e-ISSN:2582-7219

# INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH IN SCIENCE, ENGINEERING AND TECHNOLOGY

Volume 7, Issue 10, October 2024

INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 7.521

☎ 6381 907 438 ☎ 6381 907 438 ✉ ijmrset@gmail.com @ www.ijmrset.com

© 2024 IJMRSET | Volume 7, Issue 10, October 2024| DOI: 10.15680/IJMRSET.2024.0710006

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

# Transcript Summarizer for Youtube

**Harini S, Harshitha Reddy B, Akshaya R, Bhavya G**

Department of Computer Science and Business Systems, R.M.D. Engineering College (An Autonomous Institution),

R.S.M Nagar, Kavaraipettai, India

**ABSTRACT:** In today's digital age, online video content has become an integral part of our daily lives. As a result the need for efficient and time-saving tools for consuming this content has grown substantially. This project presents the development of a YouTube transcript summarizer website—a valuable solution designed to enhance the accessibility and usability of video transcripts. The objective of this project is to create a user-friendly web application that can automatically extract YouTube video transcripts and generate concise and informative summaries. By providing this tool, we aim to cater to a wide range of users, including content consumers seeking quick insights, educators in need of efficient content review, and businesses looking to streamline content management.

## I.INTRODUCTION

As a Computer Science student, you learn on a daily basis from videos, articles, documentation, and so on. A majority of learning happens through Youtube as well. PS Youtube also provides entertainment. A lot of time can be saved if you can summarize the content of the youtube videos. In this project, you will be creating a Chrome Extension which will make a request to the backend REST API where it will perform NLP and respond with a summarized version of a YouTube transcript. 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. Keywords: Text Summarizer, Chrome Extension, HuggingFace transformers, WebAPI.

## II. USECASE SCENARIO

A. Application of Project- Discuss real-world applications of the YouTube transcript summarizer, such as aiding content consumers, researchers, or educators. Explain how this tool can be a time-saver and enhance the learning experience
B. Existing System- Provide an overview of any existing tools or services related to video transcript summarization. Mention their strengths and weaknesses
C. Proposed system- Describe in detail your YouTube transcript summarizer website, including the user interface, features, and how it will extract and summarize video transcripts.

## III. SOFTWARE SPECIFICATION

back-end uses Flask framework to receive API calls from the client and then respond with the summarized text . This API can work only on those YouTube videos which have wellformatted closed captions in it. The same backend also hosts a web version of the Summarizer to make those API calls in simple way and show the output within the webpage.

\ Units
• Use `/` (Root Endpoint): It displays a general purpose introductory webpage and also provides links to web summarizer and API information. You can go to this point [here](https://ytsum.herokuapp.com/).

• `/web/` (Web Summarizer Endpoint): It displays the web version of the summarizer tool. The webpage has input elements and a summarize button. After clicking summarize, the `API` is called and the response is displayed to the user. You can go to this endpoint by directly clicking [here](https://ytsum.herokuapp.com/web/).
• `/api/` (API Description Endpoint): The webpage at this endpoint describes basic API information in case you would like to use it. Feel free to learn and use our API in your projects.
• `/summarize/` (API Endpoint): This endpoint is for **API purposes only**. That is why, the response type of the **`GET Request`** at this endpoint is in JSON format.

### A.BACK END

APIs have revolutionized the way applications are built and there are numerous examples of APIs being used in different applications. To set up our API, we begin by creating a back-end application directory with an app.py file. This file is initialized with a basic Flask RESTful Boilerplate. We then create a virtual environment to isolate the location where all the dependencies will reside. Once the virtual environment is activated, we use pip to install the necessary dependencies, including Flask, YouTube_Transcript_API, and transformers. It is important to ensure that the content is original and not plagiarized to maintain its integrity.

### B. GET TRANSCRIPTS

In this module, we will utilize a Python API to obtain transcripts/subtitles for a specified YouTube video. The API is capable of working with automatically generated subtitles, translating subtitles, and does not require a headless browser like other Selenium-based solutions. In app.py, we define a function that takes the YouTube video ID as an input parameter and returns the parsed full transcript as the output. Since we receive the transcript in JSON format with text, start, and duration attributes, we only extract the text data from the response and return the transcript as a single string. This process allows us to obtain the complete transcript of the video.

## IV. PROJECT DESCRIPTION

The project follows a clear flowchart as shown in Figure 1. Firstly, the user opens a YouTube video and clicks on the "summarize" button in the chrome extension. This initiates a HTTP request to the back-end of the system. Subsequently, the request is made to access the transcripts using the YouTube video ID obtained from the URL. The response to this request will be a transcript of the video in JSON format. Once the transcripts are obtained in text format, the system performs transcript summarization, which involves reducing the length of the transcript while retaining the most important information. Finally, the summarized transcript is displayed on the extension

### A.PERFORM TEXT SUMMARIZATION

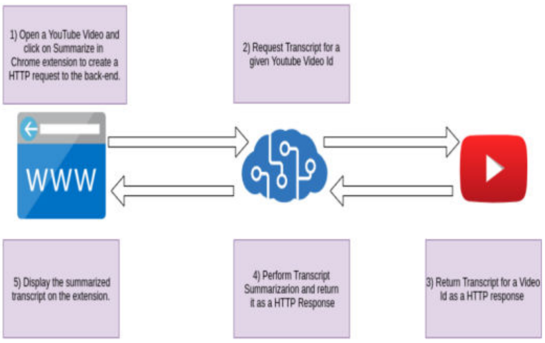
Text summarization refers to the task of condensing longer text into a shorter summary while preserving the key information and meaning of the original text. There are two main approaches used for text summarization: extractive summarization and abstractive summarization. Extractive summarization involves identifying important sentences and phrases from the original text and outputting only the necessary parts, while abstractive summarization involves generating a completely new text that is shorter than the original text, often using encoderdecoder models like Bart or T5. For this project, 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, a function is created that accepts the YouTube transcript as input and returns the summarized transcript as output. To perform the summarization, a tokenizer and a model are instantiated from the checkpoint name. The T5- specific prefix "summarize:" is added to the transcript that needs to be summarized. The PreTrainedModel.generate() method is then used to generate the summary.

### B. REST API ENDPOINT

The next step is to define the resources that will be utilized in the implementation of this backend service. As this is a straightforward application with only a single endpoint, the only resource we need to define is the summarized text. In app.py, we create a Flask API Route with a GET HTTP Request method and a 17 | Page URI of http://[hostname]/api/summarize?youtube_url=#{url}. We then extract the YouTube video ID from the YouTube URL obtained from the query parameters. After that, we generate the summarized transcript by executing the transcript generation function and the transcript summarizer function. Finally, we return the summarized transcript with an HTTP Status OK and handle HTTP exception as required.

### C. DISPLAY SUMMARIZED TEXT

To enable interaction between the extension and backend server, we need to add functionality to make HTTP REST API Calls. In popup.js, we attach an event listener to the Summarize button with the event type "click" and pass an anonymous callback function. In the callback function, we use the chrome.runtime.sendMessage method to send an action message to contentScript.js to generate the summary. We also add an event listener, chrome.runtime.onMessage, to listen for message results from contentScript.js, which will execute the outputSummary callback function. In the callback function, we use JavaScript to programmatically display the summary in the div element. We also need to inject the content script contentScript.js into a particular page and execute the script automatically. In contentScript.js, we add an event listener chrome.runtime.onMessage to listen to the message generator, which will execute the generate Summary callback function. In the callback function, we extract the URL of the current tab, make a GET HTTP request using the XML HTTP Request Web API to the backend, and receive the summarized text as a response. Then, we send an action message result with the summary payload using chrome.runtime.sendMessage to notify popup.js to display the summarized text.


1) Open a YouTube Video and click on Summarize in Chrome extension to create a HTTP request to the back-end.
2) Request Transcript for a given Youtube Video Id
WWW
5) Display the summarized transcript on the extension.
4) Perform Transcript Summarizarion and return it as a HTTP Response
3) Return Transcript for a Video Id as a HTTP response

## ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance, Support and kind co-operation from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

It is our immense pleasure to express our deep sense of gratitude to our respected chairman Thiru R. S. Munirathinam, our vice chairman Thiru R. M. Kishore, and our director Thiru R. Jothi Naidu for the facilities and support given by them in the college.

We are extremely thankful to our principal Dr. N. Anbuchezhian, M.S, M.B.A, M.E, Ph.D., for giving us an opportunity to serve the purpose of education.

We are indebted to Dr. G. Amudha, M.E, Ph.D., Professor, Head of the Department in Computer Science and Business Systems for providing the necessary guidance and constant encouragement for successful completion of this project on time.

We extend our sincere thanks and gratitude to our project guide Dr. S. Deepa B.Tech,M.E, Assistant Professor in the Department of Computer Science and Business Systems, who guided us all along till the completion of our projectwork.

## REFERENCES

[1]. 'Automated Video Summarization Using Speech Transcript' byCuneyt M. Taskiran, Aronon Amir, Dulce B. Ponceleon, Edward J. Delph

[2]. "Digital video Summarization Techniques", Ashenafi Workie, Rajesh Sharma, Yun Koo Chun

[3]. S. Tharun, R. Kranthi Kumar, P. Sai Sravanth, G. Srujan Reddy, B. Akshay, "Survey on Abstractive Transcript Summarization of YouTube Videos", in International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

[4]. Nallapati, R., Zhou, B., Gulcehre, C., & Xiang, B. (2017). Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 31, No. 1).

[5]. Nguyen, T. T., Nguyen, M. Q., Nguyen, L. T., & Nguyen, H. N. (2019). A hybrid approach for summarizing youtube video transcripts. Information Processing & Management, 56(6), 1444-1459.

[6]. Zeng, J., Wei, F., & Liu, S. (2020). Learning to summarize from human feedback on summary prototypes. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 5641- 5647).

[7]. Huang, X., Shi, Y., Xiong, W., & Zhang, J. (2021). EduSum: A largescale dataset and neural model for automated educational video summarization. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 452-462).

[8]. https://atmamani.github.io/blog/building-restful-apis-with-flask-inpython/

[9]. https://pypi.org/project/youtube-transcript-api/

[10].https://medium.com/swlh/parsing-rest-api-payload-and-queryparameters-with-flask-better-than-marshmallowaa79c889e3ca

[11].https://developer.chrome.com/docs/extensions/mv2/

[12].https://developer.mozilla.org/enUS/docs/Web/API/XMLHttpRequest/ Using_XMLHttpReques

INTERNATIONAL JOURNAL OF
MULTIDISCIPLINARY RESEARCH
IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com