the different testing strategies that have been used and will be used in the future.

### **Chapter 5: Results and Evaluation**

In this chapter, we discuss the various results we obtained from implementing the different models and different summarization techniques. We have used hugging face and OpenAI for our implementation and then evaluated the results obtained from them.

### **Chapter 6: Conclusion and Future Scope**

This is the last chapter and here we discuss the summary of the project along with its limitations, along with suggesting improvements in the field of summarization using LLM. This chapter also discusses the future scope of the project. It also gives a perspective on LLM and its future scope and applications.

# CHAPTER 2

## LITERATURE SURVEY

### 2.1 OVERVIEW OF RELEVANT LITERATURE

#### 2.1.1 INTRODUCTION

The collection of research papers explores various aspects of automatic text summarization and the development of applications utilizing LLMs. A wide range of strategies and challenges related to text and video summarizing are covered in the literature study. One method uses NLTK and Spacy for summarizing articles and YouTube transcripts, emphasizing the benefits of extractive summarization while battling noisy text and inadequate abstractive summarization powers. Another article highlights security problems and investigates the relevance of the LangChain framework in quickly constructing LLM applications. Third research, which cites dataset size limits, investigates learning to summarize using LLMs with the CNN/Daily Mail dataset and the BART model. Furthermore, a YouTube transcript summarizer that makes use of Latent Semantic Analysis and Cosine Similarity has difficulties in efficiently controlling word count. Additionally, Spacy's NLP-based automated text summary works well for producing targeted summaries, but it does not go into detail on its drawbacks. Progress in abstractive grounded summarization of podcast transcripts is shown, while acknowledging the challenges presented by voice recognition mistakes. Finally, an abstractive summarizer for YouTube videos is suggested, recognizing the constraints of efficiency and scalability, especially when dealing with huge datasets. When taken as a whole, these studies provide insights into how summarizing approaches are changing and the challenges they face.

S.No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1	Article and YouTube Transcript Summarizer Using Spacy and NLTK Module [Smith, J., & Johnson, A. (2023)]	SSGM Journal of Science and Engineering [2023]	The NLTK package is utilized for text Processing.	The use of Spacy and NLTK modules for text processing such as tokenization, entity recognition and calculating word frequency	Limited support for abstractive summarization and challenge in handling 'noisy text'.
2	Creating Large Language Model Application Utilizing LangChain: A Primer on Developing LLM Apps Fast [O. Topsakal and T. C. Akinci 2023]	[ICAENS 2023]	Idea was given to use the Langchain framework to develop applications utilizing large language models.	Insights into LangChain's usage, fostering rapid application development using LangChain for LLM applications.	Security concerns in LLM application development
3	On Learning to Summarize with Large Language Models as References [Yixin Liu, Alexander R. Fabbri 2023]	Submitted on 23 May 2023 at Cornell University	lar/DailyMail dataset Used Bart - Large CNN model	Used contrastive learning method based on reward system utilizing LLMs for summarization.	Test done on a small dataset which may affect the efficacy of model
4	YouTube Transcript summarizer [Sulochana Devi, Rahul Nadar 2022]	[IRJMETS 2022]	Latent Semantic Analysis [LSA] Cosine Similarity.	Automatic summarization using NLP grounded algorithms aims to produce short videotape	LSA can't manage word count. Limits the ability of NLP to understand data.
5	Implementation of NLP based automatic text summarization using spacy [N. C. P. Prakash et al]	[IJHS 2022]	Spacy Algorithm, linguistic & Statistical Features	The use of Spacy algorithm results in fewer iterations and more focused Summary	Does not discuss the limitations or performance of the Spacy and NLP model used.
6	Towards Abstractive Grounded Summarization of Podcast Transcripts [K. Song, C. Li, X. Wang, D. Yu, and F. Liu]	Association for Computational Linguistics [2022]	Abstractive Summarization, Large Podcast dataset	Improved Summarization Techniques in terms of Automatic Human evaluation	Speech Recognition Errors are abundant
7	Abstractive Summarizer for YouTube videos [S. Tamane et al.]	[ICAMIDA 2022]	Hugging Face Transformer, REST-API for Backend Request	Combining multiple approaches to achieve ideal outcome.	It includes large amounts of data which results in limitations of scalability and efficiency.
8	Automatic Summarization of YouTube video using TF-IDF. [R. Albeer, H. Al-Shahad, H. Aleqabie, and N. Al-Shakarchy]	[IJEECS 2022]	Term Frequency - Inverse Document Frequency [TF_IDF]	TF-IDF was evaluated using the Rouge method on the CNN-daily-master dataset.	TF - IDF may become slow and memory consuming when dealing with extensive datasets.
9	Natural Language Processing (NLP) based Text Summarization - A Survey [I. Awasthi, K. Gupta, and P Bhogal]	ICICT [2021]	Abstractive methods, Extractive methods, long short-term memory (LSTM) RNN model.	Less repetitive and more concentrated summaries.	Less repetitive and more concentrated summaries.

Table 2.1: Literature Overview

**Smith, J., & Johnson, A. [1]** These days, many students and professionals find it convenient to use summation tools that produce condensed summaries from long texts thereby conserving time. Individuals can use these tools to elaborate and explain complicated concepts clearly and generate summaries ranging in length depending on their interests. Summarization is about coming up with an abridged form of a longer text that carries the essential ideas excluding irrelevant data. Summarizers rely on a rule-based approach or natural language processing algorithms to mark key sentences and phrases. On the other hand, abstractive summarization is more complicated as it entails producing a fresh sentence. Incorrectness may be found in the product (including erroneous spelling or meaning). There are many ways errors can be detected such as measuring the original text with rouge or bleu metrics in comparison with the summary. The software packages are also applicable in summarizing files that might have been stored on a computer like articles and research papers. The file's text can be retrieved and summarized, allowing one to save as a file.

**O. Topsakal and T. C. Akinci [2]** is about LLMs, and it emphasizes LangChain, a popular open-source technology used for expediting the development of artificial intelligence apps. Customizable pipelines with modular abstractions are some qualities of LangChain built for custom LLM-based applications. Its speed of development can be evidenced by real-world examples. A detailed description of LangChain's structure shows how it enables fast-paced development and innovation. To carry out map-reduce processes, refine methods, as well as numerous data operations; that's what it does. In essence, LangChain is perceived as an important instrument in AI system construction thanks to its data-friendly, flexible, and versatile characteristics. The article demonstrates how to develop a speedy application via using LLMs with a focus on LangChain, an open-source giant. Since it effectively manages multiple data sources, LangChain is very important for LLM application projects. Real-life scenarios show how LangChain speeds up the development of "LLM" applications by using a modular approach and variable pipelines. The research continues with map reduce and refined approaches, which are critically important in LLM systems. In addition, LangChain assists in the execution of map-reduce operations where questions are processed concurrently with documents.

**Yixin Liu Alexander R. Fabbri [3]** investigates a new learning paradigm for text summarization models treating LLMs as a reference ordeal, gold standards. In this regard, the authors propose a contrastive learning algorithm with GPT Score and

GPTRank approaches and demonstrate experimentally. This research paper looks into a new learning paradigm for text summarization models treating LLMs as a reference ordeal, gold standards. In this regard, the authors propose a contrastive learning algorithm with GPT Score and GPTRank approaches and demonstrate experimental the present paper presents a newer way of learning for text summarization models through an LLM as the reference or golden oracle.

**Sulochana Devi, Rahul Nadar [4]** suggests an automated summarization approach for color YouTube videos based on NLP-grounded methods attempting to compose summaries. Here, this algorithm summarizes YouTube's videotape reiteration and develops a videotaped abstract. They develop a web operation that takes input as a Youtube videotape link and the desired summary duration to produce an abridged Youtube video summary. This paper uses the Term Co-occurrence Matrix, which describes in what sentences the data belong together, and LSA which takes important components from the set. In general, this paper presents the idea of using NLP algorithms to summarize YouTube videos and thus to provide an easy way of creating short summaries from the user's point of view. A Term Co-occurrence Matrix and LSA help to understand sentence relationships and obtain significant attributes from the dataset.

**N. C. P. Prakash et al [5]** stated an NLP-based Automatic Text Summarization using the Spacy algorithm. There exists an endless amount of data available on the Internet, hence it is impossible to read all this data. Moreover, transforming raw information into valuable knowledge becomes extremely challenging in such a case. The paper focuses on two major text summarization approaches which are extractive and abstract. Specifically, it discusses the extractive method that uses NLP. In the extraction method, the statistical and linguistic meanings of sentences are calculated. Lastly, the paper discusses how spacy worked in their project.

**K. Song, C. Li, X. Wang, D. Yu, and F. Liu [6]** proposes a novel abstractive summarization method for podcast transcripts, aiming to address challenges such as factual inconsistencies, speech disfluencies, and recognition errors in summaries. The approach learns to produce abstractive summaries while grounding summary segments in specific regions of the transcript, allowing for full inspection of summary details. The proposed method is evaluated on a large podcast dataset and shows promising results in terms of improving summarization quality, both in automatic and human evaluation. Grounded summaries bring clear benefits in locating inconsistent information between

the summary and transcript segments. The paper investigates podcast summarization to produce textual summaries that help listeners understand why they might want to play those podcasts. The summaries include spans of text tethered to the original audio, enabling users to interpret system-generated abstracts in context. The utility of the proposed approach is demonstrated through experiments on a benchmark dataset.

**S. Tamane et al. [7]** presents a user interface that utilizes NLP and Machine Learning to generate summaries of YouTube videos. The goal is to address the challenge of finding relevant content among the enormous number of videos uploaded daily. The summarization model extracts video transcripts and generates concise summaries while preserving important information. The paper focuses on abstractive summarization, which involves analyzing the text and generating coherent summaries using paraphrasing techniques. The authors compare abstractive and extractive summarization methods, discussing the pros and cons of each approach. The implementation process is ongoing, but the paper provides an overview of the structure and studies conducted. The proposed system can be extended to other streaming services and support multiple languages for the summarized text.

**R. Albeer, H. Al-Shahad, H. Aleqabie, and N. Al-Shakarchy [8]**, The construction of an automatic summarising system for YouTube video transcription text using the term frequency-inverse document frequency (TF-IDF) method is the main objective of the paper titled "Automatic summarization of YouTube video transcription text using term frequency-inverse document frequency". To deliver essential information to consumers who might not have time to watch lengthy films, the research attempts to extract crucial keywords from the video transcription text and provide a summary. The suggested solution uses the TF-IDF approach in conjunction with natural language processing techniques to determine crucial keywords for the summary.

**I. Awasthi, K. Gupta, and P Bhogal [9]** In view of the internet's exponential data growth, the article discusses the need for text summaries and presents automatic summarising to transform unstructured data into meaningful understandings. It highlights the extractive strategies that are preferred by academics studying NLP by dividing text summarising techniques into two categories: extractive and abstractive. The study looks at various extractive and abstractive methodologies for text summary to provide less repetitive and more focused summaries. Additionally, by rewarding or penalising an agent for producing the best summaries, the study examines how

reinforcement learning techniques might be used to text summarization.

## **KEY GAPS IN THE LITERATURE**

The gap that was observed was that some researches were exhaustive in its evaluation or comparison of the performance of the Spacy and NLTK modules for article and YouTube transcript summarization. Moreover, discussion over possible limitations encountered when using the HuggingSound library for speech recognition and audio chunking were not mentioned.

Another gap that was noticed was the difficulty to critique in terms of its practicality as the paper does not include any cases nor is it backed by empirical evidence on just how fast the development of LLM-based applications using LangChain can be. A complete evaluation or contrast of LangChain with other relevant frameworks or tools is not done in the article, which limits our understanding of what makes this system unique or not. The study does not address any potential problems or limitations of using LangChain like compatibility issues or scalability problems with different platforms and code assessing LLMs and Langchain in the development of AI applications may lead to bias or ethnic problems like false or biased results. The study does not discuss these issues.

The proposed paradigm will work for the other summarization datasets or domains cannot be told regarding this specific research paper. However, the papers do not go beyond discussing the possible challenges that may be associated with employing LLMs to function as reference or gold-standard oracle in text summarization. In the papers, the experimental studies are only used for evaluating smaller summarization models that use the suggested approach in comparison with a referential LLM. The articles don't benchmark the presented approach against other existing state-of-the-art summarization models and techniques.

Although some papers highlight the evaluative criteria for the proposed summarization algorithm, they fail to explicitly state the assessment metrics. Limited dataset description: However, there is no information about the large size or diversity of the data set on which the algorithm was trained and tested. Lack of comparison with existing methods: However, the papers do not make any comparative analyses between its proposed algorithm and the already existent summarization methods, hence making it difficult to assess how effective it is compared to them.



Papers that primarily supports extractive summarization do not acknowledge the problems with inferencing and in making sense of the setting. Dealing with lexical ambiguity/polysemy words. Find it difficult to deal with domain-specific vocabulary. Limited sentence compression capabilities. Extracted sentences may be redundant.

In papers where speech recognition systems are used the main issue with those are not discussed like they may falter in accurately transcribing spoken content due to diverse linguistic variations such as accents, dialects, and pronunciation differences. Additionally, ambient noise, background sounds, and poor audio quality can further degrade the accuracy of transcriptions. These inaccuracies in the transcriptions serve as the foundation for abstractive summarization, influencing the quality of the generated summaries. Abundant speech recognition errors can pose significant challenges in the context of abstractive summarization. When dealing with transcriptions generated by speech recognition systems, several factors contribute to the increased likelihood of errors, impacting the subsequent summarization process.

The research paper that discusses the abstractive summarization methods in general have certain limitations, such as the potential for generating inaccurate or misleading summaries and the challenge of maintaining coherence and readability in the generated summaries. The effectiveness of the abstractive summarization approach for YouTube videos may depend on the quality and accuracy of the automatic transcription of the videos, as well as the complexity and diversity of the content being summarized.

One drawback is that it ignores the semantic meaning of words and instead relies solely on term frequency. Because TF-IDF favours words that appear frequently, it may miss significant phrases. In addition, it has trouble understanding synonyms, viewing concepts as separate entities. The summarising process may be impacted by the method's sensitivity to document length. Furthermore, TF-IDF does not take word order into account, which could result in summaries that are not coherent. TF-IDF may have trouble allocating the right weights to new phrases, making handling them difficult.

The vanishing gradient problem causes RNNs to perform poorly on extended sequences, which results in partial summarization in longer documents. These models may not fully comprehend the global environment, and they are less useful for texts with intricate organisational patterns because they have trouble representing hierarchical hierarchies. Managing uncommon or out-of-vocabulary terminology can be difficult and lead to

errors. When the training dataset is small, overfitting may become problematic. The body of research on summarising strategies exposes several shortcomings and inconsistencies in different frameworks and approaches. The assessment of Spacy and NLTK by Smith and Johnson is superficial and leaves out important points on the constraints imposed by the Hugging Sound package. In a similar vein another research hinders a thorough understanding of LangChain's uniqueness by failing to compare it with other frameworks and providing no actual evidence of its usefulness.

Abstractive summarization is greatly impacted by speech recognition problems, which can be caused by a variety of language differences, background noise, and low audio quality. These variables can also lead to transcribing errors. The coherence of the TF-IDF summarizing approach may be impacted by its tendency to ignore important phrases, have trouble with synonyms, and disregard word order. The vanishing gradient problem makes it difficult for RNNs to summarize lengthy documents. They also have trouble managing terms that are not in the dictionary, modeling hierarchical structures, and handling limited training datasets. All in all, the gaps draw attention to the necessity of doing more thorough analyses, comparisons, and assessments of the practical difficulties that arise in summarizing research.

# CHAPTER 3

## SYSTEM REQUIREMENT

### 3.1 REQUIREMENTS AND ANALYSIS

The design requirements for a YouTube / Podcast summarizer utilizing a LLM. Here is an outline of the requirements and an analysis of the task. The exact LLM model which we select, the scope of our project, and the features we included influenced the system requirements for a YouTube/podcast summarizer. Here are a few broad things we kept in mind:

#### 1. **Hardware:**

GPU acceleration is very helpful for LLMs, particularly those built on transformer architectures. Model training and inference can be significantly accelerated by having a strong GPU (such as NVIDIA GPUs like the Tesla V100 or A100) or by having access to cloud-based GPU services (such as AWS, Google Cloud, or Azure).

#### 2. **Software:**

Since LLMs are frequently implemented using these frameworks, make sure they are compatible with well-known deep learning frameworks like TensorFlow or PyTorch. Selecting an LLM that has been trained and is appropriate for the task at hand. Depending on what we need to summarize, models such as GPT-3, GPT-4, BERT [18], or T5 might be useful.

#### 3. **RAM:**

Memory managing the big models and datasets requires enough RAM. The magnitude of the input data and the model will determine how different the requirements are. Make sure to have adequate RAM so that data can be processed.

#### 4. **Storage:**

We required enough storage space to hold training data, model checkpoints, and any extra resources, depending on the size of dataset and the models.

## **5. Network:**

If the summarizer pulls information from external sources (like YouTube or podcast APIs), we need a dependable and quick internet connection.

## **6. Development Environment:**

Assembling the tools required for deep learning, Python development, and any libraries or packages needed for the project.

## **7. The Ability to Scale:**

We intended to implement summarizer on a large scale thinking about building the system with scalability in mind. Maintaining several instances of the summarizer may require orchestration (using Kubernetes) and containerization (using tools like Docker)[13].

### **3.1.1 Analysis:**

Previous research has demonstrated that a large pre-trained language model like GPT-3, Gemini Pro are highly adept at generating high-quality summaries in text summarization tasks. However, it would probably have to be custom fine-tuned [14] specifically on long conversational style transcripts if one needs. Some of these challenges may include scalability issues relating to ingestion and automation of speech-to-text for input data, and limitations on the use of (long text, summary) for custom datasets creation because of poor availability in the conversational domain. It is necessary to carefully assess information retention and compactness. Such a system would make it easy to access brief information from summarized language spoken content where pivotal points are retained.

The YouTube Video Summarizer project involving AI in understanding and abstraction of videos constitutes a great milestone towards enriching YouTube users' experience of content on YouTube. Similarly, it is like a smart agent that watches the video, giving summaries from which we can base our decision on whether to view the whole video or not. The system behind the scenes utilizes advanced computer programs that are able not only to read the video but also detect the surrounding images. YouTube is a good platform for this device as it can handle different kinds of videos, ranging from educational to fun vlogs. Although useful, the project has some problems. It can fail to interpret the meaning of an individual word and difficult compound sentences at times.

Also, it may not be able to get the mood or setting of a video while mainly trying to extract relevant passages. The objective of this project is to streamline YouTube viewership, enabling a quick analysis of whether videos are relevant to watch. Despite this fact, it does what it can, and the team is constantly working on ways of further improvement.

## **3.2 PROJECT DESIGN AND ARCHITECTURE**

### **3.2.1 METHODOLOGY**

#### **1. Data Collection:**

Accessing a wide range of YouTube videos and podcasts to train the model. The dataset ought to include different themes and ways of talking. Transcribe the audio material into text through speech-to-text technology suitable for the linguistic model. We decided to use the YouTube transcript [8] API to load the subtitles of the videos.

#### **2. Large Language Model:**

Selecting among the contemporaneous sophisticated big language models like Google Gemini and alternative models for summarization projects. Another choice was made to use the Hugging Face LLM model as they are easily fine-tuned with prompt engineering.

#### **3. Summarization Algorithm:**

**Abstractive Summarization:** Use an abstractive summarization [17] algorithm such that the model can produce a high-quality summary using its own words.

**Extractive Features:** There are also optional extractive features [15],[18],[24] that can be used to emphasize a few crucial sentences within the source material.

#### **4. User Interface:**

Creating an easy-to-use platform where users can insert YouTube video/podcast links and get the summarized content. Offer alternative ways of presenting the summaries e.g., in text.

#### **5. Scalability:**

Use of parallel processing technique to process many requests simultaneously. The GPU requirements and the hardware requirements must be defined in a manner that is helpful for the developer.[19]

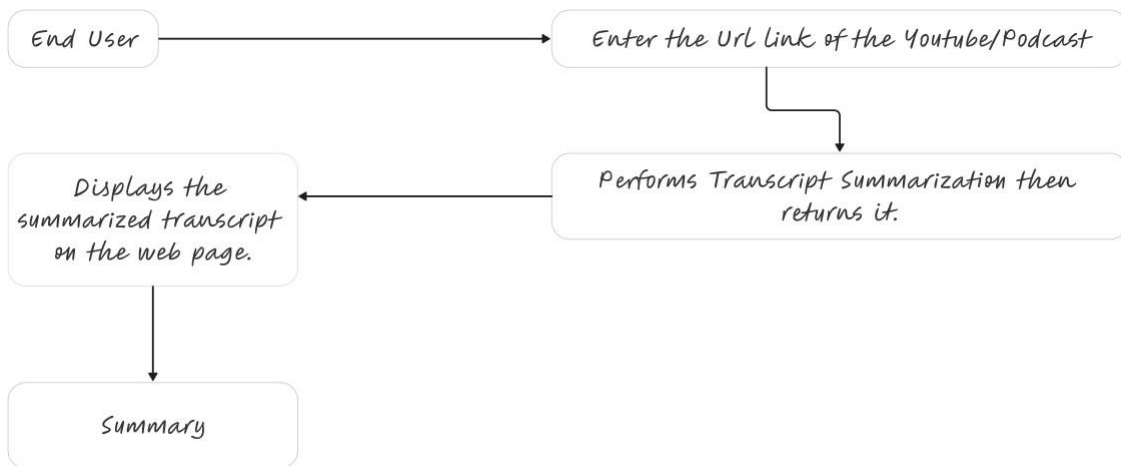


Fig 3.1: Flow Graph of The Project

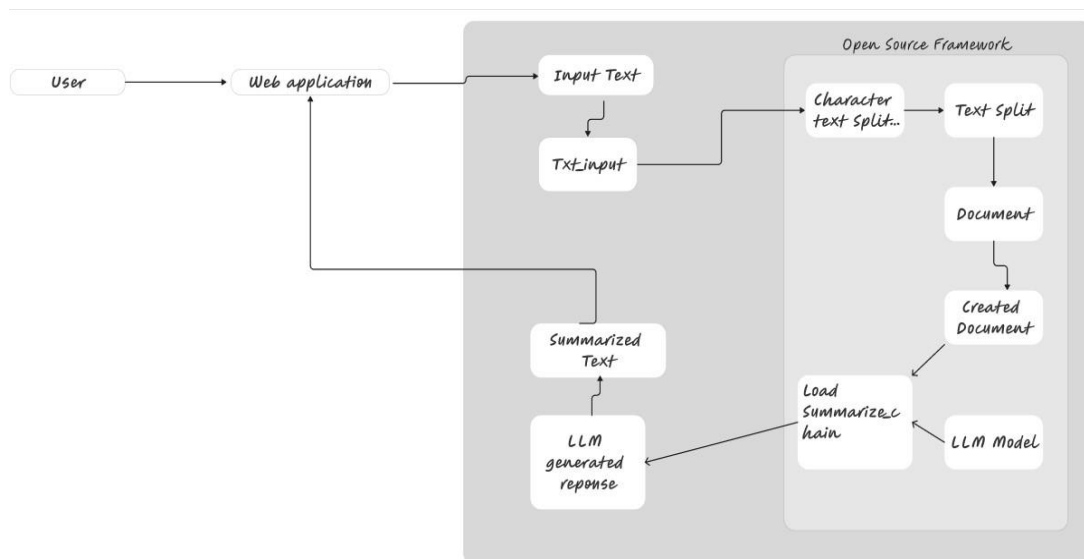


Fig 3.2: Flow Graph with LLM

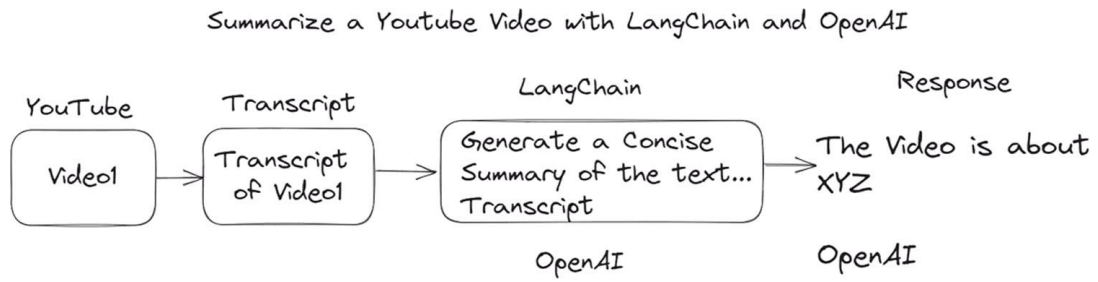


Fig 3.3: Flow Graph with YouTube Transcript And LLM

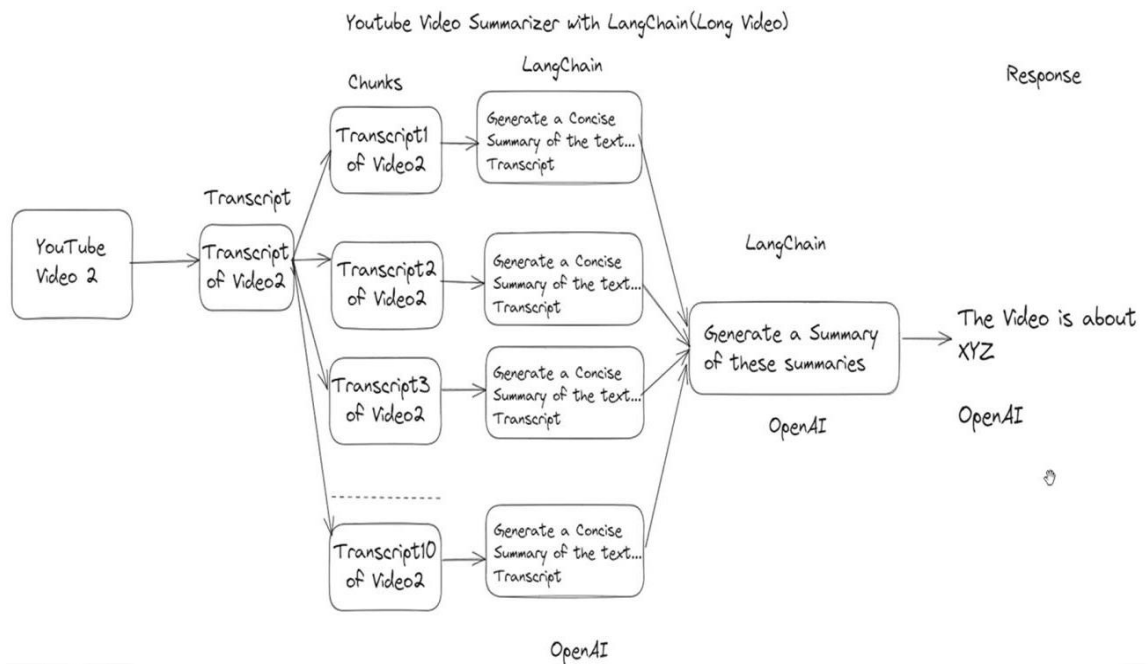


Fig 3.4: Flow Graph for Long YouTube Video/Podcast

### 3.2.2 DATA PREPARATION

The project consists of preparing the data which is been done by downloading the video from a particular YouTube URL and then extracting its transcript after it was extracted Here's the breakdown of the data preparation steps:

The download\_transcript function is defined to download the video and extract its transcript using youtube\_transcript\_api libraries.

```
def extract_transcript_details(youtube_video_url):
    try:
        video_id = youtube_video_url.split("=")[1]
        transcript_text=YouTubeTranscriptApi.get_transcript(video_id)

        transcript = ""
        for i in transcript_text:
            transcript += " " + i["text"]

        return transcript

    except Exception as e:
        raise e
```

Fig 3.5: The download\_transcript function is defined to extract its transcript.

## 3.3 IMPLEMENTATION

The implementation of the YouTube Video Summarizer project LLMs like OpenAI's GPT involves several key steps [21]. To begin, the project sets up the necessary dependencies, including the installation of relevant Python libraries such as LangChain for natural language processing and open-ai for accessing the GPT models. Having set up a proper development environment, the project utilizes the YouTube-Transcript-API to get video details such as names and transcriptions which are then used for summarization at a fundamental level. In the project we are using the below mentioned libraries:

### 1. Matplotlib.pyplot:

To create visualisations like plots, histograms, and charts, we need this package. Matplotlib.pyplot is a tool used in video summarizer projects to visualise information about the movies, including their length distribution, the frequency of specific keywords in the transcripts, and the similarity between various summaries.



## **2. NumPy:**

It is an effective Python library for numerical computation. It is utilised for several activities in a video summarizer project, including mathematical calculations, feature extraction, and data modification. For instance, it is employed in matrix operations for text feature analysis or in the processing of numerical data taken from video metadata.

## **3. Sklearn.metrics.pairwise.cosine\_similarity:**

This function from scikit-learn calculates the cosine similarity between pairs of samples in two datasets. It is used to measure the similarity between different video transcripts or between a reference summary and generated summaries. This can help assess the quality of the summarization algorithm.

## **4. Transformers.pipeline:**

Natural language processing models that have already been trained are accessible through the transformer's library. Text summarization is only one of the many NLP tasks that can be utilised by these models for with ease thanks to the pipeline feature. Transformers. Pipeline can be used in a video summarizer project to create transcript summaries utilising cutting-edge NLP models such as BERT or GPT.

## **5. Youtube-Transcript-API:**

With the help of this Python library, one can obtain the transcriptions, or captions, for YouTube videos. With just a simple link on the page or website hosting the video, visitors can easily access the text transcripts of YouTube videos thanks to this library. This could be useful for a variety of things, such information extraction, video analytics, and even subtitle generation.

## **6. sklearn.feature\_extraction.text.TfidfVectorizer:**

This scikit-learn component is used to create a matrix of TF-IDF features from a set of raw documents. Before being fed into machine learning models for summarization or similarity calculations, the text data retrieved from video transcripts is preprocessed and vectorized using the TfidfVectorizer.

## **7. Seaborn:**

Based on matplotlib, Seaborn is a library for statistical data visualisation. It offers a sophisticated plotting tool for the creation of visually striking and educational graphs.

The seaborn can be used to assess the effectiveness of various summarising techniques, examine relationships between video attributes and summary quality, and visualise the distribution of summary lengths.

#### **8. Time:**

It can be used to measure the performance of different operations such as fetching videotranscripts, processing text data, or training machine learning models. For example, anyone wants to track the time taken to retrieve transcripts from YouTube or to summarize multiple videos.

### **3.3.1 IMPLEMENTATION USING GOOGLE GEMINI PRO:**

We have built a Streamlit application that uses the transcripts of YouTube videos to generate summaries. Importing the required libraries and setting up the Google GenerativeAI library with environment variables' API keys are the first steps. After asking users to enter the URL of a YouTube video, the application uses the YouTube Transcript API to extract the transcript from the video. Next, using the transcript a Generative AI model known as the "Gemini-pro" model is used to provide a summary of the video material. Using TF-IDF vectorization, the application determines how similar the original transcript and the generated summary are. For clarity, it presents the similarity score together with a heatmap visualization. All things considered, the program offers a productive means of condensing video material and illustrating the consistency between the transcript and its synopsis.

```

genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

prompt = """You are Youtube video summarizer.You will be taking the transcript text
and summarize the entire video and providing the important summary in points
within 250 words. Please generate the summary here :"""

def extract_transcript_details(youtube_video_url):
    try:
        video_id = youtube_video_url.split("=")[1]
        transcript_text=YouTubeTranscriptApi.get_transcript(video_id)

        transcript = ""
        for i in transcript_text:
            transcript += " " + i["text"]

        return transcript

```

Fig 3.6: Transcript is extracted using the YoutubeTranscript API.

```

def generate_gemini_content(transcript_text,prompt):
    start_time = time.time()
    model = genai.GenerativeModel("gemini-pro")
    response = model.generate_content(prompt+transcript_text)
    end_time = time.time()
    time_taken = end_time - start_time
    return response.text,time_taken

```

Fig 3.7: The Gemini API is called, and it starts summarization.

```

def calculate_similarity(text1, text2):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform([text1, text2])
    cosine_sim = cosine_similarity(tfidf_matrix)
    return cosine_sim[0, 1]

```

Fig 3.8: Cosine Similarity function is used to calculate the similarity.

```

# Create a heatmap for similarity
fig, ax = plt.subplots()
sns.heatmap([[similarity_score]], annot=True, cmap="YlGnBu", xticklabels=["Summary"],
            | yticklabels=["Transcript"]])
plt.title("Similarity between Transcript and Summary")
st.pyplot(fig)

```

Fig 3.9: Heatmap is plotted between the transcript and summary.

### 3.3.2 IMPLEMENTATION USING HUGGINGFACE MODEL – stevliu/my\_awesome\_billsum\_model

The user enters the URL of a YouTube video, this Python script extracts the transcript and uses a Hugging Face pipeline to summarise the transcript into manageable pieces. To determine how well the summary method encapsulated the main points of the film, it computes the cosine similarity between the original transcript and the created summary. It concludes by printing the amount of time required for the summarization process as well as the transcript and summary similarity score, providing a useful tool for assessing and analysing the summarization of video information.

```

# Define the YouTube video URL
youtube_url = input("Enter a youtube url:")

# Download the video and get its transcript
def extract_transcript_details(youtube_video_url):
    try:
        video_id = youtube_video_url.split("=")[1]
        transcript_text = YouTubeTranscriptApi.get_transcript(video_id)

        transcript = ""
        for i in transcript_text:
            transcript += " " + i["text"]

    return transcript

```

Fig 3.10: Fetches the transcript of a YouTube video from a user-provided URL.

Then, it concatenates the transcript text into a single string.

```

# Extract transcript from the YouTube video
transcript = extract_transcript_details(youtube_url)

# Use Hugging Face pipeline for summarization
pipe = pipeline("summarization", model="stevhliu/my_awesome_billsum_model")

# Set the maximum length of the summary
max_summary_length = 60

# Split the video text into chunks
chunk_size = 1000
chunks = [transcript[i:i + chunk_size] for i in range(0, len(transcript),
                                                    chunk_size)]

# Measure the time taken to generate summaries
start_time = time.time()

```

Fig 3.11: Utilizes a Hugging Face summarization pipeline to summarize the transcript of a YouTube video, splitting the text into chunks for processing.

```

# Generate summaries for each chunk
summaries = []
for chunk in chunks:
    summary = pipe(chunk, max_length=max_summary_length, min_length=27,
                  length_penalty=2.0, num_beams=4, early_stopping=True)
    summaries.append(summary[0]['summary_text'])

# Combine the summaries into a single document
final_summary = " ".join(summaries)

end_time = time.time()

# Calculate and print the time taken
time_taken = end_time - start_time
print(f"Time taken to summarize: {time_taken:.2f} seconds")

```

Fig 3.12: Generates summaries for each chunk of the transcript using the previously defined Hugging Face pipeline.

```

# Compute cosine similarity between transcript and summary
def calculate_similarity(text1, text2):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform([text1, text2])
    cosine_sim = cosine_similarity(tfidf_matrix)
    return cosine_sim[0, 1]

similarity_score = calculate_similarity(transcript, final_summary)
print(f"Similarity Score between Transcript and summary: {similarity_score}")

```

Fig 3.13: Calculates the cosine similarity between the transcript and the final summary using TF-IDF vectorization and prints the similarity score.

```

fig, ax = plt.subplots()
sns.heatmap([[similarity_score]], annot=True, cmap="YlGnBu", xticklabels=
["Summary"], yticklabels=["Transcript"])
plt.title("Similarity between Transcript and Summary")
plt.show()

```

Fig 3.14: Creates a heatmap to visualize the similarity score between the transcript and summary.

### 3.3.2.1 IMPLEMENTATION USING HUGGINGFACE MODEL – Falconsai/text\_summarization

The user enters the URL of a YouTube video, this Python script extracts the transcript and uses a Hugging Face pipeline to summarise the transcript into manageable pieces. To determine how well the summary method encapsulated the main points of the film, it computes the cosine similarity between the original transcript and the created summary. It concludes by printing the amount of time required for the summarization process as well as the transcript and summary similarity score, providing a useful tool for assessing and analysing the summarization of video information. It was observed that the similarity score and the time taken for summarization was different for this model.

### 3.3.3 SUMMARIZATION OF TWO VIDEOS PARALLELY USING HUGGINGFACE MODEL

```
from transformers import pipeline
from concurrent.futures import ThreadPoolExecutor
from youtube_transcript_api import YouTubeTranscriptApi
import re
```

Fig 3.15: The important libraries were installed.

```
# Define a function to get the transcript of a single YouTube video
def get_video_transcript(url):
    # Extract video ID from different URL formats
    video_id_match = re.search("(?:v=|\/videos\/|embed\/|youtu.be\/|\/v\/|\/e\/|watch?v=|&v=%2Fvideos%2F)

    if video_id_match:
        video_id = video_id_match.group(1)
        try:
            transcript = YouTubeTranscriptApi.get_transcript(video_id)
            # Combine the transcript into a single string
            text = ' '.join([entry['text'] for entry in transcript])
            return text
        except Exception as e:
            print(f"Error getting transcript for {url}: {e}")
            return ""
    else:
        print(f"Invalid YouTube URL: {url}")
        return ""
```

Fig 3.16: A function is made to get the transcript of the videos and the transcript is combined in a single string.

```
# Define a function to summarize a single text
def summarize_text(text):
    summarizer = pipeline("summarization", model='facebook/bart-large-cnn')
    summary = summarizer(text, max_length=150, min_length=50, length_penalty=2.0, num_beams=4)
    return summary[0]['summary_text']

# List of YouTube video URLs
youtube_url_1 = input("Enter the URL of the first YouTube video: ")
youtube_url_2 = input("Enter the URL of the second YouTube video: ")
youtube_url_list = [youtube_url_1, youtube_url_2] # Separate URLs into a list
```

Fig 3.17: It summarizes the data using the model being used.

```

# Get transcripts in parallel using ThreadPoolExecutor
with ThreadPoolExecutor() as executor:
    transcripts = list(executor.map(get_video_transcript, youtube_url_list))

# Summarize transcripts in parallel using ThreadPoolExecutor
with ThreadPoolExecutor() as executor:
    summaries = list(executor.map(summarize_text, transcripts))

# Print the summaries
for i, summary in enumerate(summaries):
    print(f"Summary for Video {i + 1}:\n{summary}\n")

```

Fig 3.18: ThreadPoolExecutor to fetch transcripts in parallel for the provided YouTube video URLs.

### 3.4 KEY CHALLENGES

There are various difficulties in developing a YouTube summarizer with hugging facetransformer or other language models. The following are the main obstacles:

#### 1. Variability in Video Content:

YouTube videos come in a variety of subjects, genres, and tongues. It can be difficult for models to handle the diversity of material since they may not be familiar with domain-specific vocabulary or languages.

#### 2. Multimodal Data:

Videos include audio-visual content as well. It is difficult to separate pertinent information from several modalities and incorporate it into a cohesive synopsis. Since language models usually process text, visual data might be overlooked.

#### 3. Large-Scale Data Processing:

The duration, subject matter, and language of YouTube videos can differ greatly. Large-scale video data processing and summarization need scalable infrastructure and excellent algorithms to handle the volume of data.



#### **4. Understanding Context:**

Processing the transcript alone is not enough to grasp the context of a video. It entails doing things that might be outside the scope of language models, such as identifying essential concepts, comprehending the story flow, and distinguishing visual aspects.

#### **5. Taking Care of Loud Transcripts:**

YouTube transcripts are not always accurate and sometimes contain grammatical errors or run-on phrases. To manage such flaws in the input data, the summarizer must be resilient.

#### **6. Processing in Real Time:**

An additional level of complication is introduced when summarizing a video in real-time while it is being uploaded or broadcast. Processing at a fast pace is necessary to keep up with the video content.

#### **7. Bias and Sensitivity to Content:**

It is possible for language models to unintentionally replicate biases found in their training set. It can be difficult to make sure the summarizer generates fair and considerate summaries, particularly when dealing with content that could be divisive or contentious.

#### **8. Metrics for Evaluation:**

A summary's quality is a matter of opinion. It can be challenging to create strong assessment measures that are consistent with human judgment. While frequently employed, metrics like ROUGE – 1, ROUGE- N [10],[11], could miss the subtleties of a well-written summary.

#### **9. A Legal and Ethical Perspective:**

It is important to adhere to ethical guidelines and copyright regulations while summarizing content from YouTube videos. It is imperative to ensure adherence to YouTube's copyright policy and terms of service.

## **10. Scalability:**

Scalability is a hurdle when handling a large quantity of movies and processing them effectively. Sturdy infrastructure is needed to deploy a YouTube summarizer at scale. The major issue that we came across was the unavailability and the noisy transcript. The lack of transcripts, which might impede model creation and training, is one of the main issues. Another barrier that we still face is the Google Gemini pro key restriction on the number of tokens it can access.

The most used summarization model for is the BART model [12] hence in our project we decided to use free HuggingFace models to solve these issues, models like my\_awesome\_billsum\_model and Falcons/text summarization were used. These models proved to be invaluable tools for comprehending the subtleties of transformers and associated parts.

We still undergo persistent issues, such as Google Colab's limited GPU availability. Although the platform has been useful in effectively summarizing some videos, questions have been raised over its resilience. The 12GB RAM limit that Google Colab offers is considered insufficient for any LLM Model calculation, particularly given that LLM computations are known to be difficult and lengthy.

In conclusion, we are still dealing with problems about the accessibility of transcripts, the limitations of Google Gemini pro keys, and the limitations of GPU resources in Google Colab. Despite these obstacles, we have tried using free Hugging Face models to learn about transformer models and associated parts.

# CHAPTER 4

## TESTING

In the testing strategy for the YouTube video summarizer project, a systematic and thorough approach is adopted to guarantee the effectiveness and dependability of the AI-driven system. The process commences with unit testing, meticulously evaluating the correctness of individual components such as video downloading, transcript extraction, and summarization algorithms. This phase ensures that each module operates as intended in isolation.

Following unit testing, integration testing is executed to inspect the harmonious collaboration of these components, uncovering any unforeseen challenges that may arise when different modules interact. Functional testing is then employed to verify that the system fulfills its specified requirements, generating precise and meaningful summaries across a spectrum of videos.

The testing process extends to edge cases, where extreme or unexpected inputs are applied to gauge how well the system handles unconventional scenarios. Performance testing is integral to assessing the efficiency of the summarization process under various conditions, encompassing videos of different lengths and sizes.

Usability testing is introduced to evaluate the user interface and overall user experience, ensuring that the system is intuitive and user-friendly. Robustness testing examines the system's resilience to errors and unexpected inputs, while scalability testing gauges its capacity to handle increased load.

The incorporation of security testing aims to identify and address potential vulnerabilities, safeguarding user data and fortifying the system against common security threats. The final step involves user acceptance testing, where end-users engage with the system in a real-world setting, providing valuable feedback for further refinement and optimization.

This comprehensive testing strategy aims to ascertain that the YouTube video summarizer function's reliably and consistently, meeting user expectations and upholding the integrity of the AI-driven summarization process. Through this systematic approach,

potential issues are identified and addressed, ensuring the project's overall success and user satisfaction.

## **TOOLS USED IN THE PROJECT**

### **4.1.1 PROGRAMMING LANGUAGE:**

- **Python:** This is what makes Python a popular high-level general-purpose programming language that has very nice syntax and a large collection of modules (standard library) in it. This approach allows for legible code as seen by its clean syntax with well-indented code blocks being made accessible even to rookies. Python programming language is dynamically typed and interpreted, allowing any variables to be assigned without any prior type declaration, and the code is progressively executed one line after another. It is well known that the language has one of the biggest standards in the computer science area covering various functionalities like input output, regular expressions, networking, etc. thereby taking out the burden of the programmers creating those from scratch. It also supports many programming styles like object-oriented, procedural, and function languages, thereby offering flexibility in implementing modules or components via class and object. The automatic garbage collection of Python's memory simplifies memory allocation and ultimately enhances development efficiency. Furthermore, unlike many other programming languages, python is independent of platforms and can be deployed on a variety of operating systems. The vast array of ready-made third-party libraries and frameworks available means that it supports pretty much everything one could think of, starting from web development up to machine learning, artificial intelligence, automation, and so on. The fact that it has a large and vibrant community, which makes it an impressive array of online resources, forums as well and tutorials also comes into support for developers. Given that Python is highly interoperable with various languages and technologies, it has become the most preferred programming language in numerous areas. Considering everything, Python's readability, flexibility, as well as its massive community make it a renowned and popularly loved language in various fields in general.

#### 4.1.2 AI LIBRARIES/Frameworks

##### 1. Time:

Python's time library offers routines for handling time-related tasks. It enables users to synchronise events in Python programmes, measure durations, and retrieve timestamps. The time library can be used for developing real-time clock applications or scheduling systems, as well as for performance monitoring, timeout implementation, rate restriction in API interactions, and other purposes. It is an adaptable tool for managing time-related tasks in a variety of fields, such as scientific computing, system management, and software development.

##### 2. Matplotlib.pyplot:

It is a feature-rich Python plotting package that lets users make a broad range of static, interactive, and publication-quality figures. For creating plots, scatter plots, bar charts, line plots, histograms, and more, it offers a high-level interface. Offering a wide range of customisation choices and compatibility for multiple plot types, annotations, and colour maps. It is frequently used in domains like data science, scientific research, engineering, finance, and academia for activities involving data exploration, analysis, and visualisation. It is a flexible tool for producing visualisations for reports, presentations, and interactive applications because it also provides smooth interface with web apps, GUI frameworks, and Jupyter notebooks.

##### 3. Transformers (Hugging Face):

The famous open-source resource, Hugging Face's Transformers library provides various options for pre-trained NLP models which include transformer-based models like BERT, GPT, and many more. We required such a library for tasks like text summarization to exploit pre-trained transformer architectures. This package made it easier for us to integrate strong language models into our project.

##### 4. YouTube- Transcript-API:

Retrievable YouTube video transcription, also known as captions, is obtained using a Python package identified as YouTube-transcript-API. It helps to retrieve text-based video transcripts via a programmatic fashion. This may be useful as a project if the audio needed from some YouTube spoken videos that is to be included in it. Having gotten the transcripts, the process of summarisation begins.

## **5. NumPy:**

Large multidimensional arrays and matrices are supported by NumPy, a core library for numerical computing in Python, which also offers a set of mathematical methods to effectively work with big arrays. Array manipulation, mathematical operations, linear algebraic routines, Fourier transforms, and random number generation are some of its primary characteristics. NumPy's speed, memory efficiency, and user-friendliness, it is extensively utilised in scientific computing, data analysis, machine learning, and engineering applications. Thanks to its array-oriented computing capabilities, users may apply algorithms for tasks like data pretreatment, signal processing, optimisation, and simulation, as well as carry out intricate numerical computations and work with sizable datasets. Furthermore, NumPy forms the basis of numerous other scientific computing and data analysis tools inside the Python ecosystem.

## **6. Seaborn:**

It is based on matplotlib, Seaborn is a Python visualisation package that offers a high-level interface for making visually appealing and educational statistical visuals. Plotting options include scatter, line, bar, histogram, and heatmap plots; statistical estimation and data aggregation are also supported natively. Among Seaborn's primary characteristics are its smooth interaction with pandas' data structures, its ability to handle data aggregation and binning automatically, and its wide range of customisation choices for plot aesthetic control. It is frequently utilised in domains including data science, economics, biology, and social sciences for statistical visualisation, presentation-quality visuals, and exploratory data analysis. Furthermore, Seaborn makes complicated visualisation jobs easier by offering default settings and user-friendly APIs for producing aesthetically pleasing plots with less coding.

## **7. Cosine\_similarity:**

The scikit-learn library contains a function called cosine\_similarity that is used to calculate the cosine similarity between pairs of samples, which are usually represented as vectors. It goes from -1 (totally unlike) to 1 (exactly similar), measuring the cosine of the angle between two vectors. Information retrieval, recommendation systems, text mining, and clustering are among the fields in which cosine similarity is applied. In many different fields, including

collaborative filtering, document classification, natural language processing, and document comparison, it is frequently utilised to compare documents, determine the degree of similarity across text documents or features, and locate related objects. Furthermore, cosine similarity is used in machine learning applications such as nearest neighbour methods for similarity-based classification or regression and clustering algorithms (e.g., k-means).

#### **8. TfidfVectorizer:**

A part of the scikit-learn library called TfidfVectorizer is used to turn a set of unprocessed text documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) characteristics. It calculates each document's TF-IDF representation, where each feature denotes a term's significance inside the document in relation to the corpus. TfidfVectorizer is used for document classification, information retrieval, and text mining. Preprocessing text data for machine learning tasks like sentiment analysis, document clustering, and text categorization is a typical usage for it. Furthermore, TfidfVectorizer makes it possible to convert textual data into a numerical format that can be used as input into machine learning algorithms. By capturing the significance of keywords and reducing the impact of frequent phrases, it enhances model performance.

#### **4.1.3 GOOGLE COLAB AS IDE -**

For Python projects, Google Colab, also known as Collaboratory, is a dynamic, cloud-based Integrated Development Environment (IDE). Colab's all-cloud operation removes the requirement for local installations, giving customers convenient access to computational resources free from hardware restrictions. One of its most notable features is that it offers free usage of GPUs and TPUs, which are very helpful for applications related to deep learning and machine learning. Because of Colab's easy integration with Google Drive, users can share and store their project notebooks with ease. The platform facilitates an interactive and iterative development process by supporting Jupyter Notebooks, which enable users to integrate code, visualizations, and narrative text in a single document. Pre-installed with popular Python libraries and frameworks such as NumPy, Pandas, and TensorFlow, Colab streamlines setup and ensures users have access to robust tools for data analysis and machine learning. With real-time collaboration capabilities, multiple users can work on the same notebook simultaneously, making it

conducive for team projects and research collaborations. Access to external data sources, interactive visualizations, Markdown support for documentation, and simplified sharing of code snippets further enhance Colab's appeal. In essence, Google Colab provides a versatile, collaborative, and accessible environment for Python development, making it particularly well-suited for projects in data science, machine learning, and research.

#### **4.1.4 VS CODE AS IDE -**

Microsoft created the well-known integrated programming environment (IDE) Visual Studio Code (VS Code), which is free and open source. With support for features like syntax highlighting, code completion, debugging, version control, and extensions, it provides a lightweight and adaptable environment for scripting in a variety of programming languages. With VS Code, developers can customise their coding experience and increase productivity thanks to its wide ecosystem of extensions, customisable layout, and user- friendly interface. Due to its widespread usage in web development, software engineering, data science, and other programming jobs, it is a very adaptable and potent tool for contemporary workflows in software development.



## 4.2 TEST CASES AND OUTCOMES:

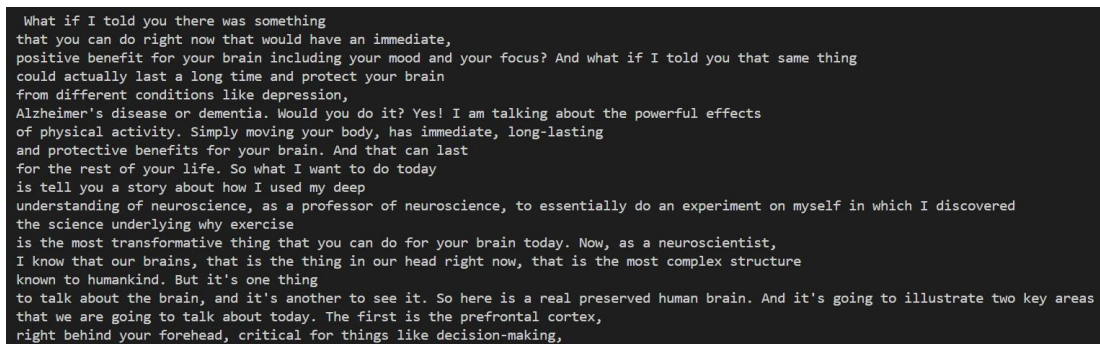
In this section test cases are used to access the system and the anticipated results are discussed below.

### Test Case 1:

**Objective:** Check if transcript is extracted.

**Expected Output:** Transcript should be extracted.

**Result:** In this test case we check if the transcript is downloaded when the link is input by the user. The test result observed indicates that YouTube transcript is being loaded when the user is giving the input.



```
What if I told you there was something
that you can do right now that would have an immediate,
positive benefit for your brain including your mood and your focus? And what if I told you that same thing
could actually last a long time and protect your brain
from different conditions like depression,
Alzheimer's disease or dementia. Would you do it? Yes! I am talking about the powerful effects
of physical activity. Simply moving your body, has immediate, long-lasting
and protective benefits for your brain. And that can last
for the rest of your life. So what I want to do today
is tell you a story about how I used my deep
understanding of neuroscience, as a professor of neuroscience, to essentially do an experiment on myself in which I discovered
the science underlying why exercise
is the most transformative thing that you can do for your brain today. Now, as a neuroscientist,
I know that our brains, that is the thing in our head right now, that is the most complex structure
known to humankind. But it's one thing
to talk about the brain, and it's another to see it. So here is a real preserved human brain. And it's going to illustrate two key areas
that we are going to talk about today. The first is the prefrontal cortex,
right behind your forehead, critical for things like decision-making,
```

Fig 4.1: Transcript is downloaded.

### Test Case 2:

**Objective:** To check the behavior of the application when incorrect URL is input.

**Expected Output:** Should throw an error.

**Result:** In this test case we check if the model could detect if the URL added by the user is valid. The test result observed indicates that if an incorrect URL is added then the app will display an error.

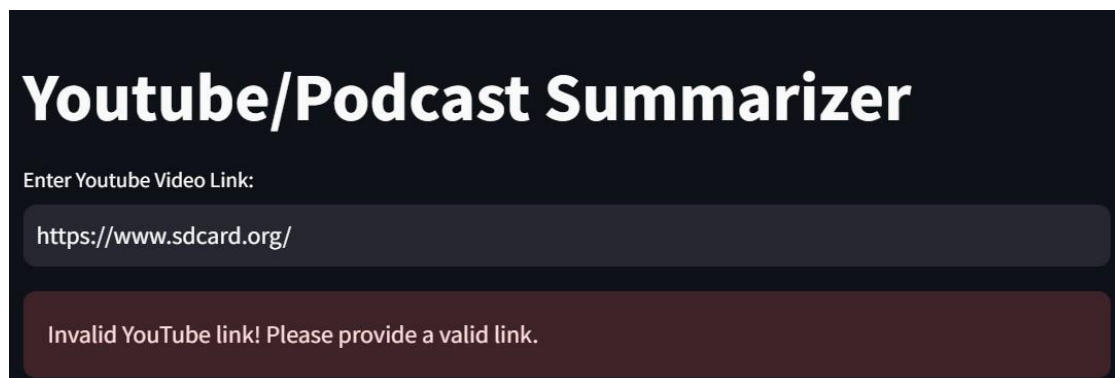


Fig 4.2: Output when incorrect URL is input.

### Test Case 3:

**Objective:** To test the behavior of the application if the user doesn't input any URL.

**Expected Output:** Error should be thrown in the app.

**Result:** In this test case we check if the app can detect if the URL was added by the user. The test result observed indicates that if no URL is added then the app will display an error.

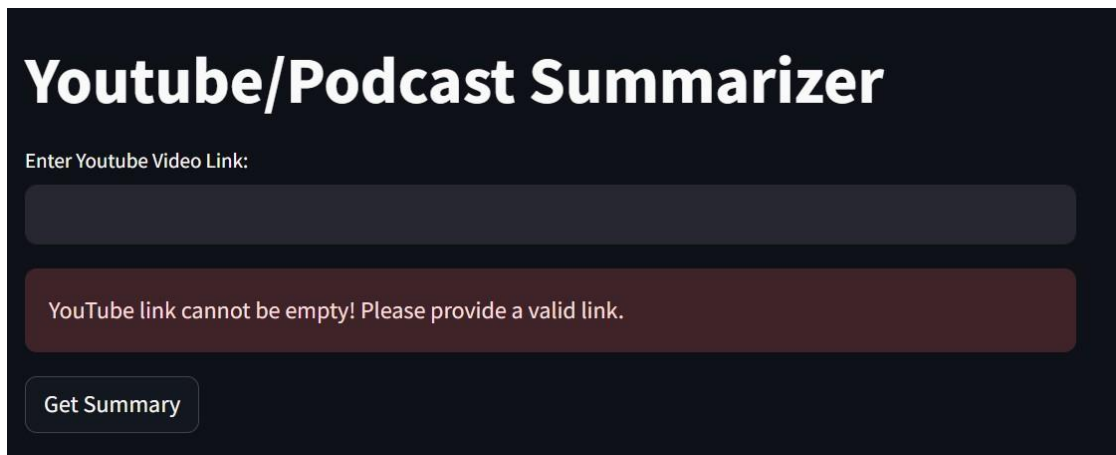


Fig 4.3: Output when no URL is input.

# CHAPTER 5

## RESULTS AND EVALUATION

### 5.1 RESULTS

After implementing summarization techniques with the help of different LLMs like GoogleGemini Pro, HuggingFace models we observed the following results.

#### A. Result of summarization performed by the Google Gemini Pro API-

In the result the similarity score, time taken for summarization and a heatmap were plotted.

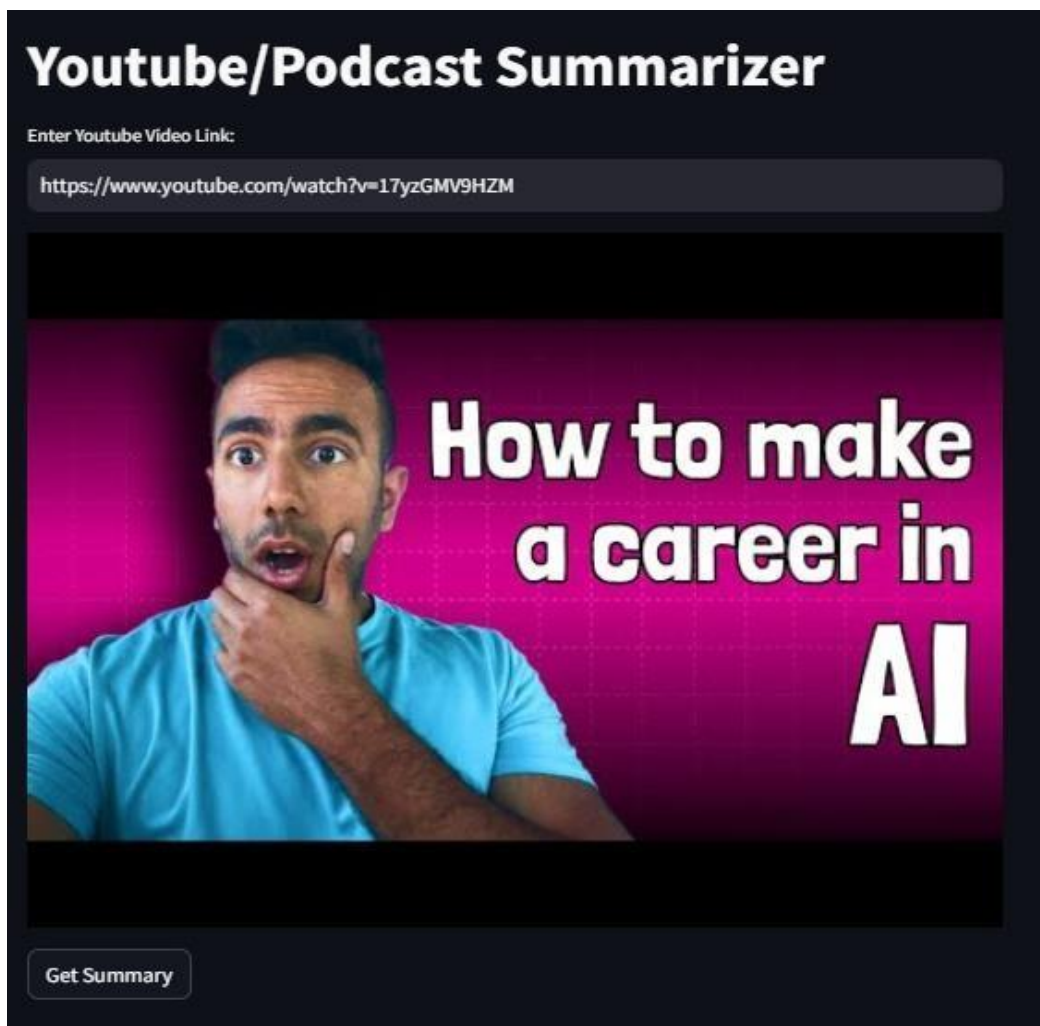


Fig 5.1: User must enter the link of video which is to be summarized.

# SUMMARY:

**Summary**

Artificial Intelligence (AI) has existed since the 1940s, but recent advancements in GPUs and data availability have made its practical application more feasible.

The speaker's AI journey began during their undergraduate studies at the Indian Institute of Technology Delhi, where they collaborated with the IBM Research Lab. They applied AI to contact center automation, utilizing techniques like SVMs.

Their PhD research focused on computer vision and multimedia networking, building immersive 3D teleconferencing systems. After their PhD, they joined a multimedia networking startup acquired by Citrix.

Google's development of LLMs in recent years has accelerated AI research and applications. The speaker believes AI is poised for widespread adoption, transforming various industries and job landscapes.

However, concerns exist about AI's potential to automate jobs, particularly in repetitive or creative fields. The speaker emphasizes the importance of staying updated with AI advancements to remain competitive in the job market.

Fig 5.2: Generated summary of the YouTube video.

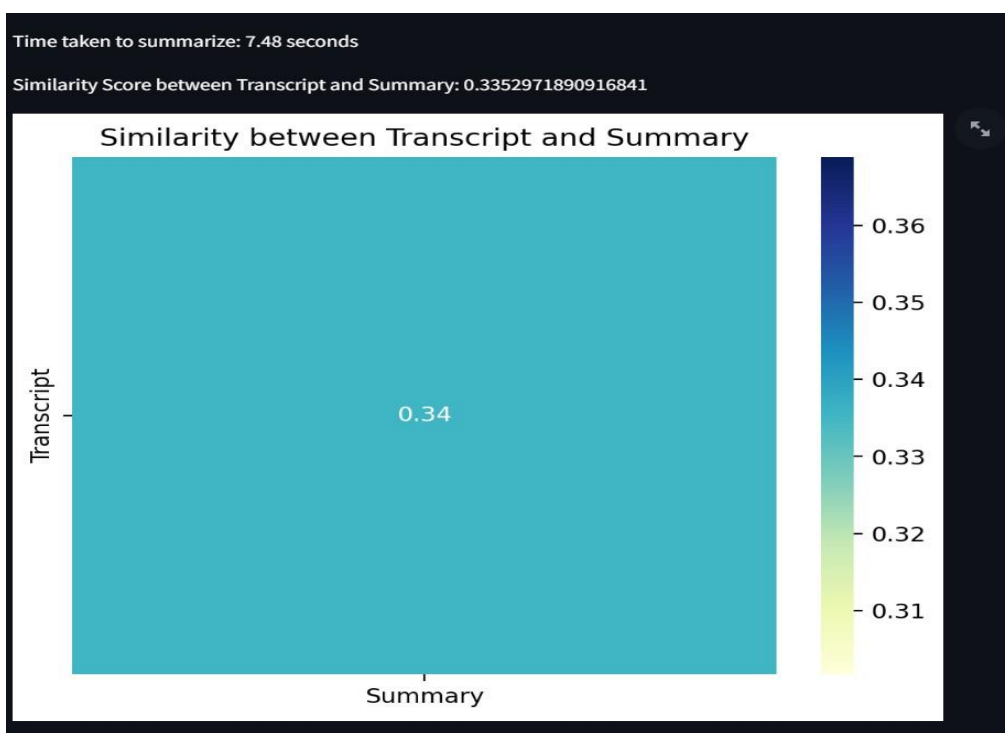


Fig 5.3: Similarity score between Transcript and generated Summary.

## B. Result of summarization performed by the HuggingFace Model- Falconsai/text summarization.

In the above implementation YouTube video podcast is downloaded and its transcript is fetched using YoutubeTranscriptApi then it is summarized using the Hugging Face Pipeline. The model used was “Falcons-ai/text\_summarization”. The same parameters were covered, that is time taken to summarize, the similarity score and heatmap.

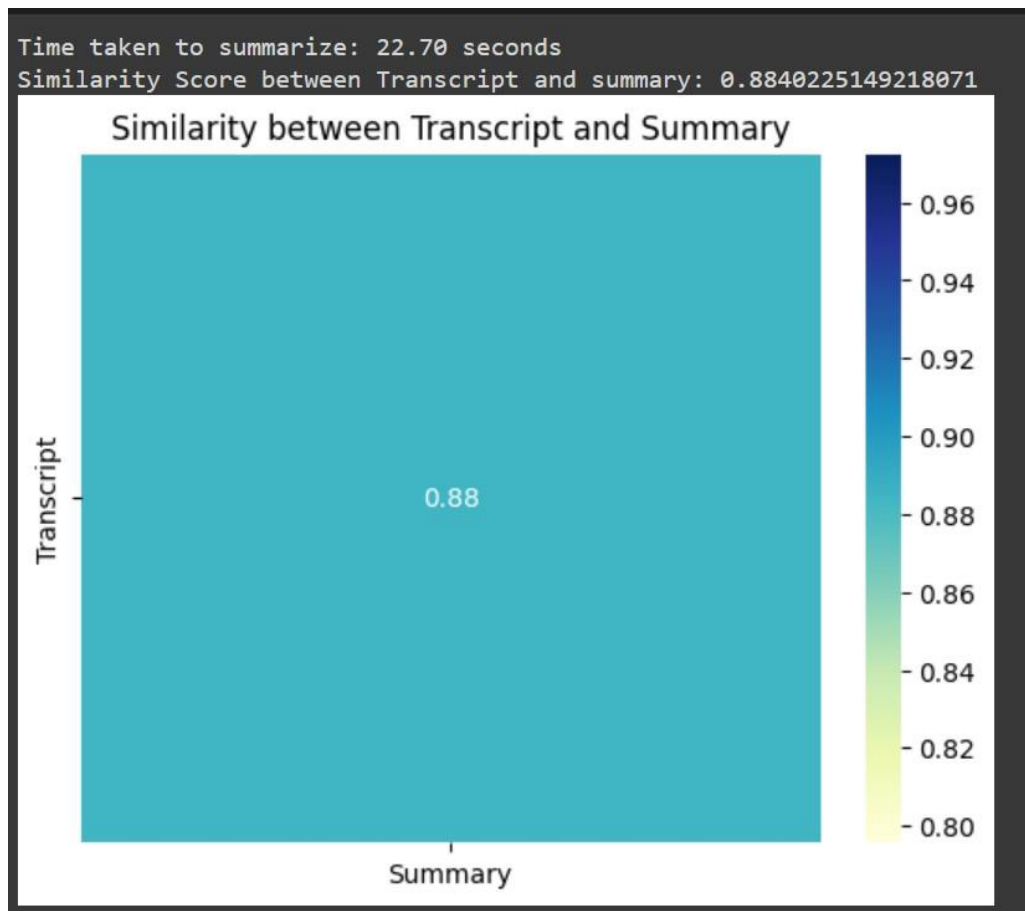


Fig 5.4: Similarity score between Transcript and generated Summary.

### C. Result of summarization performed by the HuggingFace Model -Stevhliu/my\_awesome\_billsum\_model.

A different HuggingFace model was used to summarize the YouTube video and the same parameters that were covered for the aforementioned models.

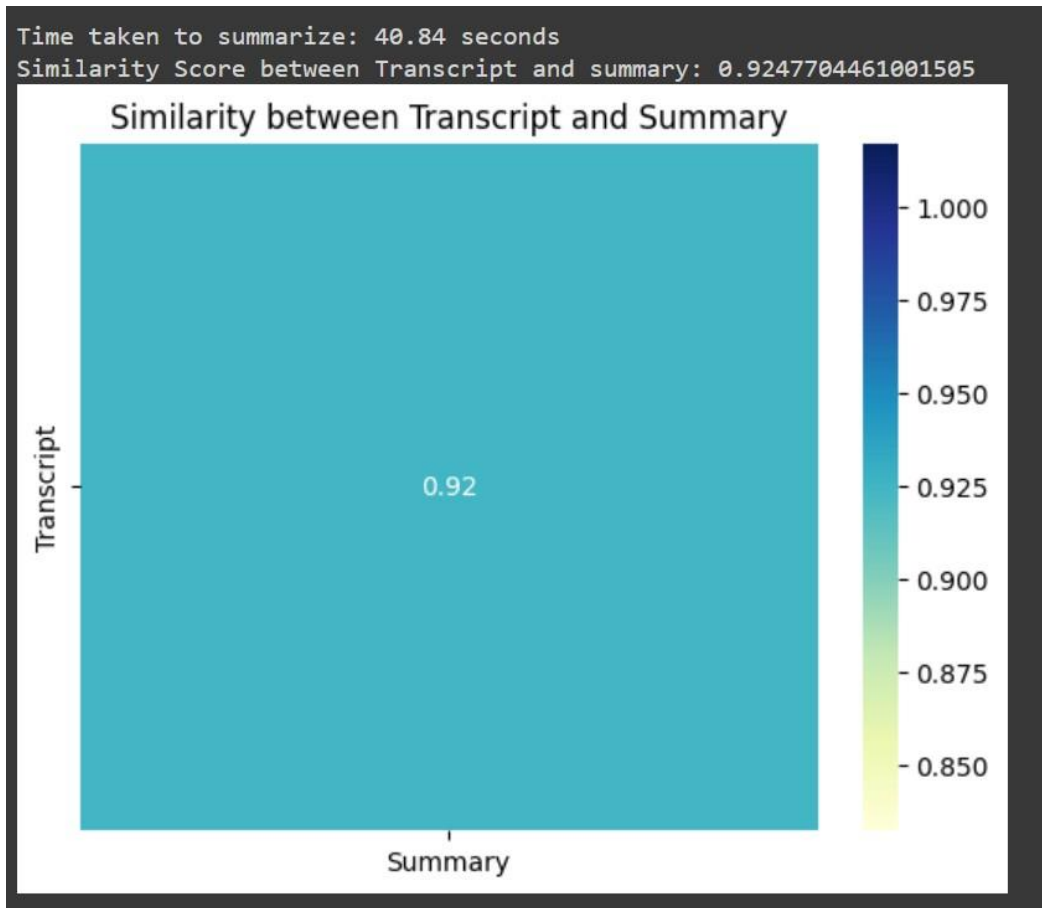


Fig 5.5: Similarity score between Transcript and generated Summary.

## D. Result of summarization of two YouTube videos simultaneously

The model used here was Facebook/bart-large-cnn and it was given the task of summarizing two videos simultaneously. One was a shorter YouTube video, and the other was a longer YouTube podcast. It was successfully able to summarize both the videos.

```
Enter the URL of the first YouTube video: https://youtu.be/55IKi7uVuyg?feature=shared
Enter the URL of the second YouTube video: https://youtu.be/55IKi7uVuyg?feature=shared
Summary for Video 1:

AI has no deep understanding at all none at all but it could do wonderful things so I've tried using it
It works reasonably well except I can't just say oh here's a picture I want and and get a nice wonderful

Summary for Video 2:

AI has no deep understanding at all none at all but it could do wonderful things so I've tried using it
It works reasonably well except I can't just say oh here's a picture I want and and get a nice wonderful
```

Fig 5.6: Result of multiple video summarization

# CHAPTER 6

## CONCLUSION AND FUTURE SCOPE

### 6.1 CONCLUSION

Developing a YouTube/Podcast Summarizer using AI. The thorough examination looks at Python code created with the OpenAI Hugging Face pipeline in mind for summarising YouTube videos. With the help of libraries like transformers and Youtube Transcript\_API. Important discoveries include the assessment of the original code's temporal efficiency and the incorporation of many tools, such as Gemini Pro API HuggingFace models namely “Stevhliu/my\_awesome\_billsum\_model” and “Falconsai/text summarization”. We observed that the most efficient results were given by the Google Gemini Pro Model in terms of time taken to summarize and similarity score. The other models were relatively less efficient and lacked summarization ability. The revised implementation includes improvements and makes use of Google Gemini Pro to provide better outcomes. With a special post-processing step, it optimizes transcript extraction, video processing, and summary for improved readability. Time efficiency measures and fine-tuning parameters are used, which add to a thorough analysis of the code's performance.

To sum up, both methods effectively automate the summary of YouTube videos, showcasing the efficient integration of models and libraries. The metric of time efficiency offers significant insights that direct continuous improvement endeavors. It is encouraged to get in touch with developers and maintainers with questions or for help.

#### **Limitations**

In our project, we leverage fixed summarization models such as Google Gemini Pro and HuggingFace models to distill information from video content. While these models generally perform well, their effectiveness hinges on the nature of the video content itself. To optimize summarization outcomes, it's imperative to explore various forms of summarization and select the most suitable method for the specific application at hand. This entails considering factors like the complexity and diversity of the content. Our



project employs a chunking approach to process video transcripts, dividing them into smaller parts or phrases for summarization. However, this method runs the risk of producing fragmented summaries, especially when crucial information spans across multiple chunks. Fine-tuning parameters such as chunk size and overlap can enhance coherence, but it's essential to maintain a holistic perspective to ensure no critical information is lost in the process.

Summarization parameters such as `max_length`, `min_length`, `length_penalty`, `num_beams`, and `early_stopping` play a pivotal role in shaping the quality and relevance of the produced summaries. These values need to be carefully adjusted according to the unique characteristics of the video content. Experimenting with different parameter configurations can help optimize summarization outcomes and meet specific requirements effectively.

An underlying assumption of our project is that the video content is in English. However, this assumption may prove misleading if the video contains content in other languages not supported by the summarization model. To address this limitation, integrating a language detector and utilizing language-specific models could enhance the project's language processing capabilities and ensure accurate summarization across diverse linguistic contexts.

The project relies on external APIs like the YouTube API and Hugging Face Model Hub for data retrieval and summarization. However, maintaining proper authentication and monitoring changes in these APIs poses a challenge. It's crucial to stay vigilant and adapt to any modifications that may affect the validity or functionality of the project's codebase. When dealing with lengthy videos, particularly those where only certain segments are relevant, relying solely on Hugging Face models to download and transcribe the entire video may not be the most efficient approach. Instead, alternative methods such as extracting transcripts directly from the video without downloading it have been implemented to streamline the summarization process and conserve resources.

## 6.2 FUTURE SCOPE

Several improvements are planned for further versions to improve the summarizer's usability and flexibility. Firstly, users could have the freedom to select from a wide range of Hugging Face summary models, customising their choices to fit their tastes or particular use cases. This modification could greatly increase the summarizer's adaptability. Secondly, although the English content in the present edition works well, there are plans to provide models optimised for other languages. Furthermore, investigating language identification algorithms may increase the tool's suitability in various linguistic circumstances. Among the most important things to do in the future is to strengthen error management systems. The summarizer may be made more dependable and robust by improving the error-handling procedures, particularly when video URLs stop working or accessibility problems arise with APIs. The model also gives different results repeatedly as it requires more training which won't be sufficient just by instruction tuning and prompt tuning. For more efficient results fine tuning is required which requires high GPU usage. Ultimately, expanding adaptive input management skills is essential to support a wide range of applications. It would be more flexible and useful if the script could handle different podcast or video URLs on the fly. This would allow it to serve a larger range of users with different summarising requirements. These upcoming improvements should improve the summarizer's usability and functionality, guaranteeing that it will remain relevant and useful for jobs involving the summation of text and video.

## REFERENCES

- [1] O. Topsakal and T. C. Akinci, "Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast", ICAENS, vol. 1, no. 1, pp. 1050–1056, Jul. 2023.
- [2] Ms Bhandare, Aishwarya Chigare, Utkarsha Patil, Shweta Sangle. "YouTube Transcript Summarizer." International Research Journal of Modernization in Engineering, Technology and Science, vol. 04, no. 03, March 2022.
- [3] Reshma Shaik, Saloni Bargat, and Prof. Shilpa Ghode, "Article and YouTube Transcript Summarizer Using Spacy and NLTK Module", ssgmjse, vol. 1, no. 1, pp. 126–131, Jun. 2023.
- [4] R. A. Albeer, H. F. Al-Shahad, H. J. Aleqabie, and N. D. Al-Shakarchy, "Automatic summarization of YouTube video transcription text using term frequency-inverse document frequency," Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 3, pp1512-1519, Jun. 2022.
- [5] H. Gupta and M. Patel, "Method Of Text Summarization Using Lsa And Sentence Based Topic Modelling With Bert," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 2021, pp. 511-517,
- [6] Y. Liu, A. R. Fabbri, P. Liu, D. Radev, and A. Cohan, "On Learning to Summarize with Large Language Models as References," arXiv (Cornell University), doi: 10.48550/arxiv.2305.14239.
- [7] S. Devi, R. Nadar, T. Nichat, and A. Lucas, "Abstractive summarizer for YouTube videos," in Advances in computer science research, 2023, pp. 431–438.
- [8] K. Song, C. Li, X. Wang, D. Yu, and F. Liu, "Towards abstractive grounded summarization of podcast transcripts," Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Jan. 2022.
- [9] Awasthi, K. Gupta, P. S. Bhogal, S. S. Anand and P. K. Soni, "Natural Language Processing (NLP) based Text Summarization - A Survey," 2021 6th International

Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2021, pp. 1310-1317, doi: 10.1109/ICICT50816.2021.9358703.

- [10] S. Gehrmann, Y. Deng, and A. Rush, "Bottom-up abstractive summarization," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)\*, Brussels, Belgium, 2018, pp. 4098–4109.
- [11] T. Falke, L. F. R. Ribeiro, P. A. Utama, I. Dagan, and I. Gurevych, "Ranking generated summaries by correctness: C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in \*Text Summarization Branches Out\*, Barcelona, Spain, 2004, pp. 74–81.
- [12] P. Manakul and M. Gales, "CUED\_SPEECH AT TREC 2020 PODCAST SUMMARISATION TRACK." Accessed: May 16, 2024.
- [13] C. Zhu, Y. Liu, J. Mei, and M. Zeng, "MediaSum: A Large-scale Media Interview Dataset for Dialogue Summarization," arXiv (Cornell University), Jan. 2021.
- [14] K. Merchant and Y. Pande, "NLP Based Latent Semantic Analysis for Legal Text Summarization," 2018 International Conference on Advances in Computing, Communications, and Informatics (ICACCI), Sep. 2018,
- [15] R. Khan, Y. Qian, and S. Naeem, "Information Engineering and Electronic Business," Information Engineering and Electronic Business, vol. 3, pp. 33–44, 2019.
- [16] P. Verma and A. Verma, "Accountability of NLP Tools in Text Summarization for Indian Languages," Journal of scientific research, vol. 64, no. 01, pp. 258–263, 2020.
- [17] Tippaya Thinsungnoena et al 2015, "The Clustering Validity with Silhouette and Sum of Squared Errors". The 3rd International Conference on Industrial Application Engineering (ICIAE2015).
- [18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North*, vol. 1, 2019.
- [19] W. Fu and P. O. Perry, "Estimating the Number of Clusters Using Cross-Validation,"

Journal of Computational and Graphical Statistics, vol. 29, no. 1, pp. 162–173, Sep. 2019.

- [20] Y. Liu, A. R. Fabbri, P. Liu, D. Radev, and A. Cohan, “On Learning to Summarize with Large Language Models as References,” arXiv (Cornell University), doi: 10.48550/arxiv.2305.14239.
- [21] D. Trautmann, A. Petrova, and F. Schilder, “Legal prompt engineering for multilingual legal judgement prediction,” CoRR, vol. abs/2212.02199, 2022
- [22] W. X. Zhao et al., “A Survey of Large Language Models,” arXiv:2303.18223 [cs], Mar. 2023.
- [23] Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” 2018.
- [24] S. S. Naik and M. N. Gaonkar, "Extractive text summarization by feature-based sentence extraction using rule-based concept," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2017.
- [25] A. Blair-Stanek, N. Holzenberger, and B. V. Durme, “Can GPT-3 perform statutory reasoning?” CoRR, vol. abs/2302.06100, 2023.

# APPENDIX

G35

---

ORIGINALITY REPORT

---

<b>10</b> %	<b>8</b> %	<b>7</b> %	<b>6</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

---

PRIMARY SOURCES

---

<b>1</b>	<b>arxiv.org</b> Internet Source	<b>1</b> %
<b>2</b>	<b>aclanthology.org</b> Internet Source	<b>1</b> %
<b>3</b>	<b>www.arxiv-vanity.com</b> Internet Source	<b>&lt;1</b> %
<b>4</b>	<b>mecs-press.org</b> Internet Source	<b>&lt;1</b> %
<b>5</b>	<b>as-proceeding.com</b> Internet Source	<b>&lt;1</b> %
<b>6</b>	<b>ijsrcseit.com</b> Internet Source	<b>&lt;1</b> %
<b>7</b>	<b>ijeecs.iaescore.com</b> Internet Source	<b>&lt;1</b> %
<b>8</b>	<b>www.diva-portal.org</b> Internet Source	<b>&lt;1</b> %

---


**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: 18/05/24  
 Type of Document (Tick):  **PhD Thesis**  **M.Tech/M.Sc. Dissertation**  **B.Tech./B.Sc./BBA/Other**  
 Name: ANSH CHOUDHARY MITUSHI KOHLI Department: CSE Enrolment No 2013021201174  
 Contact No. 7018432893 E-mail: ansh69@gmail.com  
 Name of the Supervisor: Dr. Deepak Gupta  
 Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): YOUTUBE | PODCAST SUMMARIZER USING AI

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 60
- Total No. of Preliminary pages = 10
- Total No. of pages accommodate bibliography/references = 03

*Mitushi*  


(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at 10% (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

*Dhyani* 18/05/24  
 (Signature of Guide/Supervisor)

*Vinod*  
 (Signature of MOD)

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Abstract & Chapters Details	
Report Generated on	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
			Character Counts	
		Submission ID	Page counts	
			File Size	

Checked by  
 Name & Signature

Librarian

Please send your complete Thesis/Report in (PDF) & DOC (Word File) through your Supervisor/Guide at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)