VIDEO TRANSCRIPT SUMMARIZER

A PROJECT REPORT

Submitted by

MANIKANTA REDDY KYATHAM

(18113047)

Under the guidance of

Mr. C. Gowdham

Assistant professor

In partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

HINDUSTAN INSTITUTE OF TECHNOLOGY AND SCIENCE

MAY 2022

HINDUSTAN
INSTITUTE OF TECHNOLOGY & SCIENCE
(DEEMED TO BE UNIVERSITY)
CHENNAI

BONAFIDE CERTIFICATE

Certified that this project report **Video Transcript Summarizer** is the bonafide work of **Manikanta Reddy K (18113047)** who carried out the project work under my supervision during the academic year 2**021-2022**.

Dr. J. THANGAKUMAR,	Mr. C. Gowdham,
HoD,	**SUPERVISOR**
Department of CSE.	ASSISTANT PROFESSOR

INTERNAL EXAMINER	**EXTERNAL EXAMINER**
Name:_________________________	Name: ________________________
Designation: ___________________	Designation:___________________

Project Viva-voce conducted on
____

TABLE OF CONTENTS

ACKNOWLEDGEMENT

First and foremost, we would like to thank **ALMIGHTY** who has provided us the strength to do justice to our work and contribute our best to it.

I wish to express our deep sense of gratitude from the bottom of my heart to my guide **Mr. C. Gowdham (Assistant Professor)**, **Computer Science and Engineering,** for his motivating discussions, overwhelming suggestions, ingenious encouragement, invaluable supervision, and exemplary guidance throughout this project work.

We would like to extend our heartfelt gratitude to **Dr. J. Thanga Kumar, Ph.D., Professor & Head, Department of Computer Science and Engineering** for his valuable suggestions and support in successfully completing the project.

We want to thank our Project Co-Ordinator and Panel members for keeping our project in the right track. We would like to thank all the teaching, technical and non-technical staff of Department of Computer Science and Engineering for their courteous assistance.

We thank the management of **HINDUSTAN INSTITUTE OF TECHNOLOGY AND SCIENCE** for providing us the necessary facilities and support required for the successful completion of the project.

As a final word, we would like to thank each and every individual who have been a source of support and encouragement and helped us to achieve our goal and complete our project work successfully.

ABSTRACT

The field of text summarization has been evolving along with the facilitations in the field of Natural Language Processing (NLP) and Computer Science. This project investigates the field of text summarization, giving an overview of different approaches towards automatically generating summaries. Further, a clarification towards the requirements and scope towards implementing a summarizer for video transcripts is given.

The length of a transcript can be shortened by applying extractive summarization with Bert model and then the T5 model is used. Here, currently Sumy with Latent Semantic Analysis (LSA) summarizer is used. The method is proved to be a good for a base summarizer to tailor for video transcripts. The quality of the summaries generated by the summarizer were evaluated by comparing them to human created summaries, the evaluation showed that the summarizer performs consistently and produces readable summaries.

LIST OF ABBREVIATIONS

Abbreviation	**Expansions**
AI	Artificial Intelligence
API	Application Programming Interface
DOM	Document Object Model
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
LSA	Latent Semantic Analysis
ML	Machine Learning
NLP	Natural Language Processing.
REST	Representational state transfer
URL	Uniform Resource Locator

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. Overview

Machine Learning (ML) as a domain is the future in all aspects of technology. It is the heart of simulation of humans in machines programmed in a way such that it thinks like humans and mimics their actions. This paved the way to so many other subfields of ML like, Deep Learning, Neural Networks, Natural Language Processing, and many other things. This made possible so many new technologies like text to speech, and speech to text engines, Image recognition, voice recognition, facial recognition, and has changed the way we see the world. We have come a long way with constant development in these fields now trying to make them more efficient and flexible. Thanks to the open-source and AI developer communities, programming a complex ML algorithm is now made easy, thereby making developers explore deeper. To apply ML in any field we need lots of data. The ML used in such fields as chess-playing to self-driving cars relies heavily on deep learning and natural language processing.

Text summarization is one such technique under ML which summarizes the long paragraphs of text into simple understandable text.

To conclude, AI in its short existence has increased understanding of the nature of intelligence and provided an impressive array of applications in a wide range of areas. It has sharpened the understanding of human reasoning, and the nature of intelligence in general.

1.2. Problem definition and scenarios

Luhn proposed one of the first text summarizing methods (Luhn, 1958). His method for generating technical paper abstracts was to utilize a frequency count of words and phrases to determine important material. This strategy is simplistic, yet it provides a good result and a new perspective on how summaries can be constructed from any text Later, as the areas of concern grew, Machine Learning and Natural Language Processing (NLP) are becoming increasingly popular, more advanced text summarizing approaches have been proposed as technology has progressed.

Long videos or texts can be unpleasant and time-consuming to go through or read, and many of us would prefer not to be obliged to do so. But what if there was an accurate summary of the whole video that we could trust to accurately portray the full content turns out to be the solution.

This would free up time and resources, allowing the individual to concentrate on other matters. Creating accurate and informative summaries, on the other hand, is a challenge. It's not a simple task. There are various factors to consider, some of which are related to the specified output format for the summary or the technique for generating the summary.

There are a lot of videos on YouTube that have transcripts. Summarization would be especially useful in circumstances where films are lengthier and different segments have varying degrees of relevance. In this respect, video summarization may be beneficial in terms of saving the viewer's time. It will increase user productivity by allowing them to concentrate solely on the important text spoken in the video.

1.3. Motivation

Throughout the day, an enormous number of video recordings are generated and shared on the Internet. It has become quite difficult to devote time to watching such videos, which may last longer than expected, and our efforts may be in vain if we are unable to extract useful information from them. Summarizing transcripts of such videos allows us to rapidly look for relevant patterns in the video and saves us time and effort from having to go through the entire content.

This stood as a driving factor to find ways to make this process a whole lot easier. The objective stands to identify the most important information from the given transcripts and present it to the end users post summarization request.

1.4. Organization of the Thesis

The literature survey is given under chapter two where the collected reference papers have been mentioned along with its detailed and comparative explanation. The third chapter describes the overall project, details about the existing systems that support the same idea of the project and the proposed work distinguished from them. The proposed system design has been mentioned in detail under chapter 4, which also includes the design flow of the project. The fifth chapter holds information on the requirements of the project including details of the software programs and applications used in the development of the system. Chapter 6 contains the module description of the project and explains the split-up of each module in detail. The seventh chapter explains the implementation of the proposed system followed by the eighth chapter which highlights the results and analysis of the project. The ninth chapter features the scope for future development of this project and how it can be modified for better implementation. It then finally concludes the project. The final chapter, chapter 10 presents the team report, which details the objectives. Reference papers that were referred for the development of this system along with the sites referred are mentioned in the end.

1.5. Overview of documentation

This document contains information about the proposed model of achieving the video transcript summarization via chrome extension. It contains information on the software, programming languages and techniques used, Also the packages used in developing this project. Block diagrams are included to inform the project flow. References of standard research papers are also added. It contains the future scope of the project for further development.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

In the field of AI, under which the project is being developed there are numerous research documents that will support the development of this project, especially in the specific field of text summarization. Papers produced here, also discuss the types of text summarization like extractive summarization and abstractive summarization and various methods of carrying them out.

2.2 Review paper on Automatic Text Summarization

This paper discusses content summary as a method of condensing the source archive into a dense structure that maintains a common understanding of the record. Abstractive and extractive content summarizing methods exist. For outlining the records, abstractive summarization necessitates the use of certain language preparation mechanisms. For situating the sentences, extractive summarization involves factual, etymological, and heuristic procedures. For the summary of material in various languages, a variety of methodologies have been developed. The methodologies for abstractive and extractive text summarization are examined in this work.

To process, store, and manage data effectively, it is now necessary to create a large amount of data quickly. It can be difficult to find the relevant data from massive amounts of stored data or vast information archives at times. With the advent of technologies such as big data, the capacity to mine information in many formats, whether textual or graphical, has never been more significant, and data mining has never been more important.

2.3 Automatic Text Summarization – A Comprehensive Survey

Because of the massive volume of textual material that increases exponentially on the Internet and the numerous archives of news stories, scientific papers, legal documents, and so on, Automatic Text Summarization (ATS) is becoming considerably more significant. Manual text summarizing takes a lot of time, effort, and money, and it's even unfeasible when there's a lot of text. Since the 1950s, researchers have been working to develop ATS procedures. There are three types of ATS approaches: extractive, abstractive, and hybrid. The extractive method chooses the most essential sentences from the input document(s) and concatenates them to create the summary. The abstractive technique converts the input document(s) into an intermediate representation before generating a summary containing phrases that differ from the originals.

Both the extractive and abstractive processes are used in the hybrid approach. Despite all the methodologies presented, the produced summaries still lag behind human-authored summaries.

The extractive strategy is the focus of most studies. It is necessary to place a greater emphasis on abstract and hybrid techniques. This study offers scholars with a complete assessment of the many components of ATS, including methodologies, methods, building blocks, procedures, datasets, evaluation methods, and future research goals.

2.4 Literature study

Two papers are taken to comparison, namely, “Review Paper on Automatic Text Summarization” and “Automatic Text Summarization – A Comprehensive Survey”. They will be addressed as former and latter in this section’s proceeding.

Authors for the former paper, Ujjwal Rani, Karambir Bidhan and for the latter are, Wafaa S. El-Kassasa, Cherif R. Salamaa, Ahmed A. Rafeab , Hoda K. Mohamed and are published in International Research Journal of Engineering and Technology (IRJET)and cited in ResearchGate respectively.

Methodology they propose are, Condensing the source archive into a dense structure that maintains a common understanding of the record which further divides the image into grids. Upon which the extractive and abstractive summarization techniques are applied. While the latter describes three varies methods to deploy to achieve automatic text summarization which are namely abstractive, extractive and hybrid. A brief explanation of each of the method is discussed.

2.5 Overview of Literature Review

This chapter briefs about the various methods of text summarization and their types. A crisp explanation of their implementation pertaining to the work done in this project and also their pros and cons have been discussed with a detailed and comparative explanation.

CHAPTER 3
PROJECT DESCRIPTION

3.1 Objective

Text summarization is a technology that has been rapidly growing with increases in its day-to-day applications. Summarization of text in various scenarios, to be particular of the project's scenarios of summarizing the transcripts of a video is one such approach towards making the jest of a video being watched clearly understood with having to spend whole duration of the video watching it. Even though there are many methods in achieving this, procuring the. This approach provides a solution by making a chrome extension available to the end user and by the request through which the summary of the video's transcripts is visible below it.

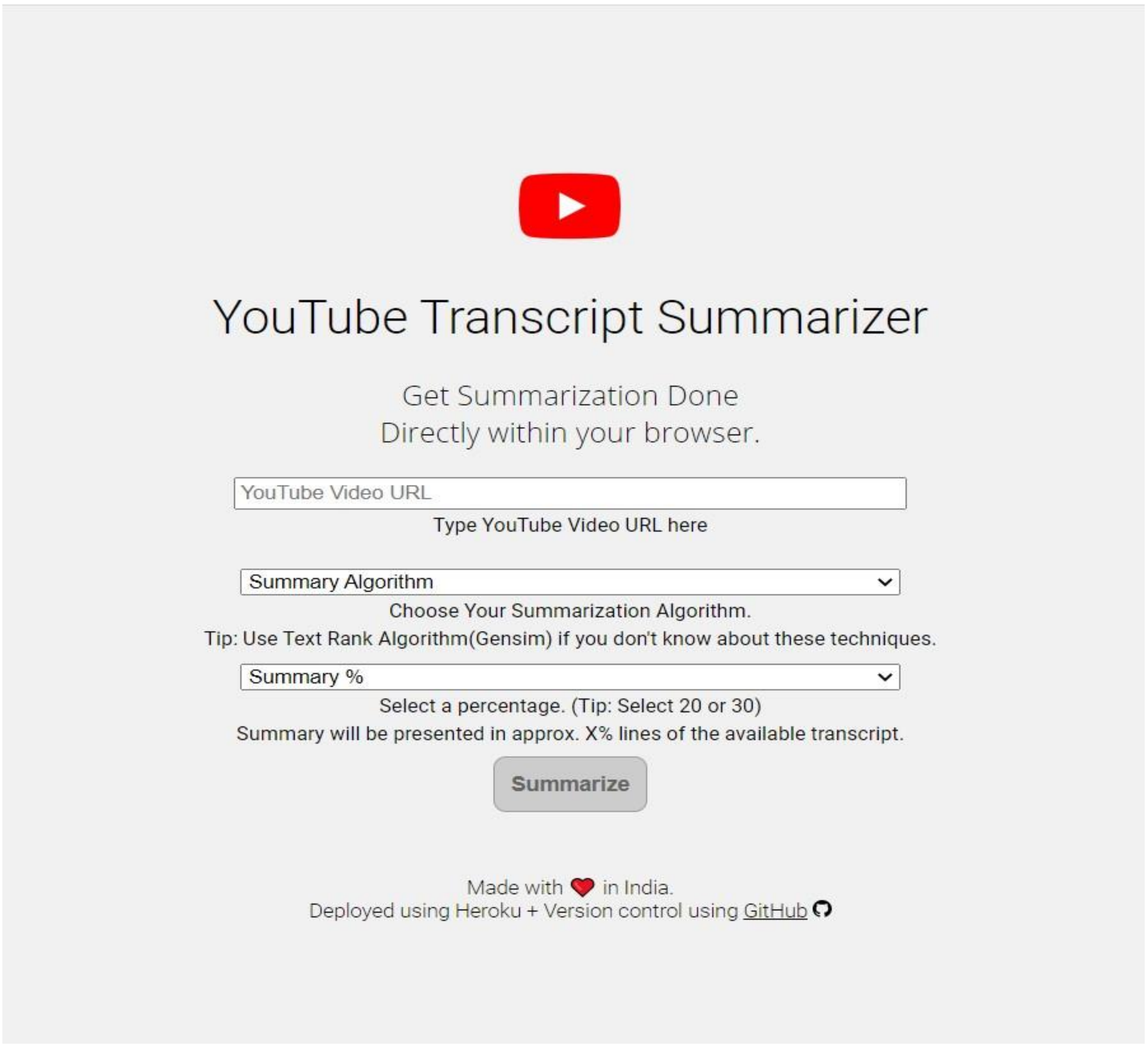
3.2 Existing Systems

Even though many systems and architectures exist to perform text summarization, this project demonstrates a different approach of accessing the backend python API, that is, in the form a chrome extension, which upon clicking the summarize button provided makes a http request to the backend to acquire the summarized text for the obtained transcripts of a video. However, when it comes to improving or enhancing the object detection model, there are very few solutions available. One such method is where a backend is hosted on a server and a request for the API is made by pasting the URL of the video in the placeholder of the label provided in the frontend. The following sub chapter details the method.

3.2.1 YouTube video summarization over Flask.

The backend here uses Flask framework to receive API calls and then respond to the calls with a summarized text. The API here can only work on the YouTube videos which have well formatted transcripts available. It hosts a web version of the Summarizer to make the API calls in an easier way and show the output within a webpage. The figures 3.1 and 3.2 below show the working and GUI interface of the system respectively. In this existing system, the user should manually paste the URL of the video to acquire the summarized transcripts.

Figure 3.1 Working of the Flask app.



YouTube Transcript Summarizer

Get Summarization Done
Directly within your browser.

YouTube Video URL
Type YouTube Video URL here

Summary Algorithm
Choose Your Summarization Algorithm.
Tip: Use Text Rank Algorithm(Gensim) if you don't know about these techniques.

Summary %
Select a percentage. (Tip: Select 20 or 30)
Summary will be presented in approx. X% lines of the available transcript.

Made with ❤ in India.
Deployed using Heroku + Version control using GitHub

Figure 3.2 User Interface.

3.3 Proposed system

To tackle the problem of the mentioned existing systems in the above sub chapter, a solution for acquiring transcripts of a video, a high-level approach is given in the following steps:

- To get transcripts/subtitles of a YouTube video using a python API
- And then to perform text on obtained transcripts using HuggingFace transformers.
- A flask backend REST API is then built to expose the summarization service to the end user.

- A chrome extension is then developed which will utilize the backend API to display the summarized text to the user.

3.4 Overview of Project Description

The third chapter, project description sums up the brief description of the overall project. It continues to detail the existing system relevant to the proposed project and their weakness. Without putting an end there, the chapter also includes the details of the proposed system which is set to break the weakness mentioned in the existing system.

CHAPTER 4
SYSTEM DESIGN

4.1 System flow

The system design is from a high-level point of view is split into five stages which together will make up the final product. The idea behind the system is to display the summarized text of the video transcript from a chrome extension. A simple visual representation of the system design is given below in fig.4.1.

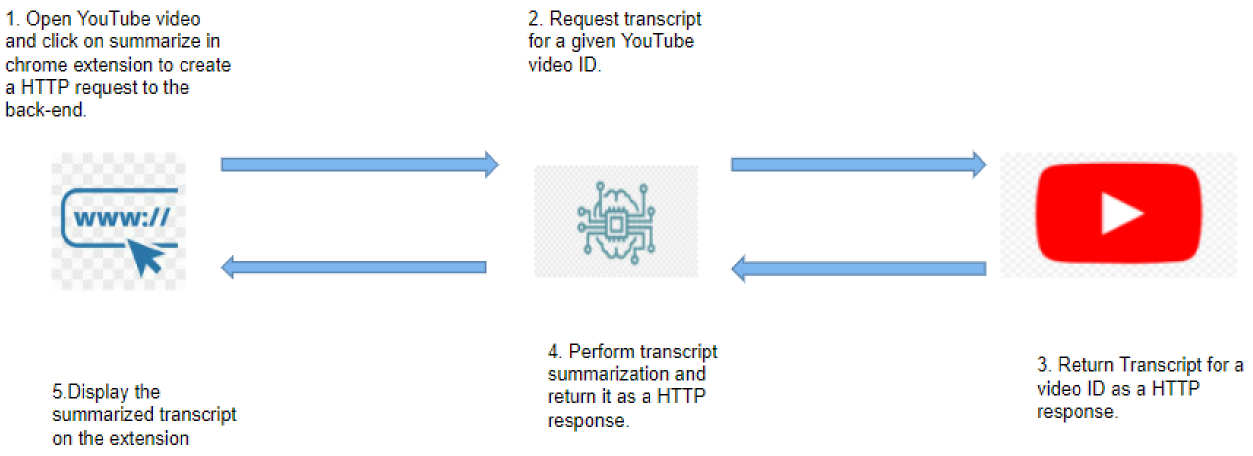

1. Open YouTube video and click on summarize in chrome extension to create a HTTP request to the back-end.
2. Request transcript for a given YouTube video ID.
www://
3. Return Transcript for a video ID as a HTTP response.
4. Perform transcript summarization and return it as a HTTP response.
5.Display the summarized transcript on the extension

Figure 4.1 Flow diagram

The system flow starts from the end-user who will open a YouTube video and click on summarize in chrome extension to create a HTTP request to the backend. This action requests a transcript for the given YouTube video ID. The backend then returns the transcript for the relevant video ID as a HTTP response. Further, transcript summarization is performed and is returned as a HTTP response too. Lastly, the summarized transcript of the requested video is displayed on the interface.

4.2 Text Summarization

Summarization is the task of condensing a piece of text to a shorter version, reducing the size of the initial text while at the same time preserving key informational elements and the meaning of content. Because manual text summarizing is a time-consuming and inherently tedious activity, automating it is expanding in popularity and so serves as a powerful motivator for academic study.

In general, there are two approaches for automatic summarization: extraction and abstraction.

The extractive approach:

Extractive summarization uses a scoring mechanism to extract sentences from a document and combine them into a logical summary. This method works by detecting key chunks of the text, cutting them out, then stitching them back together to create a shortened form.

As a result, they rely solely on phrase extraction from the original text. Most of today's summarizing research has centered on extractive summarization, which is easier and produces naturally grammatical summaries with minimum linguistic processing. Extractive summaries also include the most essential sentences from the input, which can be a single document or a collection of papers.

The following is a typical extractive summarization system flow:

1. Creates an intermediate representation of the input text in order to discover the most important information. The metrics are computed for each sentence in the provided matrix in most cases.
2. Scores the sentences based on the representation, assigning a number to each sentence that indicates the likelihood of it being included in the summary.

3. Generates a summary from the top k most important sentences. Latent semantic analysis (LSA) has been utilized in several research to identify semantically significant sentences.

The Abstractive approach:

Abstractive summarization methods aim to produce summaries by interpreting the text using advanced natural language techniques in order to generate a new shorter text — parts of which may not appear in the original document — that conveys the most critical information from the original text, requiring rephrasing sentences and incorporating information from the full text to generate summaries like those produced by a human-written abstract. In fact, an acceptable abstractive summary is linguistically fluent and covers key information from the input.

As a result, they aren't limited to only picking and rearranging passages from the original text.

Abstractive approaches make use of recent deep learning advancements. Abstractive methods take advantage of the recent success of sequence-to-sequence models since it can be thought of as a sequence mapping task where the source text should be mapped to the target summary. A neural network reads the text, encodes it, and then generates target text in these models, which are made up of an encoder and a decoder.

Building abstract summaries is a difficult process that is more difficult than data-driven alternatives like sentence extraction and requires advanced language modelling. Despite recent advancements employing neural networks inspired by the progress of neural machine translation and sequence to sequence models, they are still far from approaching human-level quality in summary creation.

4.2.1 Backend

APIs revolutionized the way we design apps; there are countless examples of APIs throughout the world, as well as several ways to structure and set up your APIs.

Here we create a back-end application directory and structure it to work with the required files. Back-end of the application needs to be isolated to avoid conflicting dependencies from other parts of the project.

The requirements and the process adapted is

• Make a directory for the back-end programme that contains the files `app.py` and `requirements.txt`.

• Create an `app.py` file with a basic Flask RESTful Boilerplate.

• Install pip in a new virtual environment that will serve as an isolated environment where everything is kept in one place (a directory).

• Activate the newly created virtual environment and install the dependencies listed below.

Using pip: -

- The flask
- youtube_transcript_api
- transformers[torch]

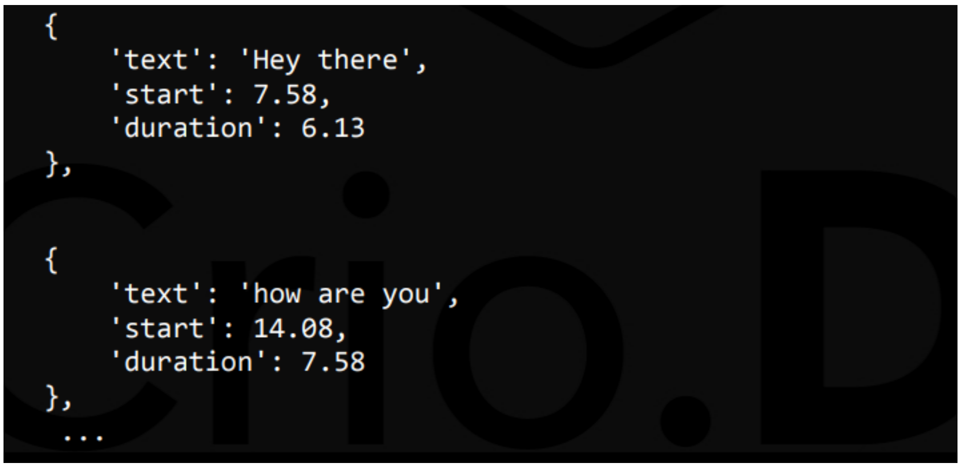
• Run pip freeze and save the output to a file called requirements.txt. This

The `requirements.txt` file specifies which Python packages are necessary to run the program.

4.2.2 Acquiring transcripts of a video

Here we utilize a python API which allows us to get the transcripts of a given YouTube video.

We construct a function in app.py that takes a YouTube video id as an input argument and returns a parsed full transcript as output. - The Transcript API will return a collection of dictionaries that looks somewhat like this:



```
{
    'text': 'Hey there',
    'start': 7.58,
    'duration': 6.13
},

{
    'text': 'how are you',
    'start': 14.08,
    'duration': 7.58
},
 ...
```

Figure 4.2 Transcript output.

We then parse the data from the output to return the transcript in whole string format.

4.2.3 Performing text summarization

Create a function in app.py that takes a YouTube transcript as an input argument and outputs a summary transcript. - From the checkpoint name, create a tokenizer and a model. An encoder-decoder model, such as Bart or T5, is typically used for summarization. - Establish the transcript to be summarized. - Add the "`summarise: `" T5-specific prefix. - To generate the summary, use the `PreTrainedModel.generate()` method.

4.2.4 REST API endpoint

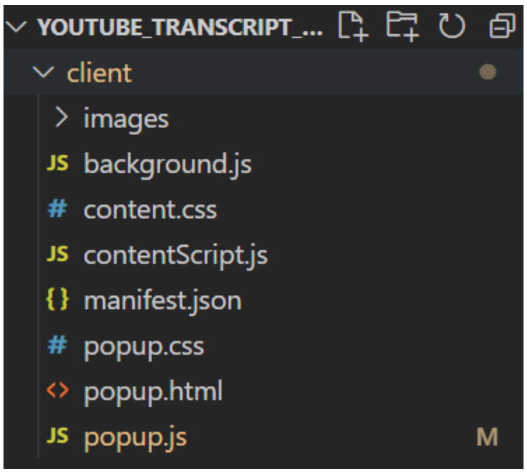
In `app.py`, create a Flask API Route with GET HTTP Request method with URL `http://[hostname]/api/summarize?youtube url=url>.`
- Get the YouTube video id from the YouTube URL you got from the query

params. Following the execution of the transcript summarizer function, run the transcript creation function to generate the summarized transcript.
- Handle HTTP errors and return the summary transcript with HTTP Status OK.

4.2.5 Chrome extension

Extensions are small programs that allow us to customize our browsing experience. They allow users to customize Chrome's functionality and behavior based on their personal preferences. They are built on web technologies such as HTML, CSS and JavaScript.

An application directory containing essential files is created.



```
YOUTUBE_TRANSCRIPT_...
  client
    images
    background.js
    content.css
    contentScript.js
    manifest.json
    popup.css
    popup.html
    popup.js                M
```

Figure 4.3 Application directory.

The below pictorially depicts different parts of the chrome extension in terms of files.

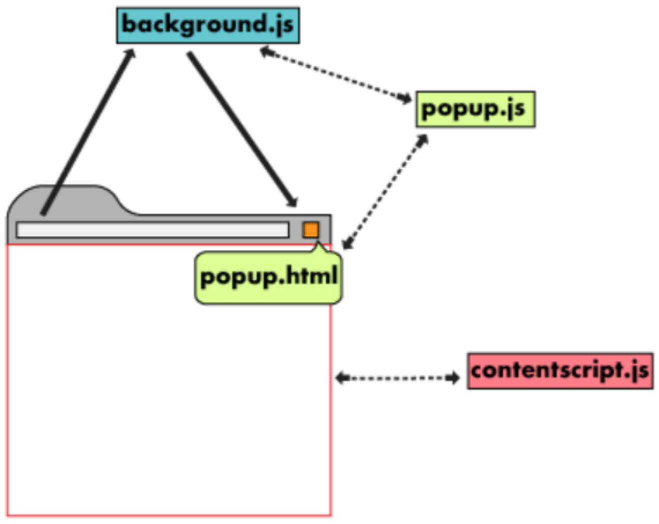


Figure 4.4 Chrome extension diagram.

4.2.6 User Interface – extension pop-up

A user interface is designed to allow the user to interact with popups, which are one of numerous sorts of user interfaces that a Chrome extension can provide. They appear upon clicking the extension icon in the browser toolbar.

• In the `popup.html` file,

- Include the popup.css file to make the styles available to the HTML elements.

- Include the `popup.js` file to allow users to interact with the HTML components.

- Add a Summarize button element that, when clicked, emits a click event that an event listener will notice and respond to.

- Add a div element where the summarized text from the backend REST API call will be shown.

• In the file `popup.css`,

- To improve the user experience, provide appropriate CSS styling to the HTML components button and div.

4.2.7 Display summarized text

We've developed a basic user interface to allow people to interact with and view the summarized text, however there are a few missing links that need to be fixed. The extension now can communicate with the backend server utilizing HTTP REST API Calls.

• In `popup.js`,

- When DOM is ready, attach the event listener with event type as "click" to the Summarize button and pass the second parameter as an anonymous callback function.
- In an anonymous function, use the `chrome.runtime.sendMessage` method to send an action message to contentScript.js, notifying it to run summary creation.
- Add the `chrome.runtime.onMessage` event listener to listen for message results from contentScript.js, which will call the `outputSummary` callback method.
- Using JavaScript, programmatically display the summary in the div element in the callback function.

In `contentScript.js`,

- To listen `messagegenerate`, add an event listener `chrome.runtime.onMessage`, which will trigger the `generateSummary` callback method.
- In the callback function, extract the current tab's URL and send a GET HTTP request to the backend using the XMLHTTPRequest Web API to receive summary text as a response.
- Send an action message result with summary payload using `chrome.runtime.sendMessage` to notify `popup.js` to display the summarized text.

4.3 Overview of System Design

This chapter discusses the overview of the system design proposed for this system. It describes the flow of the design in detail which is followed by actions that are performed in each stage of the design system. In summary, it discussed the detailed view of the system architecture.

CHAPTER 5
PROJECT REQUIREMENTS

5.1 Software used

Python 3.7.5 and JavaScript are used for programming the whole application. Since python has an intuitive syntax, basic control flow, and data structures and supports interpretive run-time, without standard compiler languages, it makes it especially useful for prototyping algorithms for AI and JavaScript for its component as an object feature, Python as a programming language and JavaScript as a scripting language were chosen over other languages for the development of this application.

5.2 Libraries and frameworks used

The list of Python libraries and frameworks used in the development of this project is mentioned in this section.

a) flask

Flask framework has been used as it is a small and lightweight framework that provides tools and features to create a web application easier. It becomes a more accessible framework as it gives flexibility in terms of building a web application quickly using a single python file.

b) youtube_transcript_api

youtube_transcript_api is a python API that allows us to get the transcripts/subtitles of a YouTube video. It also works for automatically generated subtitles and also supports translating subtitles.

c) sumy

Simple library and command line utility for extracting summary from HTML pages or plain texts. The package also contains simple evaluation framework for text summaries.

d) nltk

NLTK is a standard python library that provides a set of diverse algorithms for NLP. It is one of the most used libraries for NLP and Computational Linguistics.

e) numPy

NumPy is a library for the Python programming language, and it adds support for large Multidimensional arrays and matrices along with a large collection of high-level mathematical functions. To operate on these arrays.

f) pyngrok

pyngrok is a Python wrapper for ngrok that manages its own binary and exposes ngrok through a Python API.
ngrok is a reverse proxy tool that creates secure tunnels from public URLs to localhost, making it ideal for exposing local web servers, building webhook integrations, enabling SSH access, testing chatbots, demoing from your own machine, and more. It's even more powerful with pyngrok's native Python integration.

5.3 Overview of Project Requirements

This chapter briefs out on the requirements of the proposed project. The project is entirely a software product; therefore, the requirements are only software-based, and no hardware components are used in this project. The chapter briefs on the programming languages used and the libraries/packages used in the development of the system along with their use case.

CHAPTER 6
MODULE DESCRIPTION

The proposed system design was set out to be as two divided modules in terms of implementation. The second module, server, which is flask app contributes majorly to the design of the system’s backend, summing up to 60% of the total work. It deals with work directly on processing, summarizing the acquired transcripts. This chapter talks about the splitting of the modules and the details of each module

6.1 Server-side

The first module, namely the server is a simple Flask app with an API `/api/summarize?youtube video='url'` that can be used to get the summary of a YouTube video by making a simple GET XML HTTP request. The work in this module revolves around the acquiring the transcript file from the backend and making it available to process text summarization and then eventually is sent to the frontend to be displayed in the form of an API. The API is the JSON format.

6.2 Client-side

It's a chrome addon that makes use of the API from the server module to render the summary of a YouTube video underneath the video player. Summarize button is clicked to see a synopsis of the YouTube video.

6.3 Overview of Module Description

The chapter of module description details about planning and structuring the effort required to implement the proposed system by dividing it into two modules. It lays out a description of each module and determines the effort required of each module. The total effort required of 100% is divided into two parts depending upon the weight of each module.

CHAPTER 7
IMPLEMENTATION

7.1 Project Flow

The project has been implemented with the central idea of performing text summarization to the transcripts of the requested YouTube video by taking the ID of it from the browser url and return the summarized transcript beneath the video along with the timestamps which upon clicking them will take to that part of the video. A basic interface to request for the summary is made available with chrome extension upon clicking which the server loads the summary beneath the video. And finally, the user interface of the web forum was successfully integrated with the YouTube interface, making the summarization service seamless. With this the project is considered to be implemented successfully as per the requirements and flow which was designed in the early stages of the project. Overall, the project can be said as the beginning foundation of a much greater vision for a highly efficient system.

7.2 Working of System

The system as proposed finds it working as per the project flow. The system will work in the following way to generate the desired output. The end user will click on the extension icon which then provides the summarize button, clickable. This is how, upon clicking the summarize button, user requests for the transcript summary.

Clicking the button is technically making an HTTP request for the backend to for the transcript. Upon receiving the request YouTube from the backend provides the relevant transcript file pertaining to the video ID.

Receiving the transcript file, it is now subjected to text summarization and upon completion of which it is passed to the popup.js file which forms the user interface for the summarized text to be displayed.

7.3 Server

Flask is used to implement the server side as a restful service. The summarization is done by first generating a transcript of the video, which is done with the aid of a python library called `youtube-transcript-api` if the video already has a transcript, else the audio is taken and voice to text transformation is done. This time, several useful Python libraries are used.
The summary can then be created with the help of the transformers.
-Extractive summarization
-Abstractive summarization
The transcript length can be reduced by using extractive summarization with the Bert model, followed by the T5 model.
Sumy with the LSA summarizer is currently in use.

After making a GET HTTP request to /api/summarize?youtube video="a valid url," the summary is returned as an HTTP response.
To deliver the request over HTTPS (since YouTube is a https website and sending an HTTP request to a http website may result in a mixed content error), the app must operate on https rather than http. There are two possible ways for this:

1.Using pyngrok (a wrapper package for ngrok) to expose local host on a public url is a simple alternative. It also gives us the option of using https or http for the url, so it accomplishes our goal. This even allows the program to run in environments where localhost isn't available, such as Google Colab. As a result, this is a good option for testing. The server folder contains the colab notebook that is used for testing.

2.Another alternative is to host it on a server that can provide the necessary https url and certificates.

CORS had to be introduced as well, because if the HTTP request is headless, it will be rejected owing to the CORS policy: The requested resource does not have a 'Access-Control-Allow-Origin' header. Using the flask cors package, CORS is simply enabled for all domains on all routes.

7.4 Client (chrome extension).

If the address is of the form https://www.youtube.com/watch?v=**, the `popup.js` sends a GET request to our *Server API when you click the summary button on the popup. A preload text is placed to a div element beneath the youtube player. The text is then provided to `content.js`, which modifies the content inside the above div element. Many of the characteristics are inherited from the parent element, allowing it to blend in perfectly. In the content, more styling is applied. It may be loaded unpacked from chrome:/extensions/ and utilized.

7.5 Overview of Project Implementation

This chapter deals with the execution of the various phases of the project. It focuses on the highlights of how the project is being functioned with emphasis on the desired output. The project implementation abides by the designed system flow and takes a step-by-step approach based on the divided module which helps succeed the core objective of the project along with obtaining optimum results.

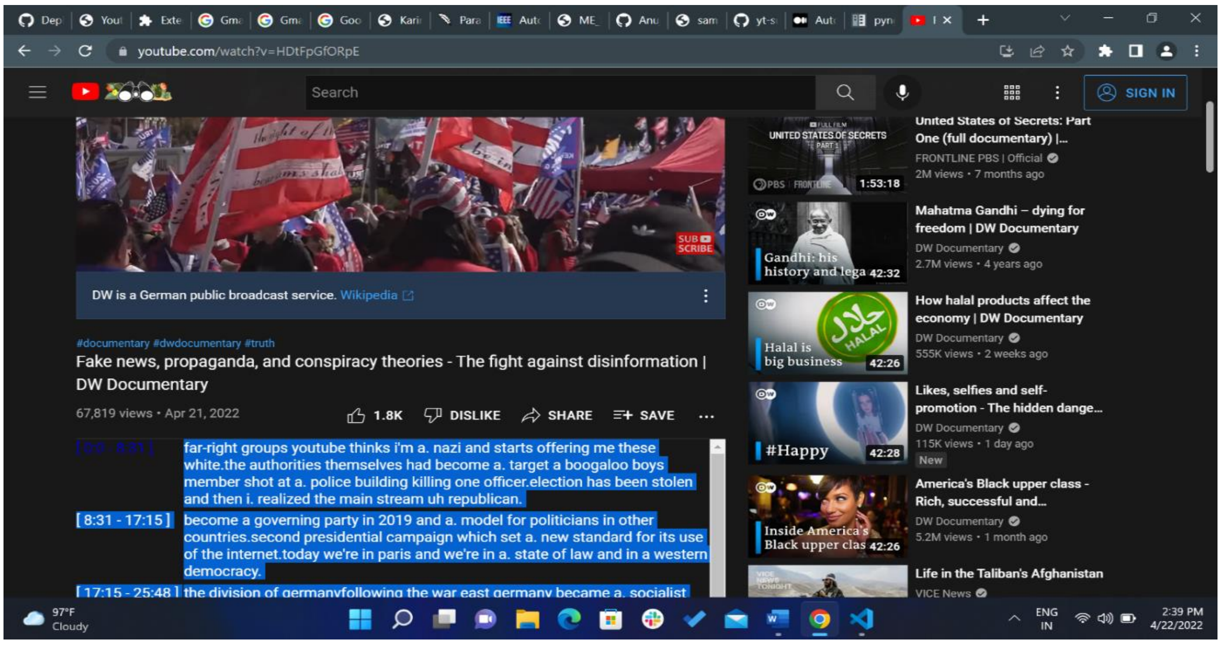
CHAPTER 8
RESULT AND ANALYSIS

8.1 Output

After the execution of the system, the system provides the output of the summarized text along with the time stamp of the text in the video, which upon clicking will take us to the particular part of the video. It produces the following.

- Summarized text of the video transcript beneath the video.
- Timestamp pertaining to the transcript.

Figure 8.1 below shows the output derived after performing text summarization on the transcripts of the requested video.


youtube.com/watch?v=HDtFpGfORpE

DW is a German public broadcast service. Wikipedia

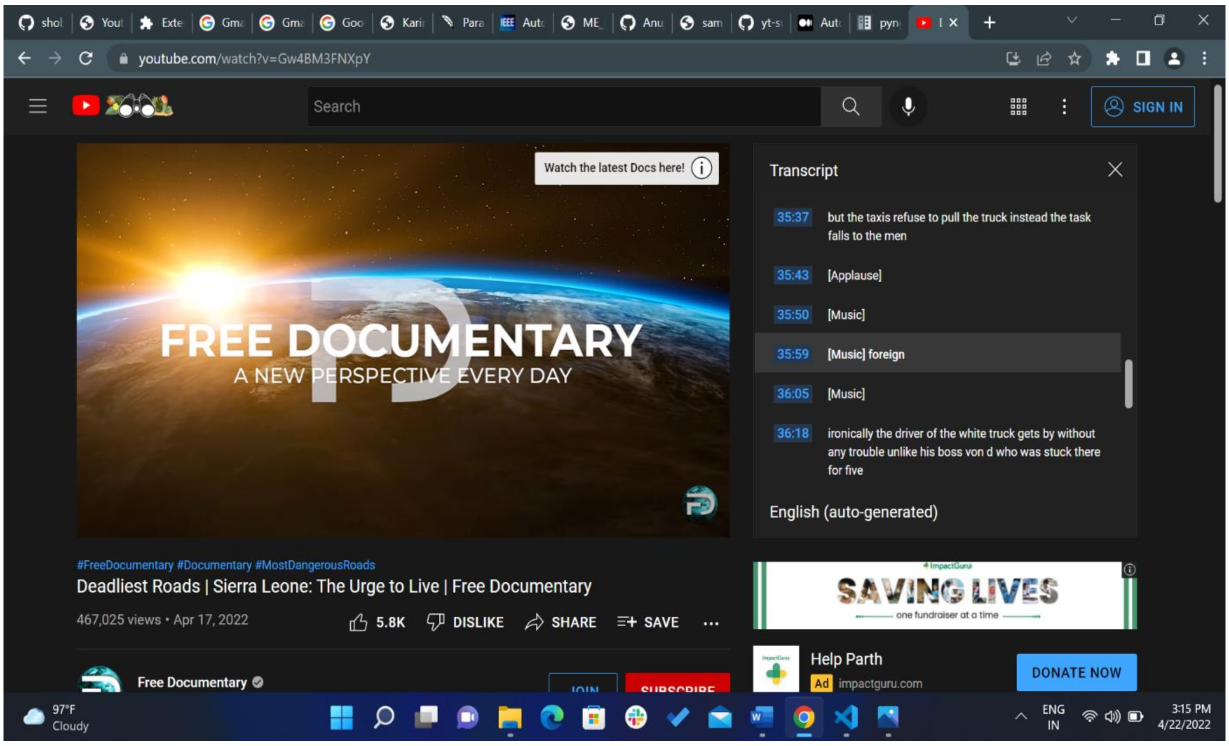
#documentary #dwdocumentary #truth
Fake news, propaganda, and conspiracy theories - The fight against disinformation | DW Documentary
67,819 views • Apr 21, 2022

[0:0 - 8:31] far-right groups youtube thinks i'm a. nazi and starts offering me these white.the authorities themselves had become a. target a boogaloo boys member shot at a. police building killing one officer.election has been stolen and then i. realized the main stream uh republican.
[8:31 - 17:15] become a governing party in 2019 and a. model for politicians in other countries.second presidential campaign which set a. new standard for its use of the internet.today we're in paris and we're in a. state of law and in a western democracy.
[17:15 - 25:48] the division of germanyfollowing the war east germany became a. socialist

United States of Secrets: Part One (full documentary) |... FRONTLINE PBS | Official 2M views • 7 months ago
Mahatma Gandhi – dying for freedom | DW Documentary DW Documentary 2.7M views • 4 years ago
How halal products affect the economy | DW Documentary DW Documentary 555K views • 2 weeks ago
Likes, selfies and self-promotion - The hidden dange... DW Documentary 115K views • 1 day ago New
America's Black upper class - Rich, successful and... DW Documentary 5.2M views • 1 month ago
Life in the Taliban's Afghanistan VICE News

Figure 8.1 – Summary of the video as output.

Figure 8.2 Shows the transcripts of a video before performing text summarization. The top right of the figure shows the transcripts.


youtube.com/watch?v=Gw4BM3FNXpY
Search
SIGN IN
Watch the latest Docs here!
FREE DOCUMENTARY
A NEW PERSPECTIVE EVERY DAY
#FreeDocumentary #Documentary #MostDangerousRoads
Deadliest Roads | Sierra Leone: The Urge to Live | Free Documentary
467,025 views • Apr 17, 2022
5.8K
DISLIKE
SHARE
SAVE
Free Documentary
SAVING LIVES
one fundraiser at a time
Help Parth
Ad impactguru.com
DONATE NOW
97°F
Cloudy
ENG IN
3:15 PM 4/22/2022

Transcript

35:37 but the taxis refuse to pull the truck instead the task falls to the men

35:43 [Applause]

35:50 [Music]

35:59 [Music] foreign

36:05 [Music]

36:18 ironically the driver of the white truck gets by without any trouble unlike his boss von d who was stuck there for five

English (auto-generated)

Figure 8.2 – Transcripts without summarization.

8.2 Analysis

The project has reached its conclusion and the results were driven as expected from the proposed system design in the initial stage of the project. The project includes performing various text summarization tasks such as performing abstractive summarization using sumy with help of tokenizers.

Text summarization is typically approached as a supervised machine learning issue in NLP (where future outcomes are predicted based on provided data).

The following is a typical example of how an extraction-based technique to text summarization can work:

1. Present a method for extracting the necessary keys from the source document. To find the essential words, you can utilise part-of-speech tagging, word sequences, or other linguistic patterns.

2. Collect text documents with key words that are positively labelled. The keys must be compatible with the extraction method specified. You can also build negatively labelled keys to improve accuracy.

3. To make the text summary, train a binary machine learning classifier. Following features can be included: -

- Length of the key phrase
- Frequency of the key phrase
- The most recurring word in the key phrase
- Number of characters in the key phrase

4. Finally, in the test phrase, compose all the important words and sentences and classify them.

8.3 Overview of Results and Analysis

This chapter captures the key accomplishments of the proposed system and summarizes the result of the project. It helps to interpret the final output and analyzes the core features. It also measures the derived solution and elucidates the obtained values. The chapter proves vital for the project as it justifies the ideation, planning and execution process of the project.

CHAPTER 9
CONCLUSION

Started with the motive of contributing to the applications of text summarization, the system from the start thrived to stick to its initial goal and in the end, was successfully implemented. Even though the project has been completed, the journey doesn’t end here, the project has a huge scope of expansion which can augment the growth of the delivered system. The scopes and enhancements of the project are proposed in the following subchapters.

9.1 Conclusion

Set out to look into the field of text summarizing and choose one or more methods for a prototype. The prototype should be able to generate legible summaries of video transcripts automatically. Furthermore, the quality of the generated summaries should be assessed in a uniform manner. Therefore, we can come to the conclusion that the set-out project, “Video Transcript Summarizer” has been successfully planned, proposed and implemented.
The project's goal of implementing a prototype summarizer has been met, although the summarizer's usefulness could be improved with future code enhancements. Furthermore, some of the more complex approaches, some of which rely on external lexical sources, offer an intriguing approach to improving the quality of the implemented summarizer.

9.2 Future Enhancement

The most exciting part comes in later in this project, as the proposed system has a wide scope in scaling and enhancing the system to become a more active system, thereby enriching the features of the set-out project. The first enhancement comes in terms of exception handling. Having only the basic set up done; this brings room for exception handling in the scenarios where the URL of the video is incorrect. This scenario might not occur with the chrome extension but can raise if used independently. Some of the exceptions can be if the URL is incorrect or is a live video. Summarization should be improved if the content can be divided down into smaller chunks and then summarized, with time stamps. Sumy is a good option.

The transcript is now only functional for YouTube videos with subtitles; audio and text processing form the beat enhancement.

A summary of 10-15 minutes can be presented as time stamps, and then a person can search for a term when it is discussed and go immediately to that point in the video.

9.3 Summary

This chapter, in detail, draws out the conclusion of the project upon the implementation of the proposed system in the beginning. It briefs out the completion of the project in regard to its vision and abstract and in the later part talks about the future of the same system, on how it can be enhanced for a higher level of its scope.

CHAPTER 10
TEAM REPORT

10.1 Objective

I, Manikanta Reddy K, being the sole member, orchestrated the flow of the system design and oversaw the aspects of technical eligibility for each step of the project evaluated and explored the various ongoing projects and research materials supporting the project.

Time extension caused by hurdles in various stages of the project were consolidated and conserved. Conceptualization and defining the features of the system with ideations is done.

Also, tracked each stage of the process making sure it aligns with the proposed and set vision. Insightful research material throughout the project were also in the scout.

10.2 Overview of the Team Report

This chapter abridges the accomplishments by navigating the path they embarked on whose core objective was to plan and organize separate goals and focus on achieving them with minimal deviation.

REFERENCES

[1] Barzilay, R., Elhadad, M.: Using lexical chains for text summarization. In: Proceedings of the ACL/EACL’97 Workshop on Intelligent Scalable Text Summarization, Madrid, pp. 10–17 (1997).

[2] Bossard, A., Généreux, M., Poibeau, T.: Cbseas, a summarization system – integration of opinion mining techniques to summarize blogs. In: Proceedings of the 12th Meeting of the European Association for Computational Linguistics (system demonstration), EACL ’09, Athens. Association for Computational Linguistics, Stroudsburg (2009).

[3] DeJong, G.: An overview of the FRUMP system. In: Lehnert, W., Ringle, M. (eds.) Strategies for Natural Language Processing, pp. 149–176. Lawrence Erlbaum Associates, Hillsdale.

[4] Amigó E, Gonzalo J, Penas A, Verdejo F (2005) QARLA: a framework for the evaluation of text summarization systems. In: ACL ’05: proceedings of the 43rd annual meeting on association for computational linguistics, pp 280–289.

[5] Banerjee S Mitra P, Sugiyama K (2015) Multi-document abstractive summarization using ILP based multi-sentence compression. In: Proceedings of the 24th international joint conference on artificial intelligence (IJCAI 2015), pp 1208–1214.

[6] Building RESTFUL APIs

https://atmamani.github.io/blog/building-restful-apis-with-flask-in-python/
Accessed on:28/01/2022

[7] Flask Reference

https://github.com/AnujK2901/yt-sum-flask

Accessed on: 12/02/2022

[8] Creating Chrome extension

https://medium.com/coding-in-simple-english/how-to-create-chrome-extension-7dd396e884ef

Accessed on:03/03/2022

[9] Installing unpacked extension

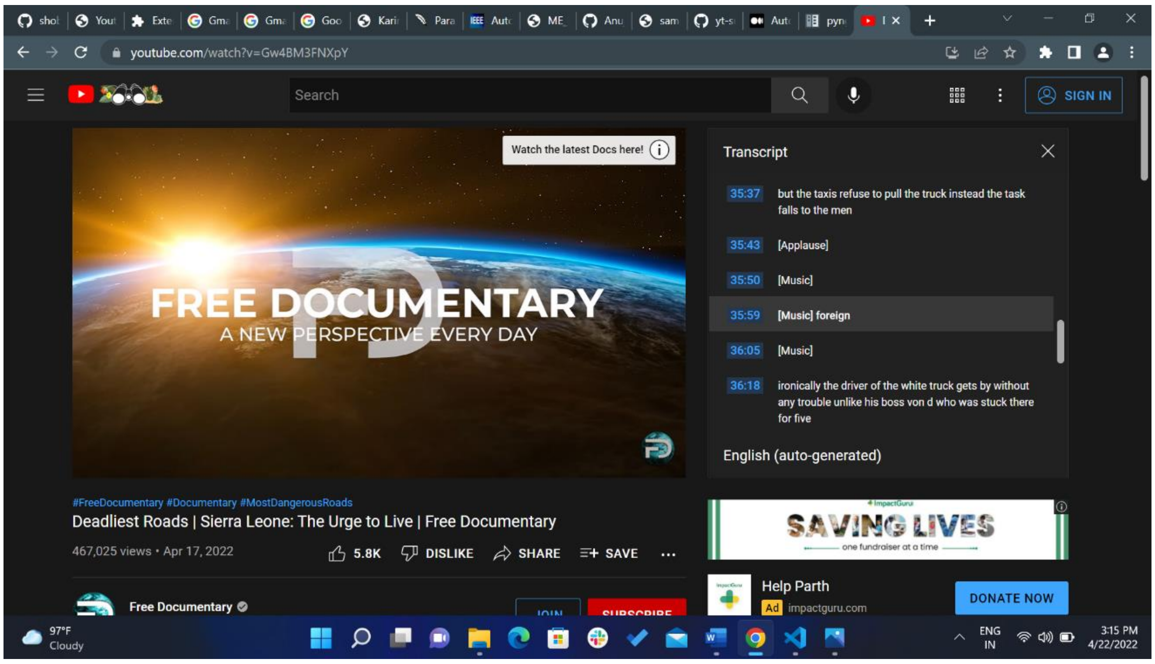
https://webkul.com/blog/how-to-install-the-unpacked-extension-in-chrome/
Accessed on: 12/04/2022

[10] UI_for_extension

www.Crio.do

Accessed on: 06/03/2021

APPENDIX A: SAMPLE SCREEN SHOTS


youtube.com/watch?v=Gw4BM3FNXpY

Transcript

35:37 but the taxis refuse to pull the truck instead the task falls to the men

35:43 [Applause]

35:50 [Music]

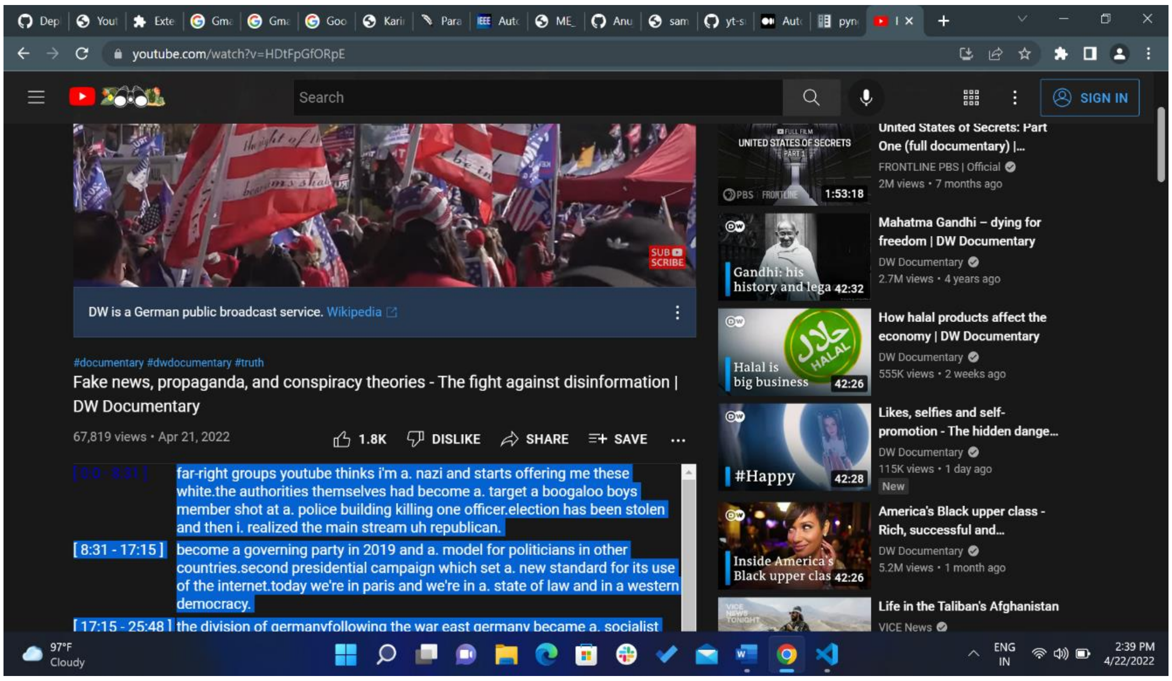
35:59 [Music] foreign

36:05 [Music]

36:18 ironically the driver of the white truck gets by without any trouble unlike his boss von d who was stuck there for five

English (auto-generated)

#FreeDocumentary #Documentary #MostDangerousRoads
Deadliest Roads | Sierra Leone: The Urge to Live | Free Documentary
467,025 views • Apr 17, 2022


youtube.com/watch?v=HDtFpGfORpE

DW is a German public broadcast service. Wikipedia

#documentary #dwdocumentary #truth
Fake news, propaganda, and conspiracy theories - The fight against disinformation | DW Documentary
67,819 views • Apr 21, 2022

[0:0 - 8:31] far-right groups youtube thinks i'm a. nazi and starts offering me these white.the authorities themselves had become a. target a boogaloo boys member shot at a. police building killing one officer.election has been stolen and then i. realized the main stream uh republican.

[8:31 - 17:15] become a governing party in 2019 and a. model for politicians in other countries.second presidential campaign which set a. new standard for its use of the internet.today we're in paris and we're in a. state of law and in a western democracy.

[17:15 - 25:48] the division of germanyfollowing the war east germany became a. socialist

United States of Secrets: Part One (full documentary) |...
FRONTLINE PBS | Official
2M views • 7 months ago
Mahatma Gandhi – dying for freedom | DW Documentary
DW Documentary
2.7M views • 4 years ago
How halal products affect the economy | DW Documentary
DW Documentary
555K views • 2 weeks ago
Likes, selfies and self-promotion - The hidden dange...
DW Documentary
115K views • 1 day ago
America's Black upper class - Rich, successful and...
DW Documentary
5.2M views • 1 month ago
Life in the Taliban's Afghanistan
VICE News
1.8K DISLIKE SHARE SAVE

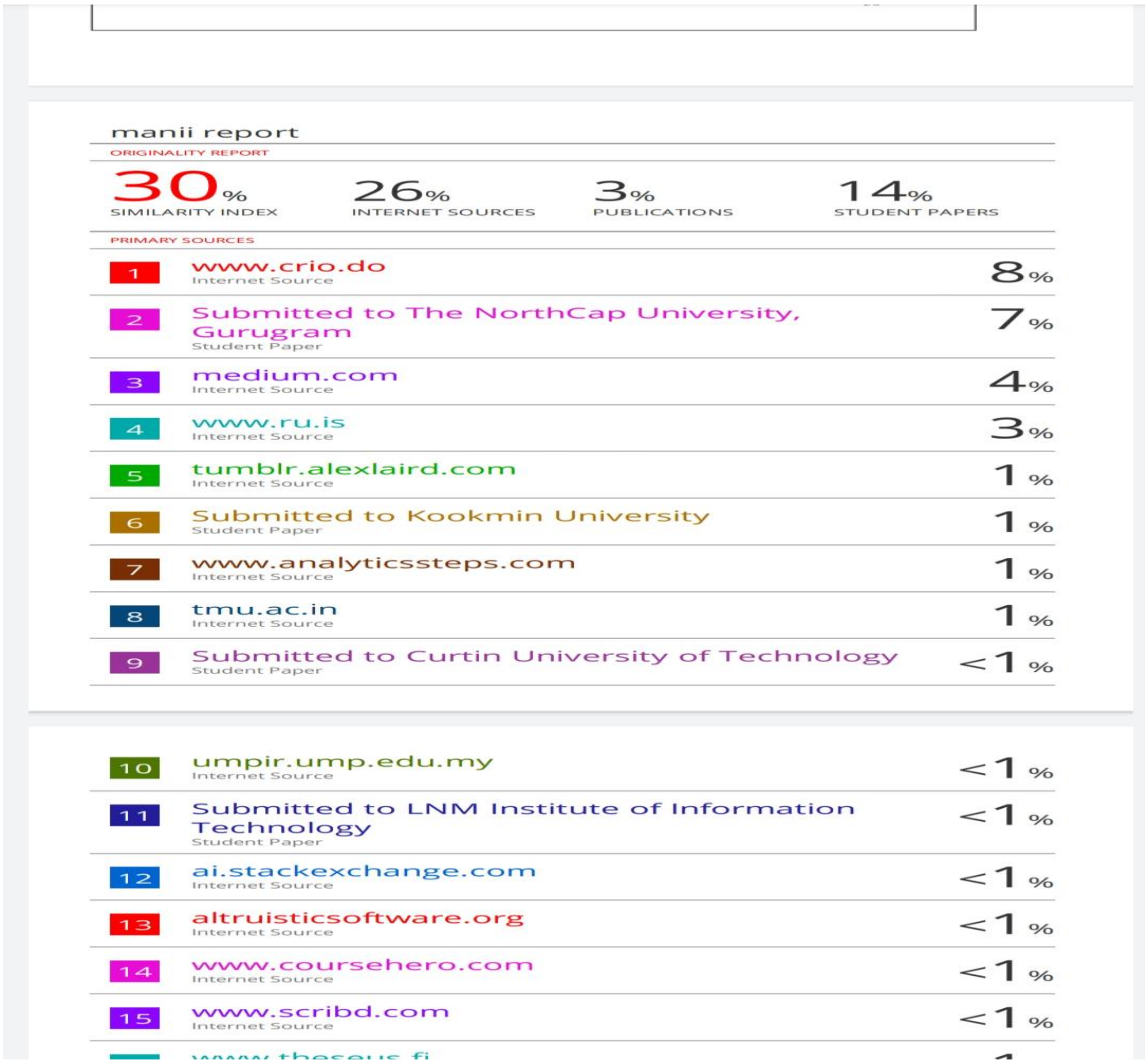
APPENDIX B: SAMPLE CODE

```
from __future__ import absolute_import
from __future__ import division, print_function, unicode_literals
from flask import Flask, jsonify
import datetime
from flask import request # used to parse payload
from youtube_transcript_api import YouTubeTranscriptApi
from youtube_transcript_api.formatters import TextFormatter
from flask import render_template
from flask import abort
from flask_cors import CORS
import os

# define a variable to hold you app
app = Flask(__name__)
CORS(app)

@app.route('/')
def hello():
return render_template('index.html')

@app.route('/time', methods=['GET'])
def get_time():
return str(datetime.datetime.now())

@app.route('/api/summarize', methods=['GET'])
def GetUrl():
"""
Called as /api/summarize?youtube_url='url'
"""
# if user sends payload to variable name, get it. Else empty string
video_url = request.args.get('youtube_url', '')
# if(len(video_url) == 0) or (not '=' in video_url):
#   print("f")
#   abort(404)

response = GetTranscript(video_url)
return jsonify(response)

def SumySummarize(text):

from sumy.parsers.html import HtmlParser
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.lsa import LsaSummarizer as Summarizer
from sumy.nlp.stemmers import Stemmer
from sumy.utils import get_stop_words

LANGUAGE = "english"
SENTENCES_COUNT = 3
import nltk;
nltk.download('punkt')

# url = "https://en.wikipedia.org/wiki/Automatic_summarization"
# parser = HtmlParser.from_url(url, Tokenizer(LANGUAGE))
# or for plain text files
# parser = PlaintextParser.from_file("document.txt", Tokenizer(LANGUAGE))
parser = PlaintextParser.from_string(text, Tokenizer(LANGUAGE))
stemmer = Stemmer(LANGUAGE)

summarizer = Summarizer(stemmer)
summarizer.stop_words = get_stop_words(LANGUAGE)
s = ""
for sentence in summarizer(parser.document, SENTENCES_COUNT):
s += (str)(sentence)
return s

def GetTextFromAudio():
import speech_recognition as sr
from pydub import AudioSegment

f = ""

# convert mp3 file to wav
for file in os.listdir(os.getcwd()):
if file.endswith(".mp3"):
f = file

if(len(f) == 0):
return f
sound = AudioSegment.from_mp3(f)

os.rename(os.path.join(os.getcwd(), f), os.path.join(os.getcwd(), "recordings", f))

sound.export("transcript.wav", format="wav")

# use the audio file as the audio source
AUDIO_FILE = "transcript.wav"
```

```
r = sr.Recognizer()
with sr.AudioFile(AUDIO_FILE) as source:
audio = r.record(source)  # read the entire audio file
return (r.recognize_google(audio))

def GetAudio(video_url):
from youtube_dl import YoutubeDL
ydl_opts = {
'format': 'bestaudio/best',
'postprocessors': [{
        'key': 'FFmpegExtractAudio',
        'preferredcodec': 'mp3'
}],
}
with YoutubeDL(ydl_opts) as ydl:
ydl.download([video_url])

def StringTime(time):
time = (int)(time)
return (str)(time // 60) + ":" + (str)(time % 60)

# video id are the last characters in the link of youtube video
def GetTranscript(video_url):
text = ""
try:
video_id = (video_url.split('=')[1]).split("&")[0]
transcript = YouTubeTranscriptApi.get_transcript(video_id)
duration = max(30, transcript[-1]['start'] // 5)
i, end, st = 0, 0, 0
text, ps_text = "", ""
summary_content = []
while(i < len(transcript)):
if(end - st < duration):
end = transcript[i]['start'] + transcript[i]['duration']
ps_text += transcript[i]['text']
ps_text += ". "
else:
# text += "[ " + StringTime(st) + " - " + StringTime(end) + "] " + SumySummarize(ps_text) +
"\n\n"
summary_content.append({"start": StringTime(st), "end": StringTime(end), "text":
SumySummarize(ps_text)})
st = end
end = transcript[i]['start'] + transcript[i]['duration']
ps_text = transcript[i]['text']
```

```
i += 1
summary_content.append({"start": StringTime(st), "end": StringTime(end), "text": SumySummarize(ps_text)})
# text += "[ " + StringTime(st) + " - " + StringTime(end) + "] " + SumySummarize(ps_text) + "\n\n"
return summary_content
except Exception as e:
# GetAudio(video_url)
# text = GetTextFromAudio()
# print('The text is: ', text)
return [{"start":StringTime(0), "end":StringTime(0), "text": str(e)}]

# server the app when this file is run
if __name__ == '__main__':
app.run()
```

APPENDIX C: PLAGIARISM REPORT



manii report

ORIGINALITY REPORT

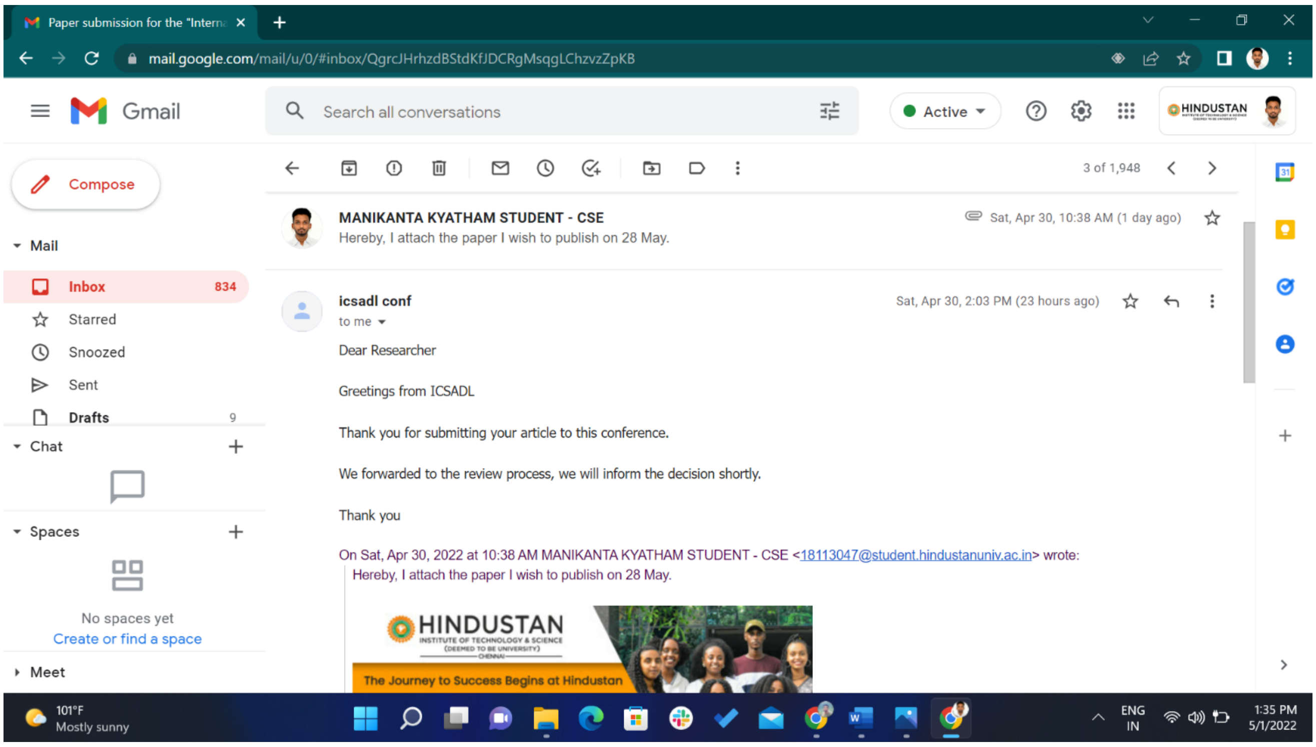
30%	26%	3%	14%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.crio.do Internet Source	8%
2	Submitted to The NorthCap University, Gurugram Student Paper	7%
3	medium.com Internet Source	4%
4	www.ru.is Internet Source	3%
5	tumblr.alexlaird.com Internet Source	1%
6	Submitted to Kookmin University Student Paper	1%
7	www.analyticssteps.com Internet Source	1%
8	tmu.ac.in Internet Source	1%
9	Submitted to Curtin University of Technology Student Paper	<1%
10	umpir.ump.edu.my Internet Source	<1%
11	Submitted to LNM Institute of Information Technology Student Paper	<1%
12	ai.stackexchange.com Internet Source	<1%
13	altruisticsoftware.org Internet Source	<1%
14	www.coursehero.com Internet Source	<1%
15	www.scribd.com Internet Source	<1%

APPENDIX D: PUBLICATION STATUS

The publication request has been communicated to "International Conference on Computer Science and Artificial Intelligence" and the paper submitted is sent to the review panel.



MANIKANTA KYATHAM STUDENT - CSE
Hereby, I attach the paper I wish to publish on 28 May.
Sat, Apr 30, 10:38 AM (1 day ago)

icsadl conf
to me
Sat, Apr 30, 2:03 PM (23 hours ago)

Dear Researcher

Greetings from ICSADL

Thank you for submitting your article to this conference.

We forwarded to the review process, we will inform the decision shortly.

Thank you

On Sat, Apr 30, 2022 at 10:38 AM MANIKANTA KYATHAM STUDENT - CSE <18113047@student.hindustanuniv.ac.in> wrote:
Hereby, I attach the paper I wish to publish on 28 May.

APPENDIX E: TEAM DETAILS

Team 19

Manikanta Reddy K,

18113047,

CSE 8A.