Project Report on

YouTube Transcript Summarizer

B.Tech(CSE)

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of : Ms. Indra Kumari

Submitted by : Yash Raj Singh (19SCSE1010564)

Anant Tyagi (19SCSE1010263)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE’S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **“CAPS….”** in partial fulfillment of the requirements for the award of the B. tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of Mr. Deependra Rastogi Designation, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Yash Raj Singh 19SCS1010564

Anant Tyagi 19SCSE1010263

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Ms. Indra Kumari

Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Yash Raj Singh:19SCSE1010564 & Anant Tyagi:19SCSE1010263 has been held on______and his/her work is recommended for the award of B.tech.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator

Date: December,

2021 *Place: Greater*

Noida

Table of Content

List of Figures

METHODOLOGY

First, we need to get the subtitles or transcript for a given Youtube video id by using the python API known as youtube_transcript_api. Since there are three types of transcript that we can extract - manually generated transcript, automatically generated transcript, and the videos that contain no transcript. We are not considering videos that do not have transcripts.

Secondly, when we get the transcript of a given Youtube video since it does not contain any punctuations like comma(,), full stops(.) which is very important for us in finding the boundaries of a sentence, so we will restore punctuations from our extracted transcript by using the python library known as “punctuator”. Now we will apply the text preprocessing methods to clean the extracted transcript by tokenizing the sentences as well as the words, lowercasing it, removing stop words like a, an, the, etc, removing punctuations, and stemming or lemmatization to generate the root form of inflected words. Performing text summarization: This task consists of shortening a large form of text into a precise summary that keeps all the necessary information intact and preserves the overall meaning. For this purpose in NLP for text summarization, there are two types of methods used:

Extractive Summarization: In this type of text summarization, the output is only the important phrases and sentences that the model identifies from the original text.

For the purpose of extractive summarization, we have used the TF-IDF model with Text Rank Algorithm.

TF-IDF(Term Frequency - Inverse Document Frequency)

After the cleaning process, we have to convert the words into it’s vectorized form so that our algorithm will process it by using TF-IDF. This is a technique to measure the quantity of a word in documents, we compute a weight to each word which signifies the importance of the word in the document and corpus. TF(Term Frequency): TF calculates the frequency of a word in a document. TF = No. of repetition of the word in the sentence / No. of words in a sentence IDF(Inverse Document Frequency): IDF is the inverse of the document frequency which measures the informativeness of term t. IDF = log(No. of sentences / No. of sentences containing words) After this, we will multiply both matrices to obtain the vectorized form which tells us which words are the most important.

Literature Survey

The Field of NLP is a field under artificial intelligence, which is used to classify, and process text with greater efficiency and accuracy than humans. While NLP as a technology can be a great technology for the love of many businesses, NLP APIs are an easy way for companies, organization, group or a single human to integrate technology into business and many more processes. Integrating NLP APIs with existing business plan, software-assisted companies to increase the efficiency and effectiveness of business processes.

Its all about teaching machines to understand how to understand human languages and extract meaning from text. It provides developers with extensive collection of NLP tools and libraries that enables developers to handle a great number of NLP related tasks such as document classification, topic modeling, part of speech (POS) tagging, word vectors, and sentiment analysis.

Modern technology can analyze more details about language than humans, without exhaustion and in a consistent, impartial way. Given the staggering amount of informal data generated on a daily basis, from medical records to social media platforms, automation will be essential to fully analyze text and speech data effectively.

Human language is amazingly complex and diverse. We express ourselves in endless ways, verbally and in writing. Not only are there thousands of languages and dialects, but within each language there is a different set of grammar and syntax rules, rules and slang. When we write, we often mispronounce words or abbreviate words, or leave punctuation marks. When we speak, we have ways of speaking in the region, and we mix, help and borrow words in other languages.

While both supervised and supervised learning, and especially in-depth reading, are now widely used in modeling the human language, there is also a need for a well-crafted understanding and background technology that is not present in these electronic learning methods.

YouTube Transcript Summarizer

Abstract

Integrated video data presentations may allow active video browsing. Such presentations provide the user with information about the content of a particular sequence being tested while maintaining an important message. We suggest how to automatically make video summaries for longer videos. Our video access method involves two tasks: first, splitting the video into smaller, compatible parts and second, setting the levels into effects. Our proposed algorithm sections are based on analysis of word frequency in speech transcripts. After that the summary is made by selecting the parts with the highest scores depending on the length of time and these are illustrated. We created and conducted a user study to check the quality of the summaries made. Comparisons are made using our proposed algorithm and a random segment selection scheme based on mathematical analysis of user learning outcomes. Finally, we can see the summarized context of the video we want to know about.

Summarization of the video is done by the Python API and NLP (Natural Language Processing). An API, or Application Programming Interface, is a server you can use to receive and send data using code. APIs are widely used to retrieve data, and that will be the focus of this first study.

When we want to receive data from an API, we need to make a request. Applications are used across the web.

Overview

Objective

In this project, you will be creating a Chrome Extension that will apply to the backend REST API where it will do NLP and respond with a summary of YouTube text.

Project Context

A large number of video recordings are made and shared online all day. It is very difficult to spend time watching such videos which may be longer than expected and sometimes our efforts may be in vain if we do not get the right information about it. Summarize the text of those videos automatically allows us to quickly look at important patterns in the video and helps us save time and effort in all the content of the video.

This project will provide us with the opportunity to experience technical expertise in the NLP state of the art to summarize the unseen text and use an exciting concept suitable for consultants and a refreshing professional project.
The summarizer is a Chrome extension that works with YouTube to extract the key points of a video and make them accessible to the user. The summary is customizable per user's request, allowing varying extents of summarization. Key points from the summarization process, together with corresponding time-stamps, are then presented to the user through a small UI next to the video feed. This allows the user to navigate to more important sections of the video, to get to the key points more efficiently.

Text Summarization Techniques

Most methods for video summarization do not make use of one of the most important sources of information in a video sequence, the spoken text or the natural-language content. Informedia, CueVideo and the system proposed in23 are some exceptions. Content text is readily available for most cable TV programs in the form of closed captions. For sequences like seminars and instructional programs where this information is not available speech recognition may be performed on audio to obtain the transcript. Once the text corresponding to a video sequence is available, one can use methods of text summarization to obtain a text summary. The portions of the video corresponding to the selected text may then be concatenated to generate the video skim. The techniques used in text summarization may be roughly divided into two groups:

• Statistical analysis based on information-retrieval techniques. In this approach, the problem of summarization is reduced to the problem of ranking sentences or paragraphs in the given text according to their likelihood of being included in the final summary. In these techniques, instead of employing natural language understanding methods, various features are extracted from the text which were shown to be correlated with the “abstractworthiness” of a sentence, and the ranking is done using a combination of these features.

• Natural Language Processing (NLP) analysis based on information-extraction techniques. This paradigm, making use of techniques from artificial intelligence, entails performing a detailed semantic analysis of the source text to build a source representation designed for a particular application. Then a summary representation is formed using this source representation and the output summary text is synthesized.24 Methods using statistical processing to extract sentences for the summary often generate summaries that lack coherence. These methods also suffer from the dangling anaphor problem. Anaphors are pronouns, demonstratives, and comparatives like “he”, “this”, and “more”, which can only be understood by referring to an antecedent clause appearing before the sentence in which the these words occur. If the antecedent clause has not been selected for the summary, anaphors may be confusing for the user. Although techniques based on NLP generate better summaries, the knowledge base required for such systems is generally large and complex. Furthermore such systems are specific to a narrow domain of application and are hard to generalize to other domains.

Natural Language Processing (NLP)

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

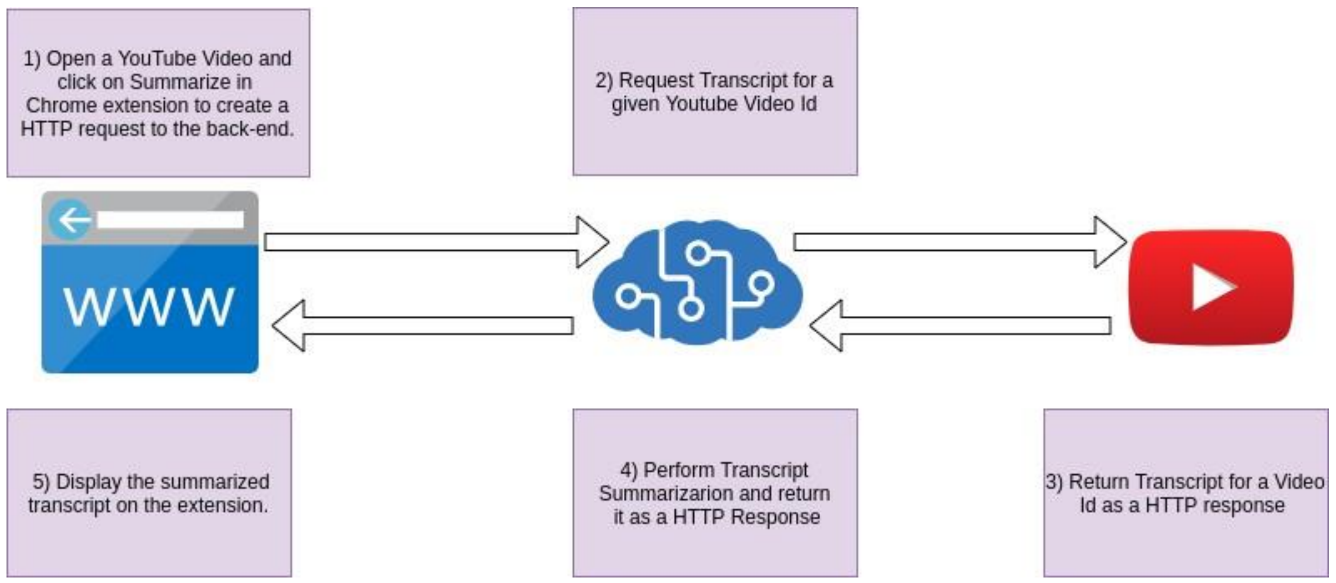
NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

NLP Tasks

- **Speech recognition**, also called speech-to-text, is the task of reliably converting voice data into text data.
- **Part of speech tagging**, also called grammatical tagging, is the process of determining the part of speech of a particular word or piece of text based on its use and context.
- **Word sense disambiguation** is the selection of the meaning of a word with multiple meanings through a process of semantic analysis that determine the word that makes the most sense in the given context.
- **Co-reference resolution** is the task of identifying if and when two words refer to the same entity.
- **Sentiment analysis** attempts to extract subjective qualities—attitudes, emotions, sarcasm, confusion, suspicion—from text.
- **Natural language generation** is sometimes described as the opposite of speech recognition or speech-to-text; it's the task of putting structured information into human language.

Project Stages

The project consists of the following stages:


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

Dig-1. Project Stages

High-Level Approach

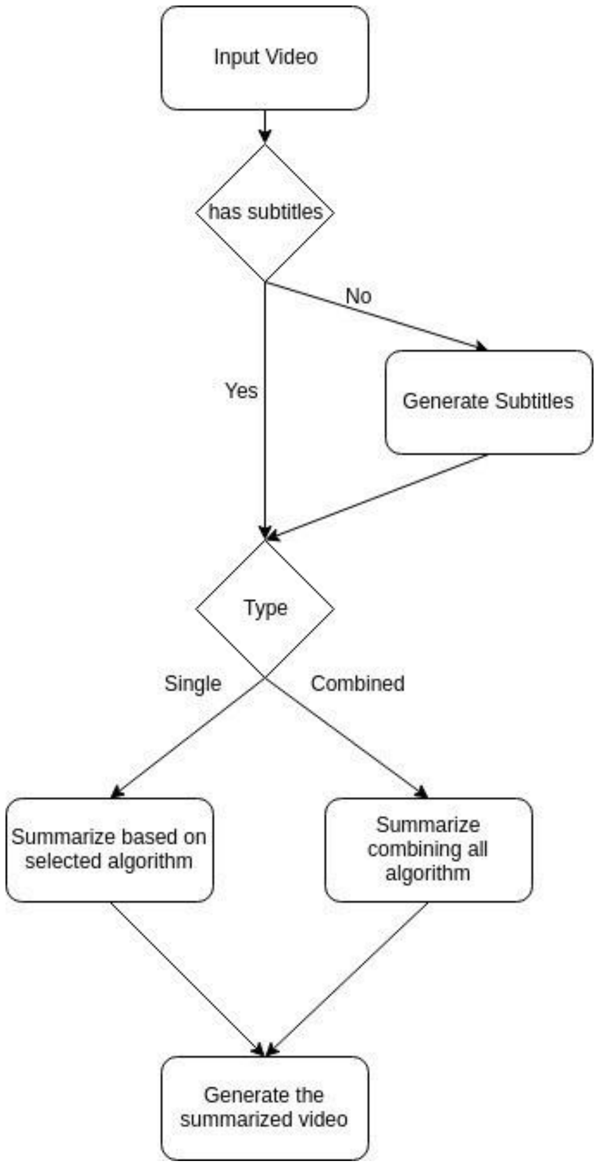
- Find the text / subtitle of YouTube Video Id that you use using the Python API.
- Make text summaries of text found using HuggingFace transformers.
- Create a Flask backend REST API to display client summarization service.
- Upgrade the chrome extension that will use the backend API to display user text.

Applications

- Meetings and video-conferencing - A system that could turn voice to text and generate summaries from your team meetings.

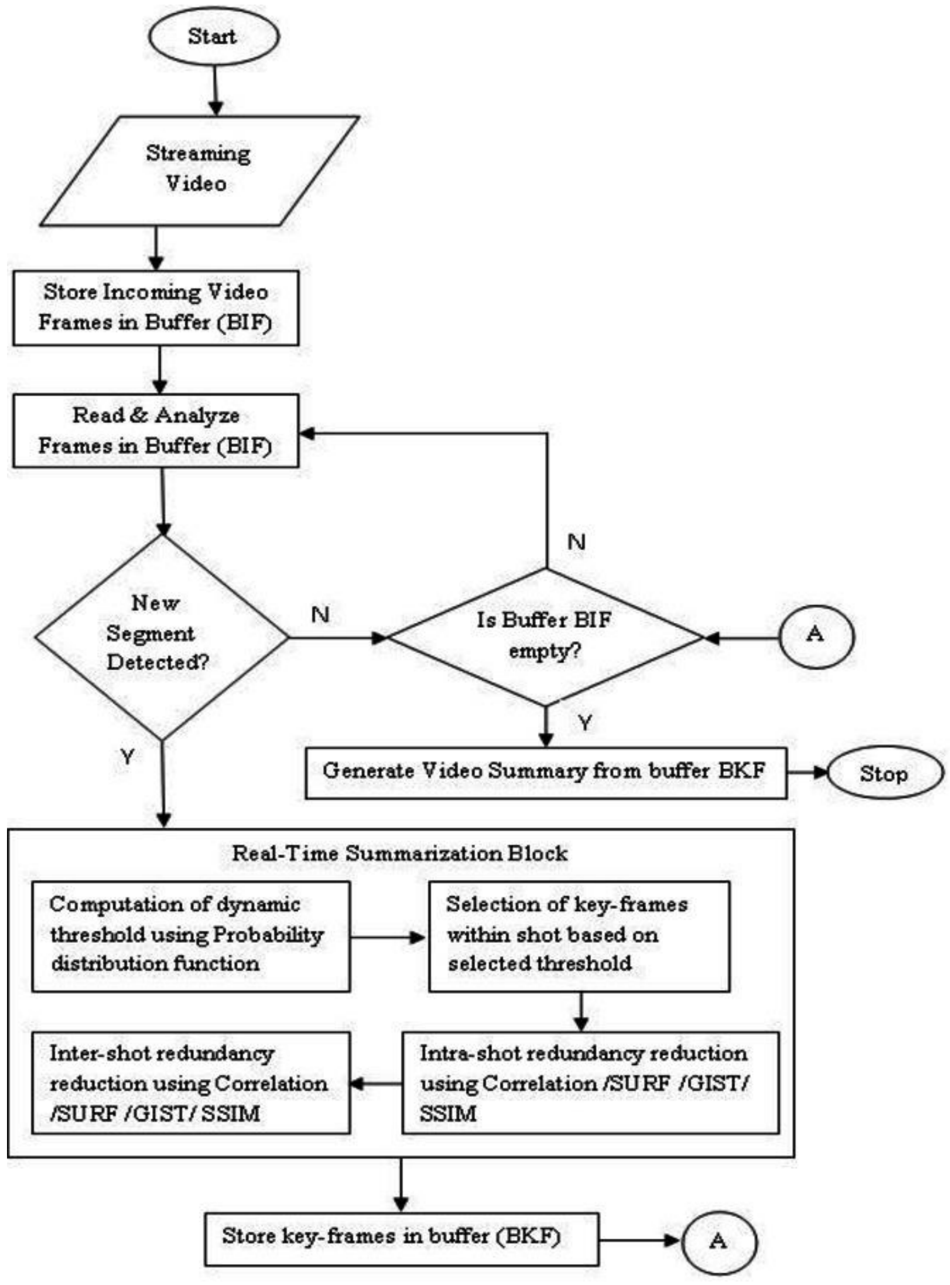
- Patent research - A summarizer to extract the most salient claims across patents.

Data Flow Diagram


Input Video
has subtitles
No
Yes
Generate Subtitles
Type
Single
Combined
Summarize based on selected algorithm
Summarize combining all algorithm
Generate the summarized video

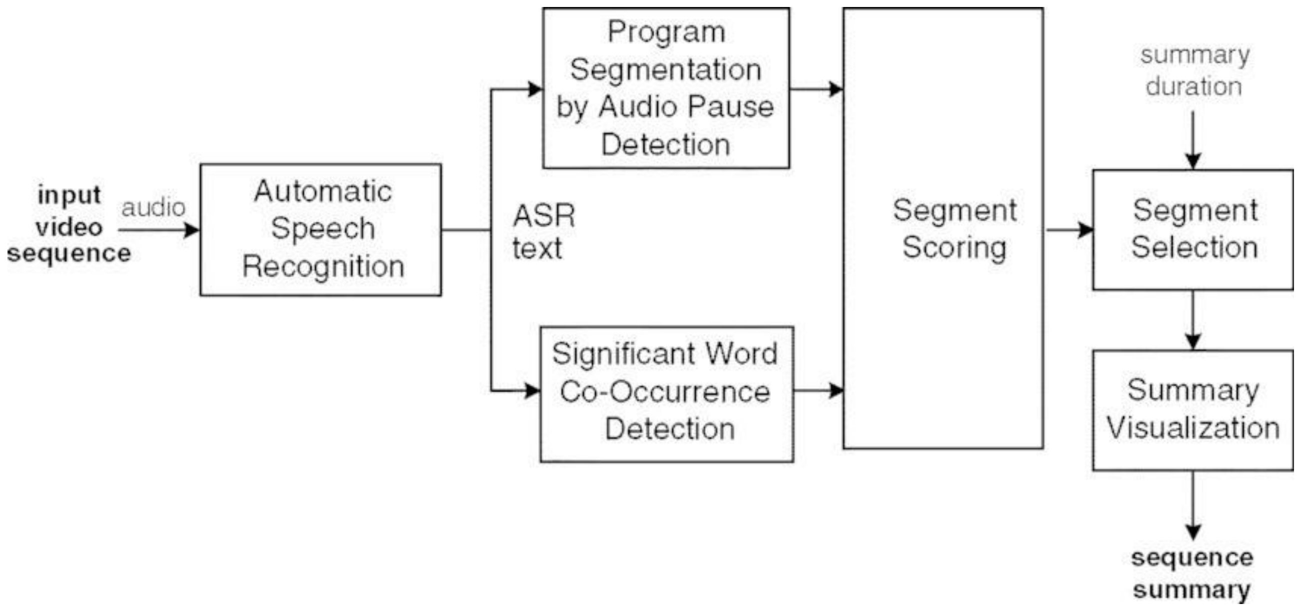
Dig-1 Data Flow Diagram

Flow chart diagram


Start
Streaming Video
Store Incoming Video Frames in Buffer (BIF)
Read & Analyze Frames in Buffer (BIF)
New Segment Detected?
N
Is Buffer BIF empty?
N
A
Y
Generate Video Summary from buffer BKF
Stop
Y
Real-Time Summarization Block
Computation of dynamic threshold using Probability distribution function
Selection of key-frames within shot based on selected threshold
Inter-shot redundancy reduction using Correlation /SURF /GIST/ SSIM
Intra-shot redundancy reduction using Correlation /SURF /GIST/ SSIM
Store key-frames in buffer (BKF)
A

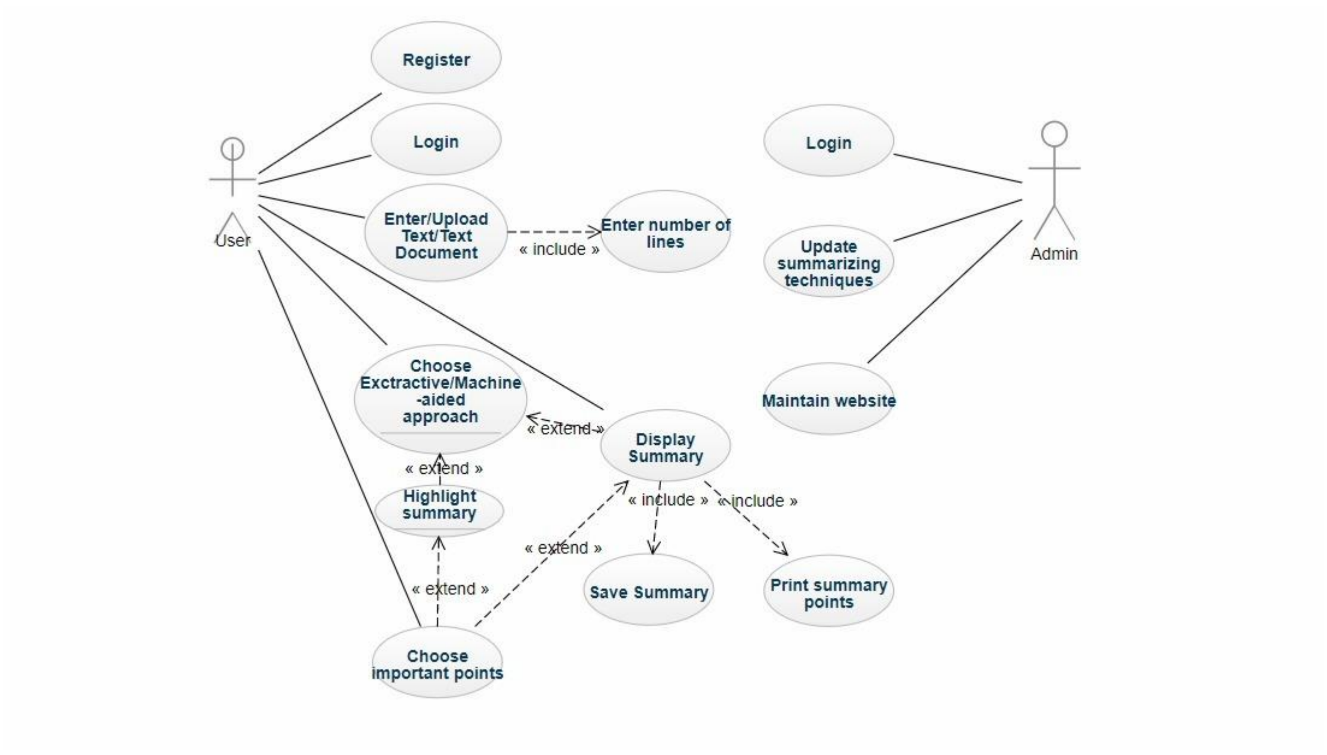
Dig-2 Flow chart

Architecture of the project



Dig-3 Architecture of the project

Use Case Diagram



Dig-3 Use Case Diagram

Task 1

Getting Started with the back-end

APIs changed the way we build applications, there are countless examples of APIs in the world, and many ways to structure or set up your APIs. In this milestone, we are going to see how to create a back-end application directory and structure it to work with the required files. We are going to isolate the back-end of the application to avoid conflicting dependencies from other parts of the project.

Requirements

• Create a backup application directory containing files named as app.py and needs.txt.

• Launch the app.py file with the basic Flask REST ful BoilerPlate with the tutorial link as mentioned in the reference section below.

• Create a new visible area with a pipe that will serve as a standalone (guide) where everything stays.

• Enable the newly created visual environment and apply the following dependencies using a pipe:

- Flask

- youtube_transcript_api

- converters [torch]

• Remove the cooling pipe and redirect the output to the requirements.txt file. This requirements.txt file is used to specify which python packages are required to run a project.

References

- Creating a Virtual Environment in Python
- Building RESTful APIs with Flask in Python BoilerPlate
- HuggingFace Transformer Python Installation

Expected Outcome

You are expected to initialize the back-end portion of your application with the required boiler plate as well as the dependencies.

Task 2

Get transcript for a given video

Have you ever wondered how to find the contents of your YouTube video? In this milestone, we are.

You will use the python API which allows you to access text / video footage provided by YouTube. It also works with auto-generated subtitles, supports subtitle translation, and does not require a headless browser like other solutions designed for Selenium!

Requirements

In app.py, - Create a function that will accept YouTube video id as the input parameter and retrieve the full transmitted text as output. - Feedback from the Transcript API will return a list of dictionaries that look like this:

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


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

- Combine data from feedback to retrieve text in all cable formats that look like this:

Hello, how are you ...

References

- YouTube Transcript API Documentation
- Read, Write and Parse JSON using Python

Expected Outcome

You should be able to download the script with the help of the work done which we will use later as the installation of the NLP processor feed into the pipeline.

Task 3

Perform Text Summarization

Text summarizing is the task of reducing text fragments into a concise summary that stores the content of key information and full meaning.

There are two different methods used to summarize the text:

• **Exclusive Summary**: This is where the model identifies key sentences and phrases from the original text and excludes only those.

• **Abstractive Abbreviation**: The model produces a completely different text shorter than the original, forming new sentences in a new way, like humans. In this project, we will use transformers in this way.

In this archive, we will use the HuggingFace library in Python to create an abstract text that was not found in the previous class.

Requirements

In app.py, - Create a function that will accept YouTube text as an input parameter and retrieve summarized text as output. - Establish token and model from test name. Summary is usually done using an encoder-decoder model, such as Bart or T5. - Explain the summary to be summarized. - Enter a specific T5 prefix “sum: “. - Use the PreTrainedModel.generate () method to make a summary.

References

- How to Perform Text Summarization using Transformers in Python
- Transformers official documentation

Note

• The Transformer model used for the above project can only take input text size up to 1024 words. Therefore, text size with more than 1024 words can throw Exception in relation to the length of the text you have been transferred to.

Expected Outcome

You should be able to ensure that the model produces a completely new abstract text that is different from the original text.

Task 4

Create REST API endpoint

The next step is to define the resources that will be exposed by this backend service. This is an extremely simple application, we only have a single endpoint, so our only resource will be the summarized text.

Requirements

In app.py,

● Create a Flask API Route via GET HTTP Request with URI http: // [hostname] / api / summary? youtube_url = <url>.

● Extract the YouTube video id from the YouTube URL found in the query parameters. - Generate summaries by making notes the production function follows the execution of the written summary function.

● Return a summary of the HTTP status OK and manage the HTTP variant if applicable.

● Launch the Flask app and check the storage location in Postman to confirm the appropriate results.

References

- Designing a RESTful API with Python and Flask
- Parsing REST API Payload and Query Parameters With Flask

Expected Outcome

You should be able to create a final point to summarize YouTube video documents and test the response with different video URLs.

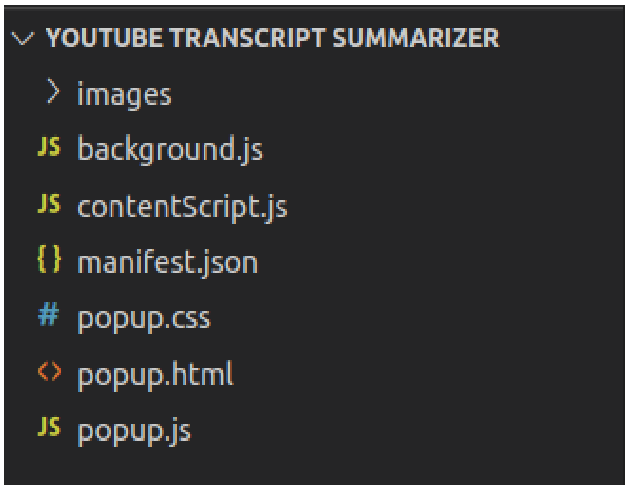
Task 5

Getting Started with Chrome Extension

Extensions are small software programs that customize the browsing experience. Enables users to customize Chrome functionality and behavior preferences. See built on web technologies such as HTML, CSS and JavaScript. In this epic, we are you will see how to create a recommended Chrome application guide and configure it to work with the required files.

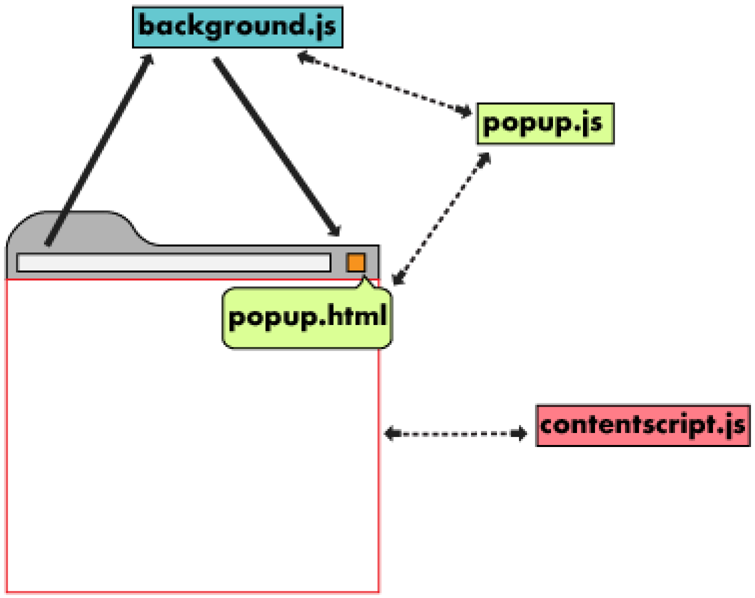
Requirements

- Create a chrome extension application directory containing essential files required as mentioned below.



```
YOUTUBE TRANSCRIPT SUMMARIZER
  > images
  background.js
  contentScript.js
  manifest.json
  popup.css
  popup.html
  popup.js
```

Dig-2. Extension Components



- The diagram below shows a brief contribution of each chrome extension design file. (Photo source: Coding in Simple English - Medium)

- Paste the following code snippet in the manifest.json.

```
{
    "manifest_version": 2,
    "name": "YSummarize",
    "description": "An extension to provide a summarized transcript of a
YouTube Subtitle eligible Video.",
    "version": "1.0",
    "permissions": ["activeTab"],


}
```

What did you guess? We already have enough to load our extension in the browser:

- Just go to chrome: // extensions and open the developer mode from the top right corner.

- Then click on Upload without downloading and select the folder containing the newly created expression file.

- When you have it, our extension is active.

References

- Check out Crio’s HTML and CSS byte to get yourself fully equipped with HTML/CSS.
- The Ultimate Guide to Building a Chrome Extension
- How to Create Chrome Extensions

Note

You will need to reload the extension every time we make changes to the extension.

Expected Outcome

You should be able to create a recommended Chrome extension application directory and configure it to work with the required files.

Task 6

Build a User Interface for Extension Popup

We need a user interface so that the user can interact with the popups which are one of several types of user interface that a Chrome extension can provide. They usually appear upon clicking the extension icon in the browser toolbar.

Requirements

- Insert a line at the bottom of the page action in the expression file that enables the user's evolutionary interface.

```
{
 .
 .
 .
 "page_action": {
        "default_popup": "popup.html",
 }
 .
 .
}
```

• In a popup.html file,

- Insert popup.css file to make styles available in HTML elements.

- Insert a popup.js file to enable user interaction and behavior with HTML objects.

- Insert a button object with an abbreviated name that when clicked you will pull out a click event that the listener of the event will get to respond to.

- Enter a div item where the summary text will be displayed when received from the backend REST API Call.

• In a popup.css file,

- Provide the appropriate CSS style on the HTML and div elements button for a better user experience.

References

- Design the user interface
- What is page_action in a manifest file.

Expected Outcome

The visual interface of the user should be objective and minimal and should improve the browsing experience without interruption to it.

Task 7

Display Summarized transcript

We have provided a basic UI to enable users to share and display a summary text but there are missing links that need to be fixed. In this archive, we will add functionality to allow the extension to connect to the recurring server using HTTP REST
API calls.

Requirements

In popup.js,

- When the DOM is ready, paste the event listener by event type as "click" on the shortcut key and pass the second parameter as an anonymous callback function.

- For anonymous operation, send the action message generating using chrome. runtime. send Message method to introduce the contentScript.js to create a summary.

- Enter the event listener chrome. runtime. on Message to listen to the message from contentScript.js that will execute the retrieval function to extract Summary.

- In the callback function, display a summary of the div item systematically using JavaScript.

Underline the content script in the expression file that will include
excess script scripting scriptSj.js and create an automatic script on a particular page

```
{
 .
 .
 .
 "content_scripts":[
   {
     "matches":["https://www.youtube.com/watch?v=*"], "js":
     ["contentScript.js"]
   }
.
.
.
}
```

● In contentScript.js,

- Enter the event listener.runtime.onMessageto listen

a message generate that will issue a function of generationalSummarycallback.

- In reverse operation, extract the URL of the current tab and apply GET HTTP using the XMLHTTPRequestWeb API back to get a summary text in response.

- Send action message with short uploads

usingchrome.runtime.sendMessageto notifypopup.js` to display a summary text.

References

- Content Scripts
- Message Passing in Chrome
- How to use XMLHttpRequest to issue HTTP requests

Expected Outcome

The visual interface of the user should be able to display a summary text at the request of the user.

Task 8

Spice it up!

With basic use made all the way, for all cats who want to know out there, these are some of the linear things that can be used to spice up the existing performance.

[Note: This is not a mandatory event.]

Requirements

• Try the following:

- Can you add function to summarize longer texts using the summarization process (by e.g. Using the LSA process)?
- Can you add text summaries from non-English video and show it in English?
- Can you add functionality to adjust the maximum length of the summary text?
- Can you add functionality to support text summaries from video without subtitles?

References

- Extractive Text Summarization Techniques With sumy
- Language Translator Using Google API in Python
- How to download YouTube video as audio using python
- Transcribing audio files using python

Expected Outcome

You should be able to add some features to your app.

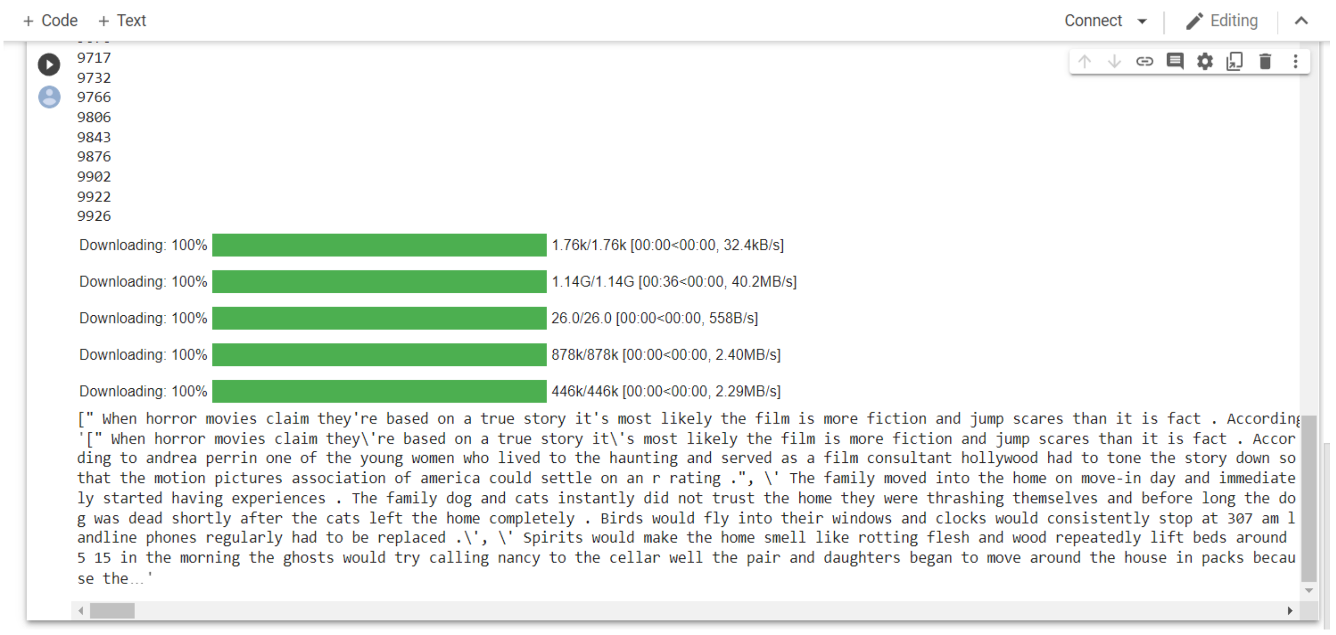
Other Approaches We Came Around While The Exploring Ours

There are also some video browsing approaches which may be used to visualize video content compactly and hence may be considered video summarization techniques. Taskiran et al.8 cluster keyframes extracted from shots using color, edge, and texture features and present them in a hierarchical fashion using a similarity pyramid. In their BMoViES system Vasconcelos and Lippman7 derive four semantic attributes from each shot and present these in a time line for the user to examine. In the CueVideo system, Srinivasan et al.3 provide a video browser with multiple synchronized views. It allows switching between different views, such as storyboards, salient animations, slide shows with fast or slow audio, and full video while preserving the corresponding point within the video between all different views. Ponceleon and Dieberger22 propose a grid, which they call the movieDNA, whose cells indicate the presence or absence of a feature of interest in a particular video segment. When the user moves the mouse over a cell, a window shows a representative frame and other metadata about that particular cluster. A system to build a hierarchical representation of video content is discussed in Huang et al.23 where audio, video, and text content are fused to obtain an index table for broadcast news.

Source Code

```
!pip install -q transformers
!pip install -q youtube_transcript_api

from transformers import pipeline
from youtube_transcript_api import YouTubeTranscriptApi

youtube_video = "https://www.youtube.com/watch?v=J4SUdqOntxQ"
video_id = youtube_video.split("=")[1]

YouTubeTranscriptApi.get_transcript(video_id)
transcript = YouTubeTranscriptApi.get_transcript(video_id)

transcript[0:5]

result = " "
for i in transcript:
    result += ' ' + i['text']
    print(len(result))

summarizer = pipeline('summarization')

num_iters = int(len(result)/1000)
summarized_text = []
for i in range(0, num_iters + 1):
    start = 0
    start = i*1000
    end = (i+1) * 1000
    out = summarizer(result[start:end])
    out = out[0]
    out = out['summary_text']
    summarized_text.append(out)

print(summarized_text)

str(summarized_text)
```

Output



```
9717
9732
9766
9806
9843
9876
9902
9922
9926
Downloading: 100% 1.76k/1.76k [00:00<00:00, 32.4kB/s]
Downloading: 100% 1.14G/1.14G [00:36<00:00, 40.2MB/s]
Downloading: 100% 26.0/26.0 [00:00<00:00, 558B/s]
Downloading: 100% 878k/878k [00:00<00:00, 2.40MB/s]
Downloading: 100% 446k/446k [00:00<00:00, 2.29MB/s]
[" When horror movies claim they're based on a true story it's most likely the film is more fiction and jump scares than it is fact . According
'[" When horror movies claim they\'re based on a true story it\'s most likely the film is more fiction and jump scares than it is fact . Accor
ding to andrea perrin one of the young women who lived to the haunting and served as a film consultant hollywood had to tone the story down so
that the motion pictures association of america could settle on an r rating .", \' The family moved into the home on move-in day and immediate
ly started having experiences . The family dog and cats instantly did not trust the home they were thrashing themselves and before long the do
g was dead shortly after the cats left the home completely . Birds would fly into their windows and clocks would consistently stop at 307 am l
andline phones regularly had to be replaced .\', \' Spirits would make the home smell like rotting flesh and wood repeatedly lift beds around
5 15 in the morning the ghosts would try calling nancy to the cellar well the pair and daughters began to move around the house in packs becau
se the...'
```

How We built it

We first set up a GitHub repo for project management and made a readme to keep track of dependencies and environment information. Our project utilized a Google Chrome extension for the frontend and a Django server for the backend, so we initially developed both parts of the project separately and integrated each service towards the end of the hackathon. We also had to choose which online APIs and services to use as our project progressed, and decided to use Punctuator, Resoomer, and YouTube closed captions as the key technologies involved in the timestamped summary generation.

Challenges We ran into

Out of the numerous challenges we encountered throughout this hackathon, the most significant challenge was an overestimation of the technologies available to us. Our overestimation of the ability of transcription services forced us to make compromises, while our attempts to get the YouTube player to control playback pressed us to create a hacky solution. Overcoming these challenges and bridging the capabilities of these technologies was an integral part of the project.

Inspiration

We spend a noticeable amount of our weekly time watching YouTube videos, be it for entertainment, education, or exploring our interests. In most cases, the overall intent is to obtain some form of information from the video. We were seeking a solution to increase the efficiency of this "information extraction" process as YouTube's speed adjustment option is the only relevant tool. And so we decided to develop YouTube Summarizer!

Conclusion

The increase in popularity of video content on the internet requires an efficient way of representing or managing the video. This can be done by representing the videos on the basis of their summary.

Learning how to set up web services using API, create Google Chrome extensions and implement the Cloud Computing. In addition, we used HTML and CSS to develop Web-Apps and write software packages in python.

We need to follow time management and fully grasp the difficulties which may occur and when to change the course of the project based to use our time more efficiently.

It is important to understand connections between different technology and to account for possible bugs when incorporating different software packages into the corpus of a final software product.

Reference

[1] Jiri Fajtl, Hajar Sadeghi Sokeh, Vasileios Argyriou , Dorothy Monekosso , and Paolo Remagnino: Summarizing Videos with Attention In: Proceedings of the ICLR. Vol. 5 (2019).

[2] Kaiyang Zhou, Yu Qiao, Tao Xiang : Deep Reinforcement Learning for Unsupervised Video Summarization In: Diversity-Representativeness Reward 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org).

[3] Mrigank Rochan, Linwei Ye, and Yang Wang: Video Summarization Using Fully Convolutional Sequence Networks In: International Conference on Learning Representations (2018).

[4] https://www.crio.do/projects/python-youtube-transcript/

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

[6] https://www.thepythoncode.com/article/text-summarization-using-huggingface-transformers-python

[7] Ferman A.M. and Tekalp A.M. Two-stage hierarchical video summary extraction to match low-level user browsing preerences. IEEE Trans. Multimedia, 5(2):244–256, 2003.

[8] P.Sushma, Dr.S.Nagaprasad, Dr. V. Ajantha Devi. Youtube: Bigdata Analytics using Hadoop and Map Reduce in International Journal of Engineering Research in Computer Science and Engineering (IJERCSE), vol 5, Issue 4, April 2018