INDEX

|  |  |  |
| --- | --- | --- |
| Serial no. | Chapter | Page no |
| 1 | Abstract | 5 |
| 2 | YouTube summarizer & Purpose | 6 |
| 3 |  | 7-8 |
| 4 |  | 9-10 |
| 5 | Future Scope of Youtube Summarizer | 11-12 |
| 6 | Python Libraries used | 12 |
| 7 | System design | 13 |
| 7 | Source Code app.py | 14 |
| 8 | Source Code main.py | 15-18 |
| 9 | Source Code requirements.txt & UI | 19-22 |
| 10 | Result: OUTPUTS | 23 |
| 11 | Conclusion | 24 |
| 12 | References | 25 |

**Abstract**

Enormous number of video recordings are being created and shared on the Internet throughout the day. In today's daily lifestyle, it has become really difficult to spend time watching such videos which may have a longer duration than expected and sometimes our efforts may become futile if we couldn't find relevant information out of it as most of the videos are been uploading to grab the attention of the viewers by tricking or misleading the viewers by thumbnails, advertisements, etc.As the number of users of YouTube are increasing rapidly from year to year, this may directly impact the number of videos to be created. In greed for the number of views, the chance that the creators of the videos may give wrong information on the original content of the video is probably very high. This may waste the valuable time and resources of the user. To further improve the user interaction with the summarizer Chrome extension is used for user-friendly interaction which consists of a summarize button. On clicking this button, the Chrome extension displays the summarized text of the current YouTube video running on the Google Chrome web browser.

The paper aims in giving all the required details to understand the working of our YouTube Summariser, Problem it is solving, use cases, system design, libraries used, future-scope and references.

## YOUTUBE SUMMARISER

A YouTube summarizer is a tool or application that aims to provide a concise summary of a YouTube video's content. It uses various techniques and algorithms to analyze the video's audio, subtitles, or transcripts and extract key information to generate a condensed summary.

The purpose of a YouTube summarizer is to save time for viewers who may not have the time or patience to watch the entire video. It provides a quick overview of the video's main points, allowing users to understand the content without having to watch the entire duration.

YouTube summarizers typically employ natural language processing (NLP) techniques to process the video's subtitles or transcripts. They may utilize methods like text summarization, keyword extraction, or sentiment analysis to identify the most important information and generate a concise summary.

The summarization process involves analyzing the video's content, identifying key sentences or phrases, and condensing them into a shorter summary. Depending on the complexity of the summarization algorithm, the output can range from a few sentences to a paragraph or more.

## PURPOSE

The purpose of a YouTube summarizer is to provide a condensed summary of a YouTube video's content in order to:

1. Save Time: YouTube videos can range from a few minutes to several hours in duration. Not everyone has the time or patience to watch lengthy videos, especially when they are seeking specific information or just want a quick overview. A YouTube summarizer helps users save time by providing a concise summary that captures the key points and main ideas of the video.
2. Provide a Quick Overview: Summaries offer a snapshot of the video's content, allowing users to quickly grasp the main points without having to watch the entire video. This is particularly useful when users want to get a sense of what a video is about or if it covers a specific topic of interest.
3. Aid Decision-making: Summaries help users make informed decisions about whether a video is worth watching in its entirety. By providing a summary, users can evaluate the relevance, usefulness, or entertainment value of the video before investing their time.
4. Enhance Accessibility: Summaries make video content more accessible for individuals who may have limitations such as hearing impairments or language barriers. Text-based summaries can be easily translated or used with screen readers, allowing a wider range of users to access and benefit from the content.
5. Facilitate Content Discovery: YouTube summarizers can help users discover new videos or channels by providing a brief overview of the content. Users can quickly scan through multiple summaries and decide which videos align with their interests, expanding their content consumption and exploration.
6. It's important to note that YouTube summarizers are not intended to replace watching the full video in all situations. They serve as a tool to provide a condensed version of the content, but may not capture all the nuances, details, or visual elements present in the original video. The purpose of a YouTube summarizer is to provide a convenient and time-saving way to access the key information from YouTube videos.

### Future Scope of Youtube Summarizer

**Advanced Summarization Techniques**: Explore more sophisticated algorithms for better summarization accuracy

**Integration with Other Platforms**: Integrate Youtube Summarizer with other online learning platforms or social media sites

**Multilingual Support**: Extend language support to include various languages for global accessibility

**User Feedback and Updates**: Continuously gather user feedback to improve and enhance the application

**Expansion to Other Media Types**: Explore summarization of audio, podcasts, or other media formats

**WORD TUNE and RECCAP are tools used for summarizing youtube videos**



## Python Libraries used in our project

##### STREAMLIT

## The Streamlit library is used to create the web application and user interface. It simplifies the process of building interactive web applications in Python.

## SUMY

## sumy is a Python library that provides various algorithms and methods for text summarization. It offers different approaches such as LSA (Latent Semantic Analysis), LexRank, and Luhn's algorithm, among others. These algorithms can be used to summarize text documents or in this case, the transcript of a YouTube video.

##### youtube-transcript-api

## The youtube-transcript-api is a Python library that provides an interface to access and retrieve the transcripts or subtitles of YouTube videos. It allows you to fetch the transcript for a given YouTube video ID

#### Transformers

#### The transformers library is used for text summarization. It provides pre-trained models and methods for performing text summarization tasks using techniques like BERT (Bidirectional Encoder Representations from Transformers).

#### In the context of your project, these libraries are used to fetch the transcript or subtitles of a YouTube video (youtube-transcript-api), perform text summarization on the retrieved transcript (sumy).

#### NLP (NATURAL LANGUAGE PROCESSING)

#### Natural language processing strives to build machines that understand and respond to text or voice data—and respond with text or speech of their own—in much the same way humans do.Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch ofartificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

## SYSTEM

In this project we get transcripts/subtitles for a given YouTube video Id using a Python API, perform text-summarization on obtained transcripts using HuggingFace transformers, build a streamlit webapp to expose the summarization service to the client and develop a chrome extension which will utilize the backend API to display summarized text to the user.

## BACK END

## *We have created a back-end application directory containing files written in Python.*

## GET TRANSCRIPTS

## *In this module, we are going to utilize a python API(youtube-transcript-api) which allows you to get the transcripts/subtitles for a given YouTube video*

## PERFORM TEXT SUMMARIZATION

## *Text summarization is the task of shortening longer text into a precise summary that preserves key information content and overall meaning.* *The transformers library is used for text summarization.*

## USER INTERFACE

*User interface is needed to ensure that the user can interact with the system at ease.*

*The Streamlit library is used to create the web application and user interface.*

**nltk (Natural Language Toolkit)**

## *The NLTK library is used for natural language processing tasks, such as tokenization and stemming, to preprocess the text data.*

## Source Code

We have explained some crucial information with our code and divided it into three parts. There are other files too such as Venv providing environment and pycache file.

1. app.py
2. main.py
3. requirements.txt

**The language used in Project is Python and done on PyCharm Community**

## Part1 : app.py

## import streamlit as st

## from main import summarize\_youtube\_video

## def main():

## st.title("YouTube Video Summary")

## # Get the YouTube link from the user

## youtube\_link = st.text\_input("Enter the YouTube video link")

## # Summarize the YouTube video on button click

## if st.button("Summarize"):

## if youtube\_link:

## # Call your summarization function from app.py

## summary = summarize\_youtube\_video(youtube\_link)

## # Display the summary1 in a styled box

## st.success("Summary :")

## st.info(summary)

## else:

## st.warning("Please enter a YouTube video link.")

## if \_\_name\_\_ == "\_\_main\_\_":

## Main()

The code you provided represents the main() function of a Streamlit application. Let's go through the code and understand its functionality:

1. The st.title("YouTube Video Summary") line sets the title of the Streamlit application to "YouTube Video Summary".
2. The st.text\_input("Enter the YouTube video link") line creates a text input box where the user can enter the YouTube video link. The input value is stored in the youtube\_link variable.
3. The if st.button("Summarize"): line checks if the "Summarize" button is clicked by the user.
4. Inside the if statement, there is a condition if youtube\_link: that checks if the youtube\_link variable is not empty.
5. If the youtube\_link is not empty, the summarize\_youtube\_video(youtube\_link) function is called, passing the youtube\_link as an argument. The returned summary is stored in the summary variable.
6. The st.success("Summary :") line displays a success message above the summary box.
7. The st.info(summary) line displays the summary text in an information box.
8. If the youtube\_link is empty, a warning message is displayed using st.warning("Please enter a YouTube video link.").
9. Finally, the if \_\_name\_\_ == "\_\_main\_\_": block ensures that the main() function is executed only when the script is directly run, and not when imported as a module.

To run the Streamlit application, you can execute the main() function. The application will display an input box for the YouTube video link. After entering the link and clicking the "Summarize" button, the application will call the summarize\_youtube\_video() function to generate a summary and display it in the application.

## Part 2: main.py

import youtube\_transcript\_api

## from transformers import pipeline

## summarization = pipeline('summarization')

## def get\_video\_transcript(video\_id):

## transcript = youtube\_transcript\_api.YouTubeTranscriptApi.get\_transcript(video\_id)

## text = ' '.join([t['text'] for t in transcript])

## return text

## def summarize\_youtube\_video(youtube\_link):

## video\_id = youtube\_link.split("v=")[1]

## try:

## full\_transcript = get\_video\_transcript(video\_id)

## num\_iters = int(len(full\_transcript) / 1000)

## summarized\_text = []

## for i in range(0, num\_iters + 1):

## start = 0

## start = i \* 1000

## end = (i + 1) \* 1000

## # print("input text \n" + full\_transcript[start:end])

## out = summarization(full\_transcript[start:end], min\_length=5, max\_length=20)

## out = out[0]

## out = out['summary\_text']

## # print("Summarized text\n"+out)

## summarized\_text.append(out)

## # Adjust the number of sentences for the summary as needed

## return summarized\_text

## except :

## return "Error: Failed to summarize the YouTube video."

## get\_video\_transcript function

Let's break down the code step by step:

1. The function takes a parameter video\_id, which represents the unique identifier of the YouTube video for which you want to retrieve the transcript.
2. Inside the function, youtube\_transcript\_api.YouTubeTranscriptApi.get\_transcript(video\_id) is called. This line of code uses the get\_transcript() function from the youtube\_transcript\_api library to fetch the transcript of the specified YouTube video.
3. The returned transcript is stored in the transcript variable.
4. The next line of code, [t['text'] for t in transcript], iterates over each element in the transcript list and extracts the text from each element. The result is a list of text strings, where each string represents a line of the transcript.
5. The join() function is then called on the list of text strings, using ' '.join(...). This concatenates all the text strings in the list, separating them with a space. The result is a single string containing the entire transcript text.
6. Finally, the function returns the concatenated transcript text as the output.

## summarize\_youtube\_video function

Let's break down the code step by step:

1. The function starts by extracting the video ID from the YouTube link. It uses the split() method to split the link based on the "v=" parameter, and then takes the second element of the resulting list (index 1). The video ID is stored in the video\_id variable.
2. Inside a try-except block, the function attempts to retrieve the full transcript of the YouTube video by calling the get\_video\_transcript() function (assuming this function is defined elsewhere in your code). The full transcript is stored in the full\_transcript variable.
3. The next few lines calculate the number of iterations needed to process the transcript in segments of 1000 characters each. This is done to prevent exceeding the maximum input length for the summarization model.
4. A loop is then used to iterate through the transcript segments. Each segment is passed to the summarization() function, which generates a summary using the specified parameters. The resulting summary is extracted from the output and stored in the out variable.
5. The summarized text segment is appended to the summarized\_text list.
6. After processing all segments, the function returns the summarized\_text list, which contains the summarized text segments.
7. If an error occurs during the process (such as failure to retrieve the transcript or summarize the video), the function returns an error message.

**Requirements.txt**

streamlit

sumy

youtube\_transcript\_api

transformers

# USER INTERFACE

# SOME OUTPUTS

# Output for a 5 min video

# 

# Output for a 8min video

# 

# Output for a 13 min video

# 

# CONCLUSION

This project has proposed a YouTube Transcript summarizer. The system takes the input YouTube video from the Streamlit app when the user clicks on the summarize button on the web page. The transcripts of that video with the help of python API is accessed. The accessed transcripts are then summarized with the transformers package. Then the summarized text is shown to the user web page. In ­summary, our YouTube Summarizer project demonstrates the power of data science and technology in simplifying the way we interact with and consume YouTube content. We believe that our tool has the potential to enhance the YouTube viewing experience for users worldwide, saving time, enhancing accessibility, and empowering informed decision-making.

# REFERENCES

I. 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),  
2021, pp. 1310-1317, doi: 10.1109/ICICT50816.2021.9358703.  
[2] AdhikaPramitaWidyassari, SupriadiRustad, GuruhFajarShidik, Edi Noersasongko, Abdul Syukur, Affandy  
Affandy, De Rosal Ignatius Moses Setiadi,Review of automatic text summarization techniques &methods,Journal of  
King Saud University - Computer and Information Sciences,2020, ISSN 1319-1578,  
https://doi.org/10.1016/j.jksuci.2020.05.006.  
[3] P. R. Dedhia, H. P. Pachgade, A. P. Malani, N. Raul and M. Naik, "Study on Abstractive Text Summarization  
Techniques," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-  
ETITE), 2020, pp. 1-8, doi: 10.1109/ic-ETITE47903.2020.087.  
[4] A. Dilawari and M. U. G. Khan, "ASoVS: Abstractive Summarization of Video Sequences," in IEEE Access, vol. 7,  
pp. 29253-29263, 2019, doi: 10.1109/ACCESS.2019.2902507.  
[5] https://huggingface.co/transformers/installation.html  
[6] https://atmamani.github.io/blog/building-restful-apis-with-flask-in-python/  
[7]https://pypi.org/project/youtube-transcript-api/  
[8] https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask  
[9]https://medium.com/swlh/parsing-rest-api-payload-and-query-parameters-with-flask-better-than-marshmallow-aa79c889e3ca  
[10] https://betterprogramming.pub/the-ultimate-guide-to-building-a-chrome-extension-4c01834c63ec  
[11] https://developer.chrome.com/docs/extensions/mv2/