YouTube Transcript Summarizer using HuggingFace Transformer

Vijay Mane, Kshitija Dupare, Prashik Dupare, Tushar Ganvir, Rutvik Ghode

Vishwakarma Institute of Technology, Pune, Maharashtra, India-411037

Corresponding author: Rutvik Ghode, Email: rutvik.ghode21@vit.edu

In the vast online realm, where tons of videos pop up every day, our busy lifestyles make it tough to sit through long films. Complicating matters, many videos lack real content and instead use tricks like fake thumbnails and ads to catch attention. The massive growth in YouTube users adds to the flood of videos. To tackle this issue, our project focuses on giving users quick access to useful transcripts, saving them from wasting time on content that doesn't offer much. We aim to speed up how users get knowledge by letting them easily read through video transcripts, avoiding the problems of misleading visuals and ads. Impressively, our project reached a high accuracy of 94.29% in creating these transcripts, especially for videos in English. Behind the scenes, we use advanced models for precise technical work. Looking ahead, we plan to improve the project by making it work with different languages and refining accuracy through smarter language understanding and pattern recognition. This keeps our project as a handy tool for smoothly handling the complexities of watching videos online.

Keywords: HuggingFace converter, Text Summarizer, Chrome Extension, Web API, Flask API.

in: *Advancements in Communication and Systems*. Ed. by Ashish Kumar Tripathi and Vivek Shrivastava. Computing and Intelligent Systems, SCRS, India., 2023, pp. 145–153. DOI: https://doi.org/10.56155/978-81-955020-7-3-13

1. Introduction

The digital era has witnessed an unprecedented surge in video content creation, with YouTube leading the charge as one of the most popular platforms. In 2020, the platform boasted a staggering 2.3 billion active viewers, showcasing the exponential growth in its user base. The continuous influx of content is evident, with over 300 hours of video uploaded every minute. This influx, however, poses a challenge for viewers, as finding relevant information within lengthy videos can be time-consuming and, at times, futile. In India, where mobile usage is widespread, a third of YouTube users spend more than 48 hours per month on the platform. As mobile users rise, the need for efficient video content access becomes crucial.

Despite the plethora of videos available, efficiently extracting desired information remains a significant challenge. Detailed narrations on various topics are abundant, but understanding the essence of a video without watching it in its entirety can be daunting. The need for streamlined access to video transcripts led to the development of tools like the YouTube Transcript API. This API facilitates the extraction of video transcripts, which can then be summarized using advanced techniques such as the hugging face transformer. The proposition of Chrome extensions adds an extra layer of user-friendliness, allowing a "summarize" button to transform the ongoing YouTube video's content into easily digestible text directly within the Chrome browser.

Previous attempts at summarizing YouTube content typically relied on thumbnails and human-generated descriptions. Acknowledging the limitations of these approaches, this project introduces a transformative solution employing the T5 encoder-decoder model. T5 stands out as a versatile model that undergoes pre-training on diverse supervised and unsupervised tasks before being fine-tuned for text-to-text applications. Recognizing the paramount importance of summarization, the project utilizes a pre-trained summarization technique to extract relevant information effectively. This innovation aims to enhance the user experience by providing intelligent and concise summaries of YouTube videos, transcending the traditional reliance on thumbnails and human descriptions.

2. Literature Survey

Automatic summary is a popular technique for distilling a document's key points. It functions by building a shorter version of the text while retaining essential details. Text methods of summarization can be categorized as extractive or abstractive. Extractive summarization approaches lessen the amount of work of summary by choosing a selection of relevant sentences from the authentic text. Although there are other methods, Natural Language Processing researchers have a particular attraction to extractive methods. The implications of terms are identified using linguistic and statistical criteria. This work by I. Awasthi et al.,(2021) examines extractive and abstract techniques for text summarization [1]. It also analyzes all of the methods indicated above, which produces a less repetitious and more condensed report.

Adhika Pramita Widyassari etal., conveyed an in-depth and accurate evaluation of text summarizing studies published between 2008 and 2019 [2]. There are eighty-five journals and conference articles as the result of the extraction and analysis of selected studies to clarify research topics/trends, datasets, preprocessing, features, techniques, methods, evaluations, and problems related to this field of study. The analysis results offer a comprehensive review of the topics/trends that are the focus of their studies in the field of text summarization; present references to accessible datasets, preprocessing, and features that have been used; and describe the techniques and methods that are commonly utilized by researchers as a comparison and means of developing novel methods and models.

Parth Rajesh Dedhia, et al., (2020) made use of Seq2Seq, Encoder-Decoder, and Pointer Mechanism. The model won't work if numerous papers are provided to it, and it is a major drawback [3]. However, the system works efficiently with a smaller number of papers. Summarizers let individuals understand

the material without needing to read it meticulously. Abstractive Text Summarizer aids in defining the content by focusing on important concepts and delivering summaries in a human-readable way. The main objective is to generate summaries that retain their meaning. For the purpose of offering a clear overview, multiple Neural Network models are employed alongside additional machine translation models. This work seeks to highlight and evaluate existing current techniques for abstractive text summarization and also to identify topics for further study.

Aniqa Dilawari and Muhammad Usman Ghani Khan (2019) used multi-line video descriptions and RCNN deep neural network models. Abstractive summation of Video Sequences is implemented in the proposed novel joint end-to-end solution [4], which then relies on a deep neural network to build the natural-language description model with an abstractive text summary of an input video. The result is a text-based video description as well as an abstractive summary, permitting viewers to make decisions between significant and irrelevant information based on their needs. Furthermore, in an actual human evaluation, our tests demonstrate that the combined model outperforms the baseline methods in various duties with informative, succinct, and accessible multi-line video descriptions and summaries.

Denis Jouvet, et al., (2018) looked at the issue of out-of-vocabulary terms in the French, English, and Arabic transcriptions of videos [5].The incorporation of voice recognition in the vocabulary to perform automatic speech transcription is discussed in this work. Using previously accessible data, baseline automated speech recognition techniques have been built. Following a discussion of the baseline ASR systems' performance, the paper presents the collection of recent textual data from the internet for updating the speech recognition vocabularies and training the language models, as well as the elaboration of development data sets required for the vocabulary selection process. The research also compares the coverage of training data gathered from the internet and GigaWord data to finite-size vocabulary composed of the most frequently occurring terms.

Word error rates in standard improved Gaussian Mixture Model (GMM) dependent Acoustical models may go above 50%, making this one of the hardest documented jobs. The use of owner-uploaded video transcripts to generate additional semi-supervised training data and deep neural network audio models with enormous state inventories are covered in the present study. H. Liao, et al., (2013) made use of bigger neural network acoustic models and semi-supervised Auto-sync data, it demonstrated notable advancements in speech recognition applied to YouTube videos [6]. When compared to previously reported sequence-trained DNN results for this task, using a "island of confidence" filtering heuristic to select useful training segments and increasing the size of the model by employing 44,526 dependent on context states with a low-rank final layer weight matrix approximation improved performance by about 13%.

This project by Kumari. P. Vijaya et al., provided a user interface that enhances the user experience, allowing for greater freedom in downloading transcript summary files and automating WhatsApp and email. Summarizing movie transcripts automatically helps to quickly detect important patterns in the film, saving time and effort from having to read through the full material. In this project [7], we utilize Python APIs for text transcription. Natural language processing (NLP) is then used to summarize the transcript. User Interface: HTML, CSS, JS, and Bootstrap for the user interface, and Python as the backend using Flask. The summary forms, such as PDF and Word, can be downloaded by the user and shared by email and WhatsApp. This project gives you hands-on experience with NLP methodologies for abstractive text summarizing.

Reshma Shaik et al., (2023) came at the conclusion that the advantages of summarizers go above simply making articles concise [8]. They are able to help individuals by creating succinct, simpler-to-read summaries of their work. Professionals who need to explain difficult ideas and concepts to a larger audience could especially benefit from this. In addition, summarizers are versatile since they can offer summaries of various lengths based on the demands of the user. While certain applications can provide a summary in just a single phrase, others may create a more thorough summary that touches on every

significant aspect of the text. All things considered, summarizing tools have shown to be an invaluable resource both for professionals and students, allowing them to produce high-quality summaries while optimizing their work and saving time.

As a result, it is clear from the previous research that document or text summarization is an important innovation to prevent wastage of time, and extract useful information from sources in a shorter period of time. So, the proposed model works to provide a concise summary of YouTube videos and enable users to get useful information from videos. It not only helps to save time but also prevents them from watching unnecessary content. So, this can be used at a large scale to make YouTube an efficient learning platform.

3. Methodology

In this project, we have made use of YouTube Transcript API which lets you retrieve the subtitles and transcripts for a specific video ID on YouTube. We then used a library called Pipeline in model HuggingFace transformers to text summarise the obtained transcripts, develop a Flask backend to showcase the summarization service to the viewer/user, and build a Chrome extension that will make use of the backend API to show the users, the summarised text. The below mentioned Figure 1 shows the bird's eye view of the complete project.

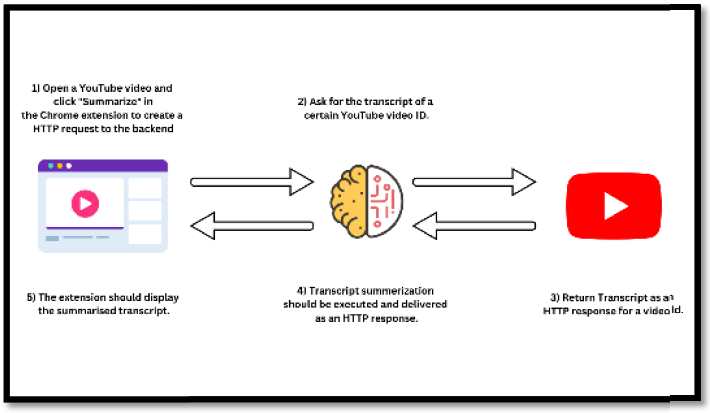

Figure 1. YouTube Transcript Summarizer's System Architecture

First, we open a YouTube video and choose the Chrome extension's summarization option. An HTTP request will be made in response. Then, using the YouTube video ID that was derived from the URL, a request will be performed to obtain the transcripts. A JSON-formatted transcript of that video will be sent as the answer. The system does Transcript Summarization after receiving the transcripts in text format. The synopsis of the transcript is then shown on the extension. The following Figure 2, and Figure 3 displays the block diagram and algorithm of the whole system, respectively.

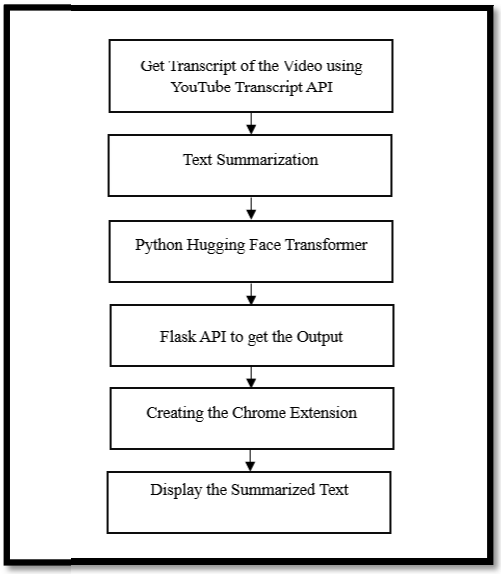

Figure 2. Block diagram to understand the flow of the entire model

Algorithm 1: Algorithm for YouTube Transcript Summarizer.
Input: Video link
Output: Summary

1. if(video has subtitles) then
   Get subtitle
2. end if
3. else
   Extract Audio chunk using Hugging Face
   Automatic speech recognization
   Generate raw text
4. end else
5. Make a transcript
6. Summarize
7. return Summary

Figure 3. Algorithm for the entire model

3.1 Back-end

There are innumerable instances of APIs around the world, and there are several methods to organize or set up our APIs [5]. APIs have transformed the way we develop apps. We make a file called app.py that contains back-end information. App.py is started with a fundamental Flask BoilerPlate [14]. The location where everything is located is then isolated by a virtual environment. Pip is used to install dependencies like Flask, youtube_transcript_api, and transformers by activating the newly built virtual environment [8].

3.2 Get Transcript

This stage makes use of YouTube Transcript API that lets you retrieve the subtitles and transcripts for a specific video on YouTube. We write a function in app.py that considers the YouTube video URL as an input argument and outputs a parsed complete transcript. Since we only require the text attribute of the transcript, which is stored in a variable called transcript list in the fragmented format from the video, we parse the data from the response to return the transcript in a complete string format [9].

3.3 Text Summarization

Text summarization is the practice of keeping the essential points and overall meaning while compressing longer content into a brief summary [11]. The two main techniques for text summarizing are:

- Extractive Summarization: Using the extractive summarization-based approach, the model only produces the keywords and phrases from the source text.
- Abstractive Summarization: The abstractive summarization-based model generates a concise and distinct text that is shorter than the original text, while also creating fresh expressions in an alternative arrangement.In order to carry out this plan, transformers will be used.

The Python HuggingFace transformers model will be used in this system to abstractively summarize the transcript that was received in the previous step. We wrote a function in the main file app.py that will take the string formatted transcript generated during the Get Transcript process as an input argument and output the condensed transcript. Create a model and tokenizer from the checkpoint name after that. Encoder-decoder types like Bart or T5 are used to aid with summarization. Once the transcript to be summarised has been determined, the T5-specific prefix "summarise:" is added. Finally, create the summary using the PreTrainedModel.generate() function.

3.4 Flask API Endpoint

The other step is to represent the resources that will be utilized to carry out this backend service. In our very simple application, there is just one endpoint, so the text summary will be our only resource [10]. The app.py file creates a Flask API Route with the GET HTTP Request method and the http://[hostname]/API/summarize?youtube_url=#url URL. The URL of YouTube that was obtained from the query parameters is then used to retrieve the YouTube video id. Run the transcript generation function after the transcript summarizer has finished its work to create the summarized transcript. We then give you the shortened transcript [10].

3.5 Chrome Extension

Chrome Extensions are computer programs that let you personalize and improve your surfing. They provide users the ability to customize Chrome's behavior and functionality to suit their own tastes. Web-based technologies like CSS, HTML, and JavaScript are used to generate them. The necessary files listed below [10] are created in this phase in a Chrome extension application directory.

To enable the loading of our extension in the browser, we must build a manifest.json file [15]. Make sure developer mode is turned on in the top right-hand corner of the webpage chrome://extensions. The folder holding the manifest file which was just prepared should then be selected after choosing Load Unpacked [16]. We extend it as a result. We will have to refresh the extension every time we make a modification to it [17].

3.6 User Interface and Extension Popup

In order to make sure that the user is enabled to interact with the pop-ups, one option among many different kinds of user interfaces that a Chrome extension might offer must show upon clicking on the icon for extension in the browser's toolbar [18]. By adding the line below to page_action in the created manifest file the User Interface for a Popup is enabled [12]. The popup.html file has been created in order to make the HTML elements and user interaction and behavior with the HTML components, respectively. Then include a button element called Summarise that, on clicking the button, it will send out an event for click that an event listener will be able to detect and react to. [12].

3.7 Display and Summarized Text

There are several gaps that need to be filled in the user interface that is used to connect with users and show the condensed content. In this stage, we'll add features that will let the extension make HTTP Flask API Calls to communicate with the backend server. When the DOM(Document Object Model) is

prepared in popup.js, the event listener with the event type of "click" is attached to the “Summarise” button, and the second parameter is passed as a callback function which is unknown. We use JavaScript in the callback method to programmatically display the summarized content in the div element. The content script contentScript.js will be demonstratively injected and spontaneously run on a certain page.[13] If the following line is added to the content_scripts section of the manifest file. Extract the current tab's URL from the callback function and send a GET HTTP request to the backend using the XMLHTTPRequestWeb API to get a condensed text as a result [15]. Utilise chrome.runtime.sendMessage to send a message to perform an action to result with a summary payload, instructing popup.js to show the summarised content.

4. Results and Discussion

The implementation of the model yielded promising results, showcasing a significant advancement in the efficiency of video content summarization. In the course of testing, a total of 70 videos were subjected to the model's analysis, with 94.29% accuracy achieved in providing accurate video transcripts. Notably, out of the 70 videos, 66 yielded exact overviews of their contents, outperforming other existing models. This substantial accuracy holds promise for users seeking to optimize their time while gaining comprehensive insights into video content. Moreover, the utility of the model is underscored by its ability to swiftly generate transcripts, with a processing time of 2-3 minutes, contingent upon the length of the video. The ensuing Table 1 encapsulates these findings, highlighting the model's efficacy in offering a time-saving and efficient solution for users navigating the vast landscape of online video content.

Table 1. Results obtained from the model

Test Results of Implemented Model	
Total Videos Tested	70
Accurate Video Transcript	66
Model Accuracy	94.29%
Time for Transcript Generation	2-3 minutes
Output Generation Time Variability	Depends on Video Length
User Benefits	Time Saving, Efficient Content Grasping

Hence, the model can help users to save valuable time and grasp the video content in fewer minutes. After, clicking the “Summarize” button, the model takes 2-3 minutes to give the transcript; however, the output generation time also depends on the length of the video. The following figures Figure 4, Figure 5, and Figure 6represents the output of the YouTube Transcript Summarizer project

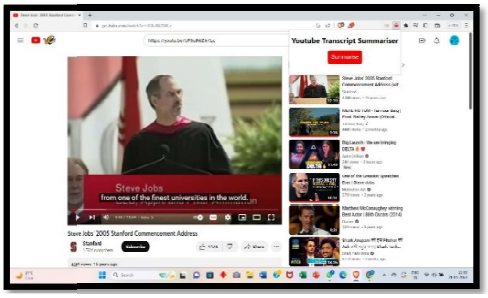

Figure4. Summarize button of the Chrome Extension

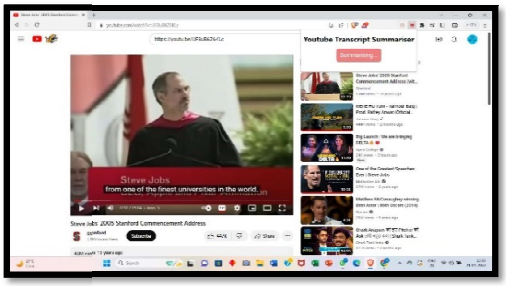

Figure5. Webpage view after Clicking the Button

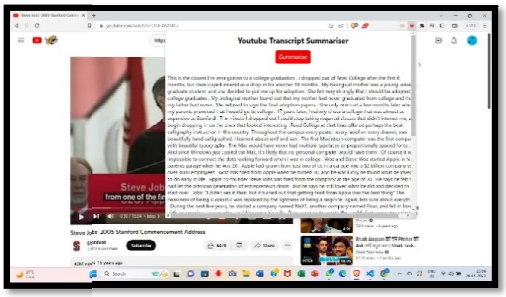

Figure6. Screenshot of Output Summarized Text

Figure6 shows the output after the summarised transcripts that have been generated. Figure5 depicts the view after clicking the button to generate the Request. In this manner, the Chrome extension will function to generate a written summary of any English-language YouTube video. Hence, one can save time by obtaining precise information from any YouTube video. It will not only save one's valuable time but also provide precise information of the discussed content in the video.

5. Conclusion and Future Scope

This project proposes a summarized text of YouTube transcripts. Whenever a user clicks the summarize button on the web page of the Chrome extension, the system receives the input YouTube video from that website and makes use of the YouTube Transcript API to obtain the video's transcripts. The Transformers package then includes a summary of the available transcripts. The user is subsequently presented with the web page for the condensed content of the Chrome add-on. The users of this extension will save a lot of valuable time. This permits us to comprehend the essential ideas of the video without watching it in its entirety.

Furthermore, it helps the user spot odd and damaging information so that it won't obstruct their viewing.This project additionally ensures a top-notch user interface for finding the summarised content due to the use of Chrome extensions. Since the URL doesn't need to be copied and pasted into terminals or given to a third party, it is possible to obtain the condensed content in this way.However, the research only enables getting the exact summarization of English videos. With further development, one can consider multi-lingual video summarization as well.

6. Acknowledgments

For developing this project successfully, we want to acknowledge our heartiest gratitude to Prof. Dr. Vijay Mane for his valuable insights, and for making this “YouTube Transcript Summarizer” a massive success. Also, we would like to thank our institution, Vishwakarma Institute of Technology, Pune (VIT, Pune) for providing us with the opportunity to work on this project and learn more about different technologies. The research and implementation of this project embedded various new skills within us, and we were able to learn more about the internal workings of HuggingFace converter, and YouTube transcript API.

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), Coimbatore, India, 2021.

[2] Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, 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.

[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), Vellore, India, 2020.

[4] A. Dilawari and M. U. G. Khan, "ASoVS: Abstractive Summarization of Video Sequences," in IEEE Access, vol. 7, 2019.

[5] Denis Jouvet, David Langlois, Mohamed Amine Menacer, Dominique Fohr, Odile Mella, et al.. Adaptation of speech recognition vocabularies for improved transcription of YouTube videos. Journal of International Science and General Applications, 2018.

[6] H. Liao, E. McDermott, and A. Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription," 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, 2013.

[7] Kumari P. Vijaya, M. Chenna Keshava, C. Narendra, P. Akanksha, and K. Sravani. "Youtube Transcript Summarizer Using Flask And Nlp." Journal of Positive School Psychology 6, no. 8 (2022): 1204-1209.

[8] Shaik Reshma, Saloni Bargat, and Shilpa Ghode. "Article and YouTube Transcript Summarizer Using Spacy and NLTK Module." SSGM Journal of Science and Engineering 1, no. 1 (2023): 126-131.

[9] https://medium.com/mlearning-ai/summarizing-youtube-video-transcripts-using-hugging-face-transformers-d3765a448cb0

[10] https://medium.com/@jawadkr111/youtube-transcript-api-how-to-deal-with-video-content-017ab88e8180

[11] https://www.geeksforgeeks.org/python-convert-speech-to-text-and-text-to-speech/

[12] https://huggingface.co/docs/transformers/installation

[13] https://atmamani.github.io/blog/building-restful-apis-with-flask-in-python/

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

[15] https://betterprogramming.pub/the-ultimate-guide-to-building-a-chrome-extension-4c01834c63ec

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

[17] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest

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