IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

YOUTUBE TRANSCRIPT SUMMARIZER

Gousiya Begum[1], N. Musrat Sultana[2], Dharma Ashritha[3]

[1]Assistant Professor, Department of CSE, Mahatma Gandhi Institute of Technology
[2]Assistant Professor, Department of CSE, Mahatma Gandhi Institute of Technology
[3]B.Tech Student, Department of CSE, Mahatma Gandhi Institute of Technology

***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.

***Keywords*—** Text Summarizer, Chrome Extension, HuggingFace transformers, WebAPI

I. INTRODUCTION

Huge number of video recordings are being created and shared on the Internet throughout the day. The number of YouTube users in 2020 was approximately 2.3 billion, and has been rapidly increasing every year. For every single minute, watch time of 300 hours of videos are uploaded to YouTube. Almost one-third of the YouTube viewers in India access videos on their mobiles and spend over 48 hours a month on the website according to the Google study.

It has become really difficult to spend time watching such videos which may have longer duration than expected and sometimes our efforts may become futile if we couldn't find relevant information out of it which we are in search of. It is frustrating and time consuming to search for the videos that contain the information we are actually looking for. For instance, there are many videos available online in which the speaker talks for a long time on a given topic, but it is hard to find the content the speaker wants to convey to the audience unless we watch the entire video.

Python has various packages which are very helpful. Currently accessing the YouTube content has been made easier through the API in the python library like the transcripts of the videos etc. By taking this advantage we directly access the transcripts of the video and summarize to view them to the user.This can be implemented by using Hugging face transformer which is one of the text summarization techniques.

To increase the user interface to the functionality part Chrome extensions can be used.This Chrome extensions contains a summarize button, which on click displays the summarized text of the current YouTube video on the Google Chrome web browser. This summarized text is generated by the hugging face transformer package.

The YouTube videos are usually summarized through manual descriptions and thumbnails. YouTube is the second most visited website worldwide. The range of videos on YouTube includes short films, music videos, feature films, documentaries, audio recordings, corporate sponsored movie trailers, live streams, vlogs, and many other contents from popular YouTubers. YouTube users watch more than one billion hours of video every day.

This project proposes the usage of a transformer package for summarizing the transcripts of the video, thereby providing a meaningful and germane summary of the video. T5 is an encoder-decoder model which is pre-trained on a set of unsupervised and supervised tasks and for which each task is converted into a text-to-text format. Our main concern is to summarize the data, so a pre-trained summarization technique is used.

II. RELATED WORK

In 2021, "Natural Language Processing (NLP) based Text Summarization - A Survey" was published by Ishitva Awasthi, Kuntal Gupta, Prajbot Singh Bhojal, Anand, Piyush kumar. The techniques used are Extractive and abstract methods for summarizing texts. The advantages are: Based on linguistic and statistical characteristics, the implications of sentences are calculated. The disadvantage is each type of summarization technique is useful in different situations. One cannot say which technique is more promising [1].

"Review of automatic text summarization techniques & methods" is developed by AdhikaPramita, SupriadiRustad, Abdul Shukur, Affandy. It was published in 2020. They have used Text summarization, Systematic review techniques. The drawback is that the Fuzzy based approach is weak in semantic problems. Many loopholes have to be improved in extractive methods [2].

"Study on Abstractive Text Summarization Techniques" is developed by Parth Rajesh Dedhia, Hardik Pradeep, Meghana Naik. It was published in 2020. They have used Seq2Seq, Encoder-Decoder, and Pointer Mechanism. The drawback is that the current model does not work if multiple documents are passed to the model[3].

"Abstractive Summarization of video sequences" is developed by AniqaDilawari, Muhammad Usman Ghani Khan. They have used multi-line video description, RCNN deep neural network model. The drawback is It focuses only on the conciseness of the summary. Memory efficiency and time constraints are not under consideration[4].

III. SYSTEM DESIGN

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 Flask backend REST API 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.

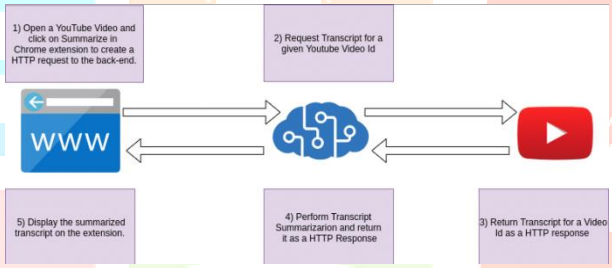

Figure 1 System Architecture of YouTube Transcript Summarizer

According to Figure 1, initially we open a YouTube video and click the button summarize in the chrome extension. This will create a HTTP request to the back-end. Then the request to access the transcripts will be made with YouTube video ID which was taken from the URL. The response will be a transcript of that video in json format. After getting the transcripts in the text format the system performs Transcript Summarization. Finally, it displays the summarized transcript on the extension.

3.1 BACK END

APIs changed the way we build applications, there are countless examples of APIs in the world, and many ways to structure or set up our APIs [5]. We create a back-end application directory containing files named as app.py. We initialize app.py with basic Flask RESTful BoilerPlate. We then create a virtual environment to isolate the location where everything resides. By activating the newly created virtual environment we install the dependencies like Flask, youtube_transcript_api, transformers using pip [6].

3.2 GET TRANSCRIPTS

In this module, we are going to utilize a python API which allows you to get the transcripts/subtitles for a given YouTube video. It also works for automatically generated subtitles, supports translating subtitles, and does not require a headless browser like other Selenium-based solutions do.In app.py, we create a function which will accept YouTube video id as an input parameter and return parsed full transcript as output. As we get the transcript in the json format with text, start and duration attributes we only need text so we parse the data from the response to return the transcript in whole string format [7].

3.3 PERFORM TEXT SUMMARIZATION

Text summarization is the task of shortening longer text into a precise summary that preserves key information content and overall meaning.

Thereare two different approaches that are widely used for text summarization:**Extractive Summarization**: In this the model identifies the important sentences and phrases from the original text

and only outputs the necessary part.

Abstractive Summarization: The model produces a completely different text that is shorter than the original, it generates new sentences in a new form. In this project, we will use transformers for this approach.

In this system, we will use the HuggingFace transformers library in Python to perform abstractive text summarization on the transcript obtained from the previous step. In app.py, we create a function which will accept YouTube transcript as an input parameter and return summarized transcript as output. Then instantiate a tokenizer and a model from the checkpoint name. Summarization is done with the help of encoder-decoder models, such as Bart or T5. Then we define the transcript that should be summarized and add the T5 specific prefix "summarize:". Finally use the PreTrainedModel.generate() method to generate the summary.

3.4 REST API ENDPOINT

The following step is to define the resources that will be used in implementation by this backend service. This is an extremely simple application, we only have a single endpoint, so our only resource will be the summarized text [8].

- In app.py, We create a Flask API Route with GET HTTP Request method with a URI http://[hostname]/api/summarize?youtube_url=#{url}. Then we extract the YouTube video id from the YouTube URL which is obtained from the query params. Then generate the summarized transcript by executing the transcript generation function following the execution of the transcript summarizer function. Finally, we return the summarized transcript with HTTP Status OK and handle HTTP exceptions if applicable [9].

3.5 CHROME EXTENSION

Chrome Extensions are the software programs that customize and enhance the browsing experience. They enable users to enhance Chrome functionality experience and behavior to individual preferences. They are built on the web using web technologies such as HTML, CSS and JavaScript. In this step, we create a chrome extension application directory containing essential files required as mentioned below [10].

We need to create a manifest.json file for loading our extension in the browser. Go to the website chrome://extensions and make sure developer mode is turned on from the top right-hand corner. Then select on Load unpacked and select the folder containing the manifest file that we just created. This creates our extension. We'll need to reload the extension every time we make a change in the extension [11].

3.6 USER INTERFACE FOR EXTENSION POPUP

User interface is needed to ensure that the user can interact with the pop ups which is one of the options among several types of user interfaces that a Chrome extension can provide which generally appear upon clicking the extension icon in the browser toolbar. Add the line below to page_action in the manifest file which enables the User Interface for a Popup [12].

In the popup.html file, we include the popup.css file to make the styles available to the HTML elements, popup.js file to enable user interaction and behavior with the HTML elements. Then add a button element named Summarize which when clicked will emit a click event which will be detected by an event listener to respond to it. Add a div element where summarized text will be displayed when received from the backend REST API Call. In the popup.css file, we provide appropriate CSS styling to the HTML elements button and div to have a better user experience [13].

3.7 DISPLAY SUMMARIZED TEXT

A user interface is implemented to enable users to interact and display the summarized text but there are some missing links which must be addressed. In this step, we will add functionality to allow the extension to interact with the backend server using HTTP REST API Calls.

In popup.js, when DOM is ready, we attach the event listener with the event type as "click" to the Summarize button and pass the second parameter as an anonymous callback function. In the anonymous function, we send an action message generated using chrome.runtime.sendMessage method to notify contentScript.js to execute summary generation. Add event listener chrome.runtime.onMessage to listen to message results from contentScript.js which will execute the outputSummary callback function. In the callback function, we display the summary in the div element programmatically using JavaScript. Add the line below to content_scripts in the manifest file which will inject the content script contentScript.js declaratively and execute the script automatically on a particular page [14].

In contentScript.js, we add event listenerchrome.runtime.onMessage to listen message generator which will execute thegenerateSummarycallback function. In the call back function, extract the URL of the current tab and make a GET HTTP request usingXMLHTTPRequestWeb API to the backend to receive summarized text as a response. Send an action messageresultwith summary payload using chrome.runtime.sendMessage to notify popup.js to display the summarized text.

IV. RESULTS & DISCUSSION

4.1 OUTPUT SCREENS

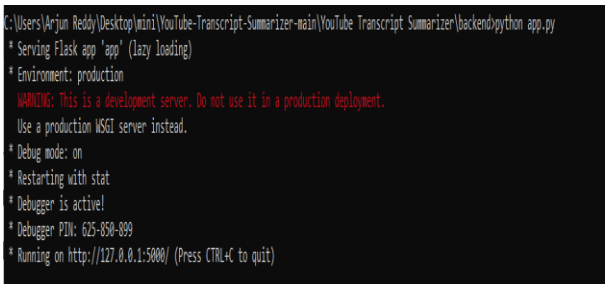

Figure 2 Screenshot of command prompt running backend app.py

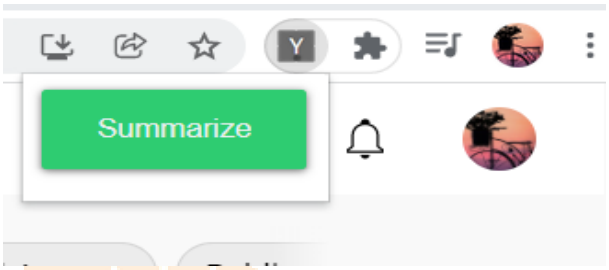

Figure 3 Summarize button of the Chrome Extension.

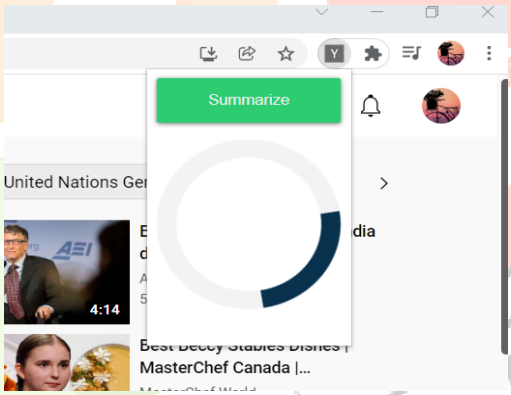

Figure 4 Screenshot of webpage after clicking the button

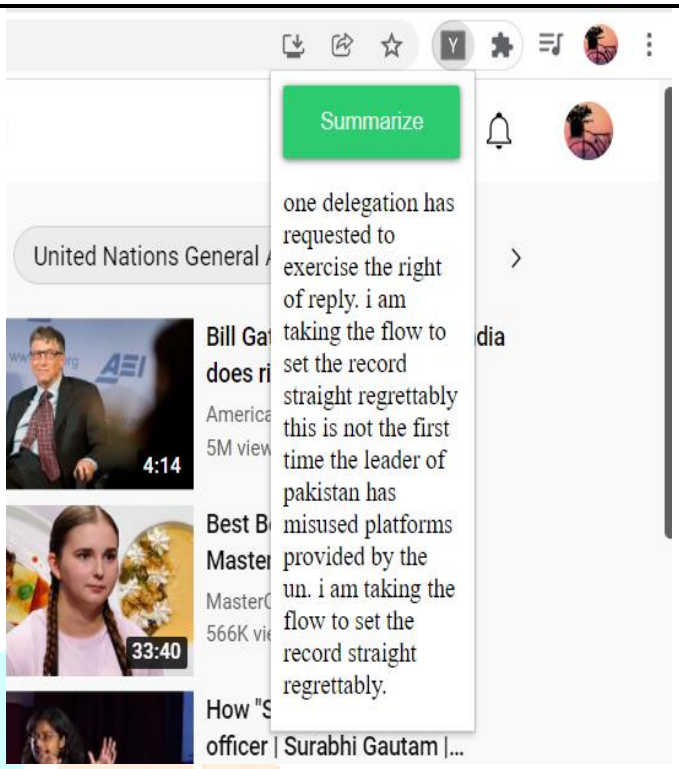

Figure 5 Screenshot of output summarized text

Figure 4 shows that after clicking the button the Request will be generated after generating the summarized transcripts the output will be shown as in Figure 5.

V. CONCLUSION

This project has proposed a YouTube Transcript summarizer. The system takes the input YouTube video from the Chrome extension of the Google Chrome browser when the user clicks on the summarize button on the chrome extension web page, and access the transcripts of that video with the help of python API. The accessed transcripts are then summarized with the transformers package. Then the summarized text is shown to the user in the chrome extension web page.

This project helps the users a lot by saving their valuable time and resources. This helps us to get the gist of the video without watching the whole video. It also helps the user to identify the unusual and unhealthy content so that it may not disturb their viewing experience.

This project also ensures great user interface experience in finding out the summarized text as chrome extensions have been used. This helps in getting the summarized text without copying the URL and pasting at terminals or by any third-party applications.

VI. REFERENCES

[1] 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/

[12] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/page_action
[13] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
[14] https://developer.chrome.com/docs/extensions/mv2/messaging/