

MINOR PROJECT REPORT ODD SEMESTER 2022

VIDEO SUMMARIZER



**Dept. of Computer Science and Engineering Jaypee
Institute of Information Technology,
Noida Sector 62**

**Faculty Supervisor
Dr. Bhawna Saxena**

Submitted By:

**NAVDEEP KAUR 20103187 B7
UNNATI JAIN 20103195 B7
MANYA GARG 20103197 B7**

TABLE OF CONTENT

1. DECLARATION
2. CERTIFICATE
3. ACKNOWLEDGEMENT
4. SUMMARY
5. INTRODUCTION
 - 5.1 GENERAL
 - 5.2 PROBLEM STATEMENT
 - 5.3 SIGNIFICANCE OF THE PROBLEM
 - 5.4 BRIEF DESCRIPTION OF THE SOLUTION APPROACH
6. ALGORITHM USED
 - 6.1 PROPOSED METHODOLOGY
7. REQUIREMENTS
 - 7.1 HARDWARE REQUIREMENTS
 - 7.2 SOFTWARE REQUIREMENTS
 - 7.3 LIBRARIES USED
8. DESIGN A DIAGRAM
 - 8.1 USE-CASE DIAGRAM
9. CODE AND OUTPUT
10. CONCLUSION
11. REFERENCES

DECLARATION

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other Institute of higher learning, except where due acknowledgment has been made in the text.

Place: IIIT NOIDA UP

Date: 3 December 2022

Enrollment Number: 201030187, 20103195 20103197

Name: NAVDEEP KAUR, UNNATI JAIN, MANYA GARG

CERTIFICATE

This is to certify that the work titled Video Summarizer submitted by Navdeep Kaur, Unnati Jain, and Manya Garg in partial fulfillment for the award of the B Tech degree of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor :

Name of Supervisor :

Dr. Bhawna Saxena

Date :

3 December 2022

ACKNOWLEDGEMENT

This entire project would not have been possible without the kind support and help of many individuals and organizations. We take this opportunity to express our profound sense of gratitude and appreciation to all those who helped us throughout the duration of this project.

We would like to express our special thanks to our mentor **Dr. Bhawna Saxena**, who gave us the golden opportunity to be a part of this wonderful project in the domain of Web Development and python and for providing guidance and expert supervision for this project. She has set a spectacular example for our young impressionable minds during creating this project. We are really thankful to her.

My thanks and appreciation also go to my colleagues in developing this project and the people who have willingly helped me out with their abilities. During the making of this project, we really learned a lot and got knowledge of new areas where we can work in the future. We express our sincere gratitude towards each and everyone who helped us towards the completion of the project.

SUMMARY

Video Summarizer is a tool used to generate a short summary of the content of a longer video document by selecting and presenting the most informative or interesting materials for potential users. It also converts the video into entire text which can be further used for text summarization. We have also implemented Live Audio Speech to text which can also be summarized further.

MOTIVATION BEHIND THIS PROJECT

Us being the covid batch, we had to watch long lectures and youtube videos for online studies. So we had to spend a lot of time on the videos. Keeping this in mind, and the problems we faced due to increased screen time, we have created a video summarizer to generate a summary of the video that we watch. This will save a lot of time for the students as now they won't have to watch the whole video, and instead can read the short summary generated.

Don't have the time to watch the whole video? Create a text summary of your video, so that you can easily get information out of it.

If you miss out on your google meets or important meetings due to any emergency, you can use this video summarizer to generate a summary of your video.

Some of the reasons for making an Automated Text Summarizer:

- Summaries reduce reading time.
- When researching documents, summaries make the selection process easier.
- Automatic summaries improve the effectiveness of knowledge.

INTRODUCTION

GENERAL

A text summarizer wraps a text to a specified short length. It condenses a long article to the main points. The need for text summarizers is increasing day by day, because of the time constraints.

We have converted video to text using python libraries and also converted text to summary using NLP. Then we also implemented the option of voice recognition and then produced the text using python speech recognition libraries.

Challenges in text summarization include topic identification, interpretation, summary generation, and evaluation of the generated summary. Most practical text summarization systems are based on some form of extractive summarization.

Key challenges in text summarization include topic identification, interpretation, summary generation, and evaluation of the generated summary. Most practical text summarization systems are based on some form of extractive summarization.

PROBLEM STATEMENT

To convert a video to text, text to the summary, and live audio speech to text.

SIGNIFICANCE OF PROBLEM

Video summarization plays an important role as it helps in efficient storage, quick browsing, and retrieval of a large collection of video data without losing important aspects.

The aim of video summarization is **to speed up the browsing of a large collection of video data and achieve efficient access and representation of the video content**. By watching the summary, users can make quick decisions on the usefulness of the video.

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. Summarizing 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 the 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.

BRIEF DISCUSSION OF THE SOLUTION APPROACH

Video to Summary

Our approach to the problem statement that we have chosen, started with the identification and learning of the required tools needed for the implementation of various algorithms used to convert video to text and text to summary.

Text Summarization Techniques

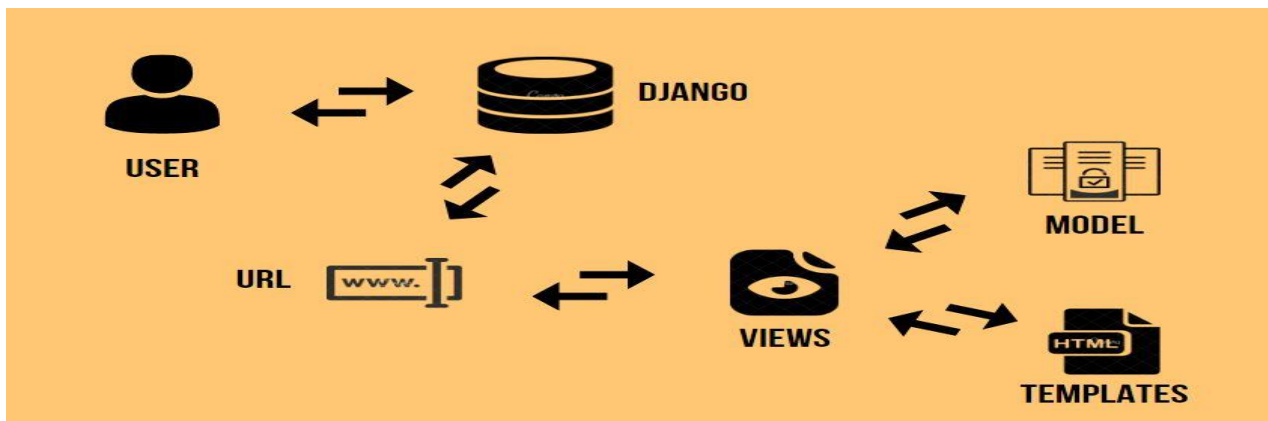
Most methods for video summarization do not make use of one of the most important sources of information in a video sequence, the spoken text or the natural-language content. Infomedia, CueVideo, and the system proposed are some exceptions. Content text is readily available for most cable TV programs 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 extracted from the text was shown to be correlated with the “abstract worthiness” 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 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.

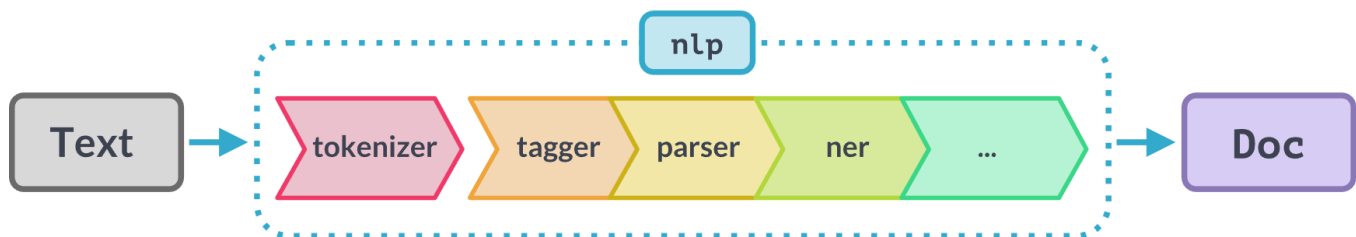
DJANGO

Django is a high-level Python web framework that enables the rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.



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 TECHNIQUES:

Extractive Approaches:

Using an extractive approach we summarize our text on the basis of simple and traditional algorithms. For example, when we want to summarize our text on the basis of the frequency method, we store all the important words and frequency of all those words in the dictionary. On the basis of high frequency words, we store the sentences containing that word in our final summary. This means the words which are in our summary confirm that they are part of the given text.

Abstractive Approaches:

An abstractive approach is more advanced. On the basis of time requirements we exchange some sentences for smaller sentences with the same semantic approaches of our text data.

We have used Statistical extraction:

Statistical extraction – uses statistical analysis to do context extraction. This is good for people names, company names, geographic entities which are not previously known and inside of well-structured text (e.g. academic or journalistic text). Statistical extraction tends to be used when high recall is preferred over high precision.

ALGORITHMS USED

We have used libraries like speech recognition for converting speech to text, and pydub for conversion. And then converted text to summary using a spacy library of python. We used NLP for a summary generation.

We also directly implemented video-to-text using: youtube_transcript_api.

Step 1: Convert the paragraph into sentences First, let's split the paragraph into its corresponding sentences. The best way of doing the conversion is to extract a sentence whenever a period appears.

Step 2: Text processing Next, let's do text processing by removing the stop words (extremely common words with little meaning such as “and” and “the”), numbers, punctuation, and other special characters from the sentences. Performing the filtering assists in removing redundant and insignificant information which may not provide any added value to the text's meaning.

Step 3: Tokenization Tokenizing the sentences is done to get all the words present in the sentences.

Step 4: Evaluate the weighted occurrence frequency of the words Thereafter, let's calculate the weighted occurrence frequency of all the words. To achieve this, let's divide the occurrence frequency of each of the words by the frequency of the most recurrent word in the paragraph.

Step 5: Substitute words with their weighted frequencies Let's substitute each of words found in the original sentences with their weighted frequencies. Then, we'll compute their sum. Since the weighted frequencies of the insignificant words, such as stop words and special characters, which were removed during the processing the stage is zero, it's not necessary to add them.

Step 6:Tokenizing the article into sentences.

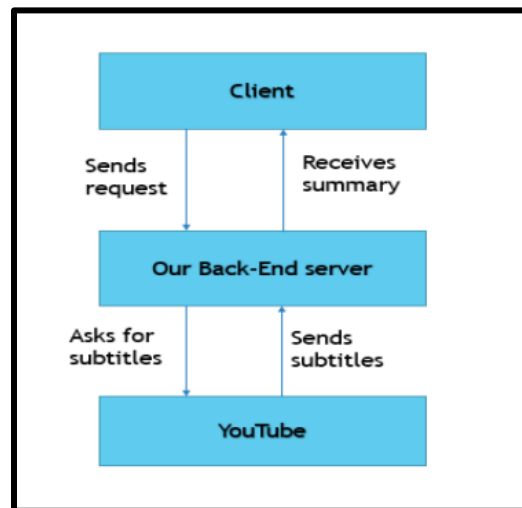
Step 7: Finding the weighted frequencies of the sentences To evaluate the score for every sentence in the text, we'll be analyzing the frequency of occurrence of each term. In this case, we'll be scoring each sentence by its words; that is, adding the frequency of each important word found in the sentence.

Step 8:Further this summary list is filtered out by taking 30% of the sentences which have the highest score. If we want a longer text summary we can also take 40-50% Instead of 30% depending on our needs.

Proposed Methodology

The steps are followed to accomplish the transcript information:

1. Client sends a request to our flask backend server
2. Django backend server asks for subtitles from YouTube using YouTube API.
3. YouTube sends the subtitles to our server and text summarization is done in our backend server
4. Client receives the summarized text



Features build and language used:

Features:

- Convert Video To Summary
- Convert Video To Text
- Convert Text To Summary
- Convert Audio To Text
- Use Mic And Type What You Speak.

Languages Used:

- Django
- Html
- CSS
- Python

Python libraries like transformers (hugging face), NumPy, IPython.display, pandas, spacy, etc have been used.

REQUIREMENTS

Hardware Used:-

- Intel i5 10th Gen
- RAM: 16 GB
- OS: Windows 11

Software Used:-

Vs Code: Transformers provides APIs and tools to easily download and train state-of-the-art pretrained models. Using pre-trained models can reduce your computing costs and carbon footprint, and save you the time and resources required to train a model from scratch. These models support common tasks in different modalities.

Visual Studio Code is a code editor in layman's terms. Visual Studio Code is “a free editor that helps the programmer write code, helps in debugging, and corrects the code using the intelli-sense method”. In normal terms, it facilitates users to write the code in an easy manner. Its features let the user modify the editor as per the usage, which means the user is able to download the libraries from the internet and integrate it with the code as per his requirements.

Python: It is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional

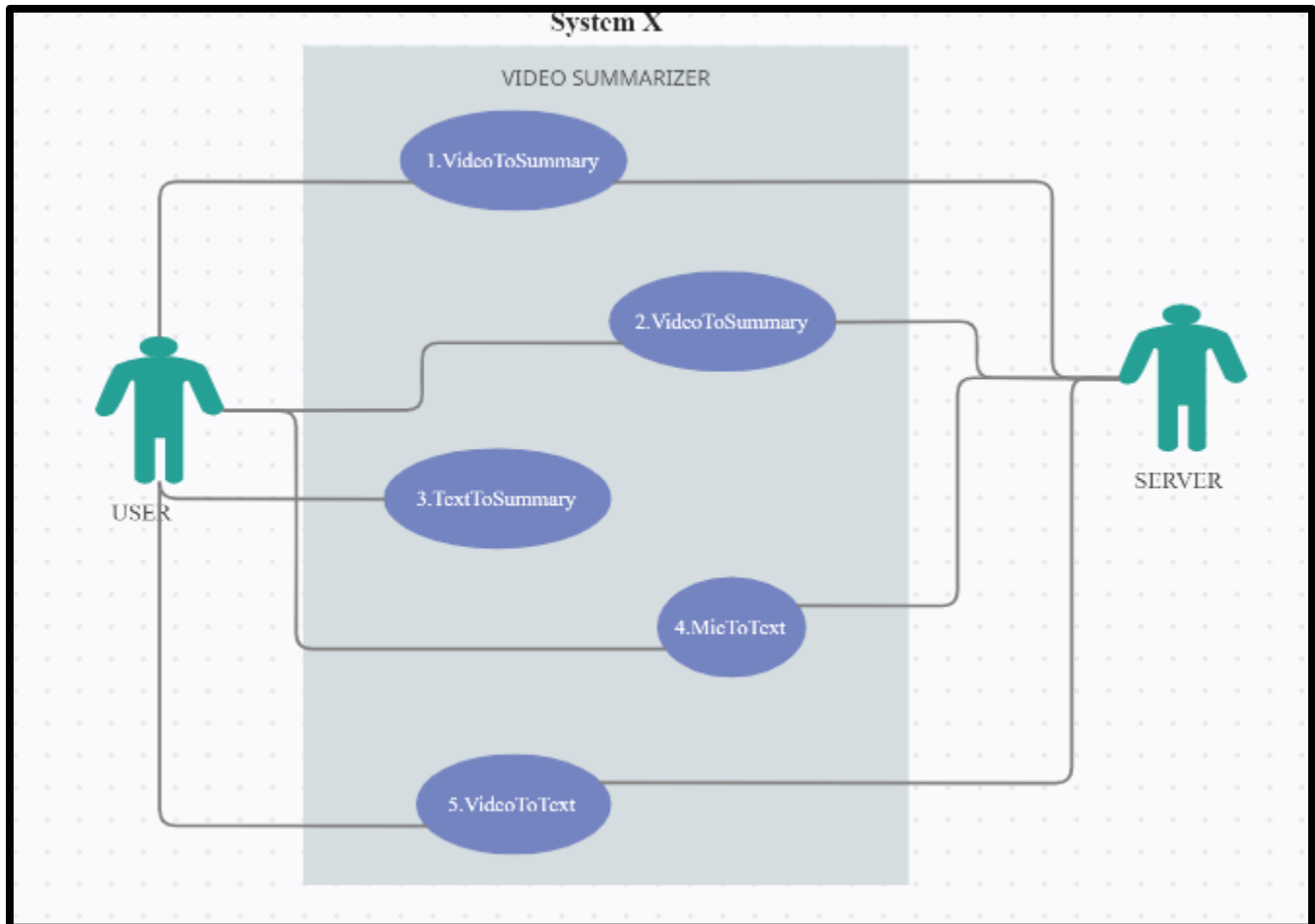
programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Libraries Used:-

- **spacy:** It is a free, open-source library for NLP in Python. It's written in Cython and is designed to build information extraction or natural language understanding systems. It's built for production use and provides a concise and user-friendly API.
- **speech_recognition:** Library for performing speech recognition, with support for several engines and APIs, online and offline.
- **heapq:** This module provides an implementation of the heap queue algorithm, also known as the priority queue algorithm. Heaps are binary trees for which every parent node has a value less than or equal to any of its children.
- **Transformers:** Transformers provides APIs and tools to easily download and train state-of-the-art pretrained models. Using pretrained models can reduce your compute costs, carbon footprint, and save you the time and resources required to train a model from scratch. These models support common tasks in different modalities, such as:NLP,computer vision,audio etc.

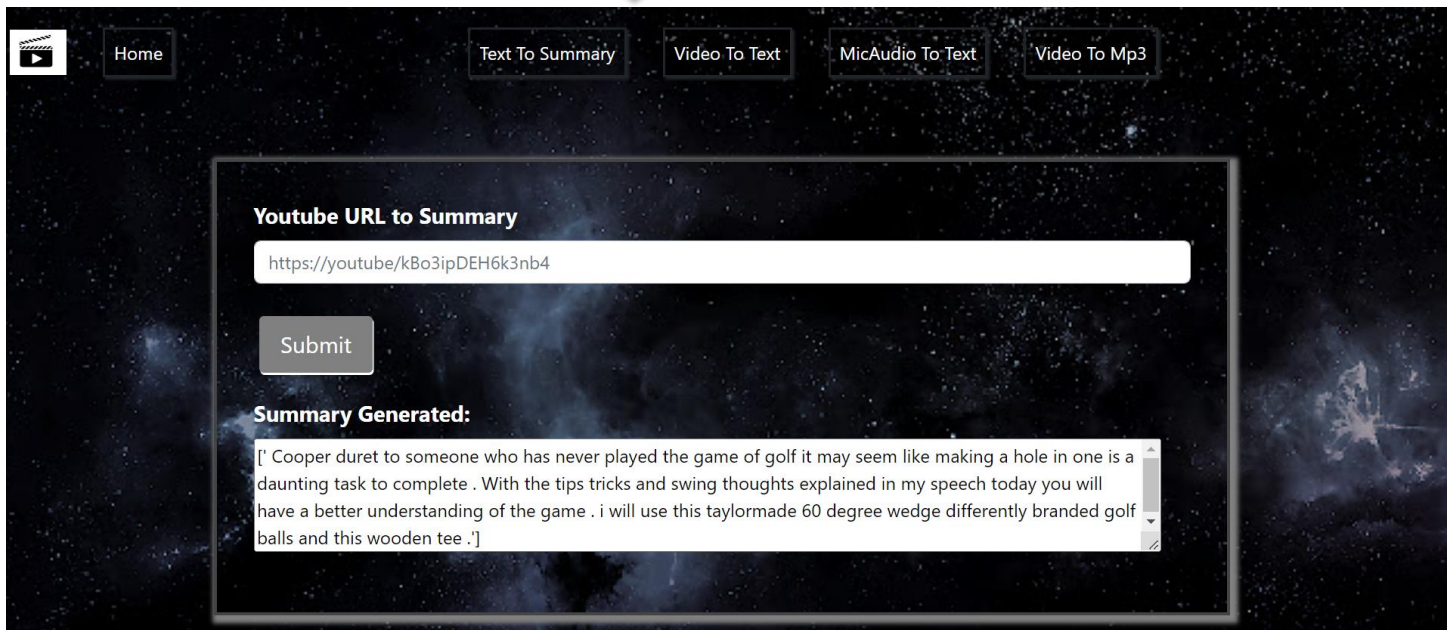
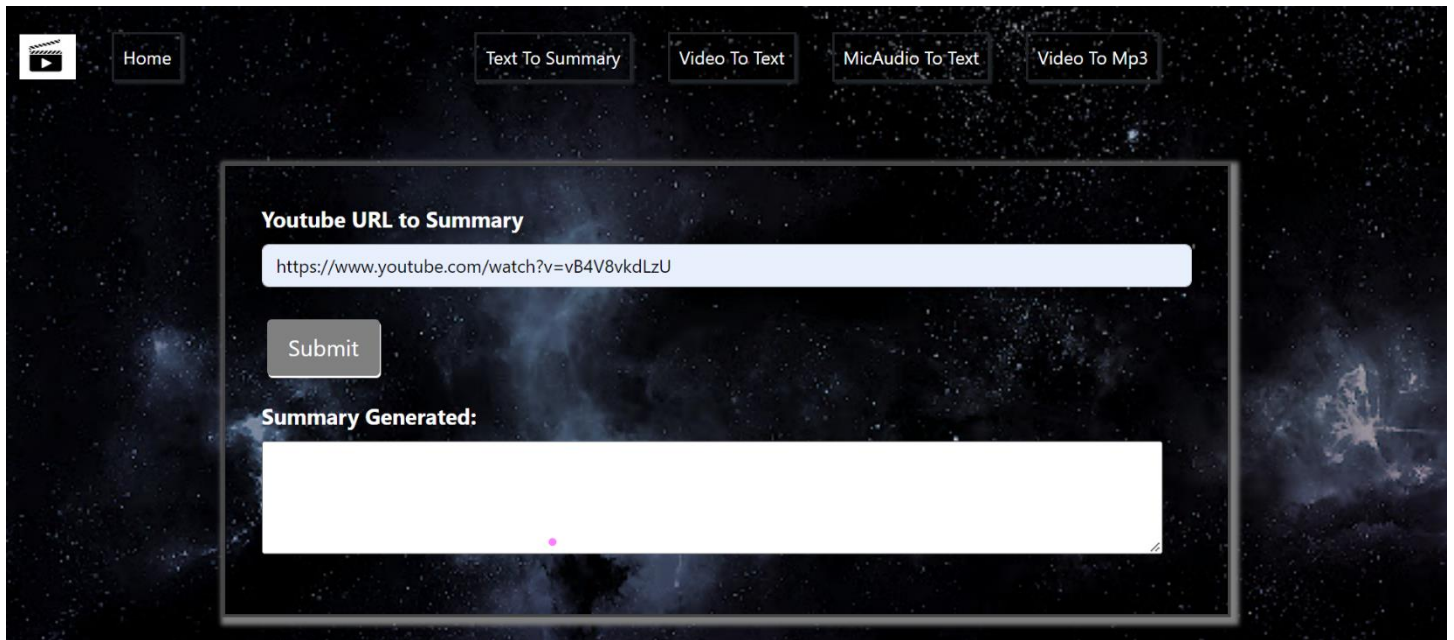
- **youtube_transcript_api**: can be used to automatically give you a transcript that you can use as plain text.
- **Pipeline**: It is API in the A Transformer library which describes the flow of data from origin systems to destination systems and defines how to transform the data along the way. The pipelines are a great and easy way to use models for inference. These pipelines are objects that abstract most of the complex code from the library, offering a simple API dedicated to several tasks.

USE CASE DIAGRAM

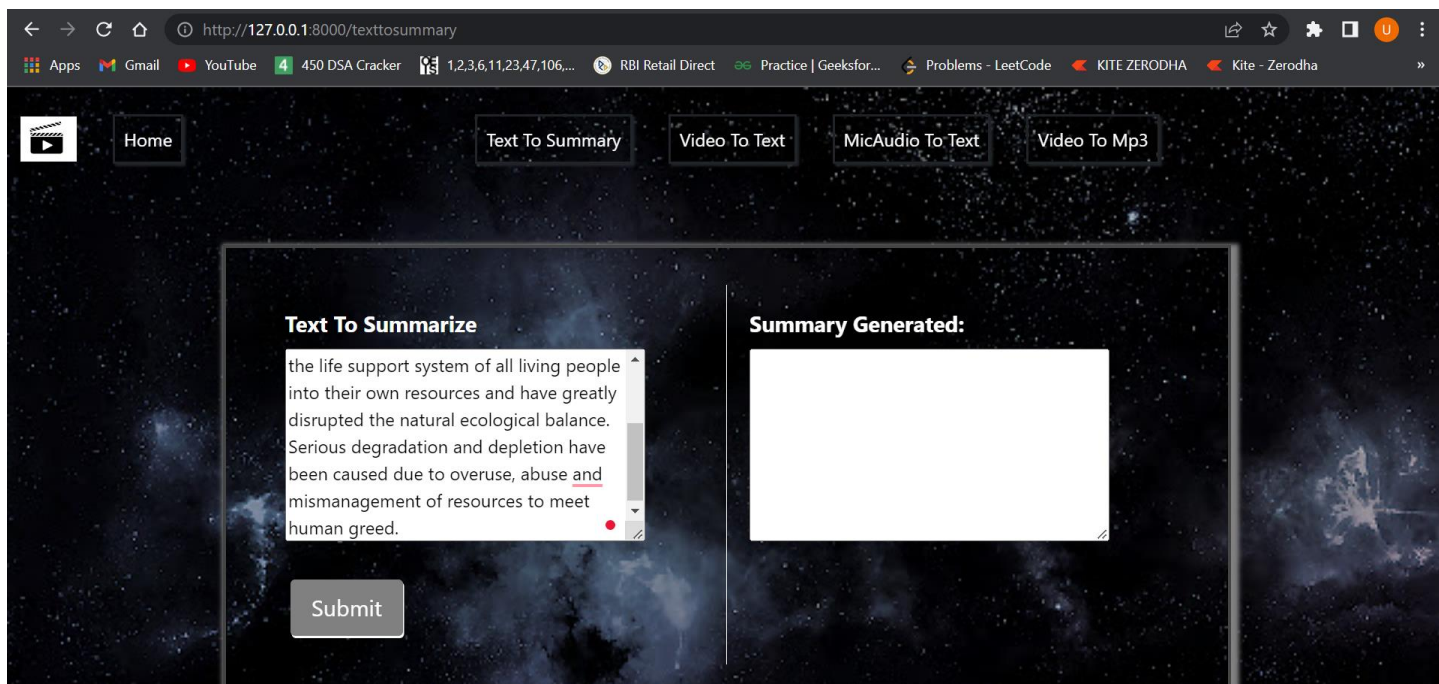


OUTPUT SCREENSHOTS

Home page: It takes the youtube video's URL whose summary you want to generate.



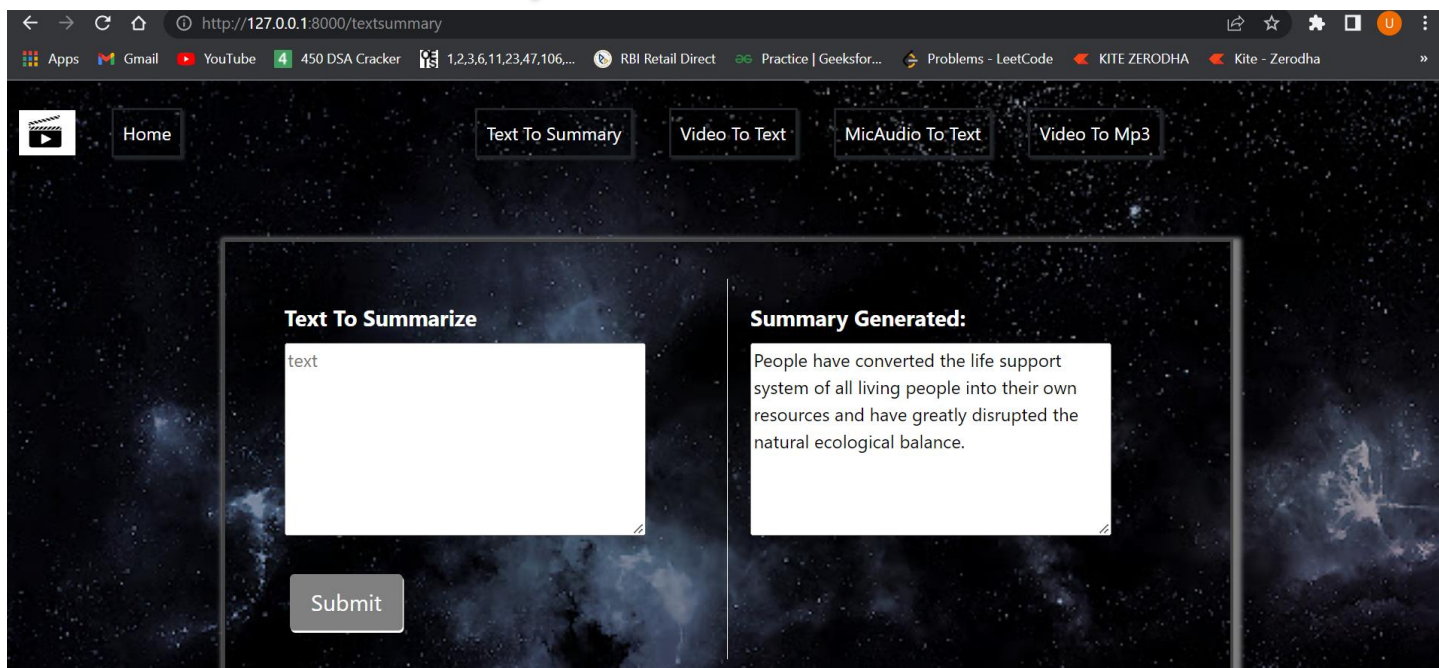
Text to summary: It generates the summary of the text you enter to summarize.



The screenshot shows a web browser at the URL `http://127.0.0.1:8000/texttosummary`. The page has a dark, starry background. At the top, there are navigation buttons: "Home", "Text To Summary", "Video To Text", "MicAudio To Text", and "Video To Mp3". The "Text To Summary" button is active. Below the navigation bar, there are two main sections. The left section, titled "Text To Summarize", contains a text input area with the following text: "the life support system of all living people into their own resources and have greatly disrupted the natural ecological balance. Serious degradation and depletion have been caused due to overuse, abuse and mismanagement of resources to meet human greed." Below the input area is a "Submit" button. The right section, titled "Summary Generated:", contains an empty white box for the summary output.

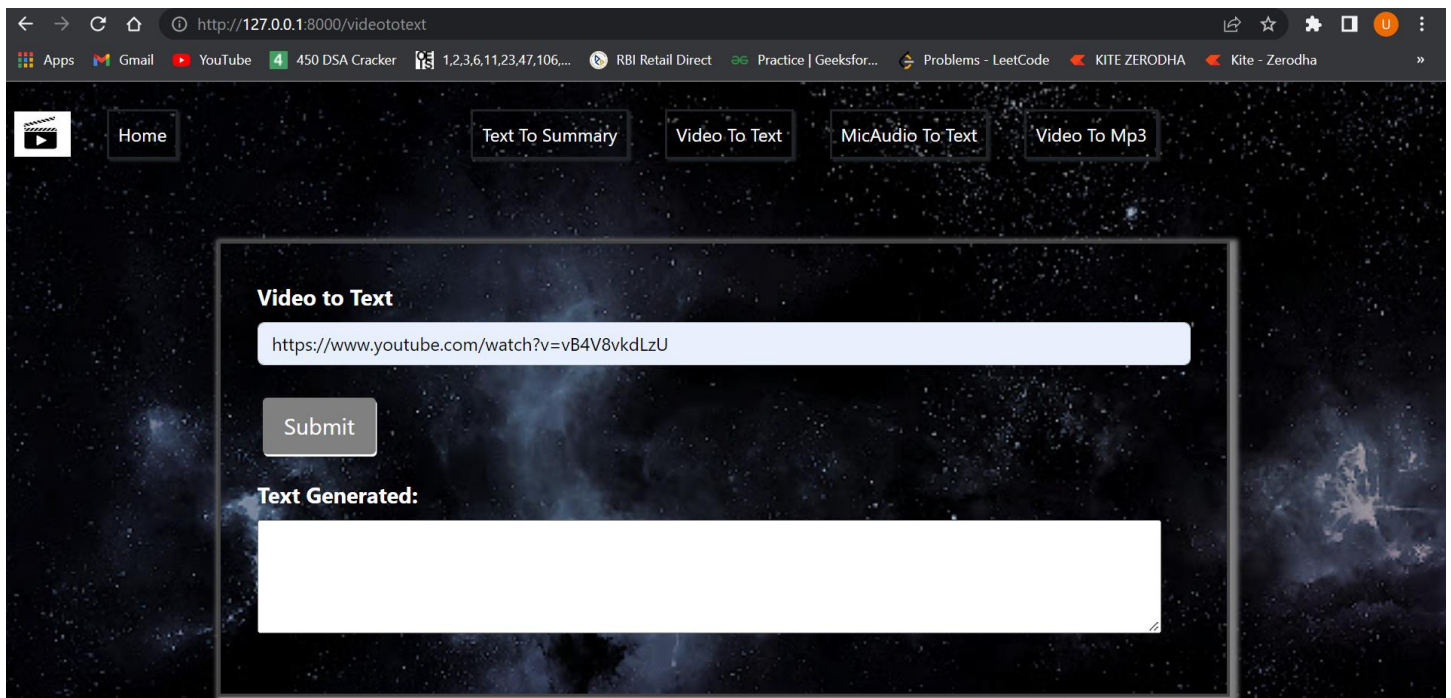


Generated summary



This screenshot shows the same web application interface as the previous one, but with the generated summary displayed. The "Text To Summarize" input area now contains the word "text". The "Summary Generated:" section now displays the following text: "People have converted the life support system of all living people into their own resources and have greatly disrupted the natural ecological balance."

Video to text:



← → ↻ 🏠 ⓘ http://127.0.0.1:8000/video2text

Apps Gmail YouTube 4 450 DSA Cracker 1,2,3,6,11,23,47,106... RBI Retail Direct Practice | Geeksfor... Problems - LeetCode KITE ZERODHA Kite - Zerodha »

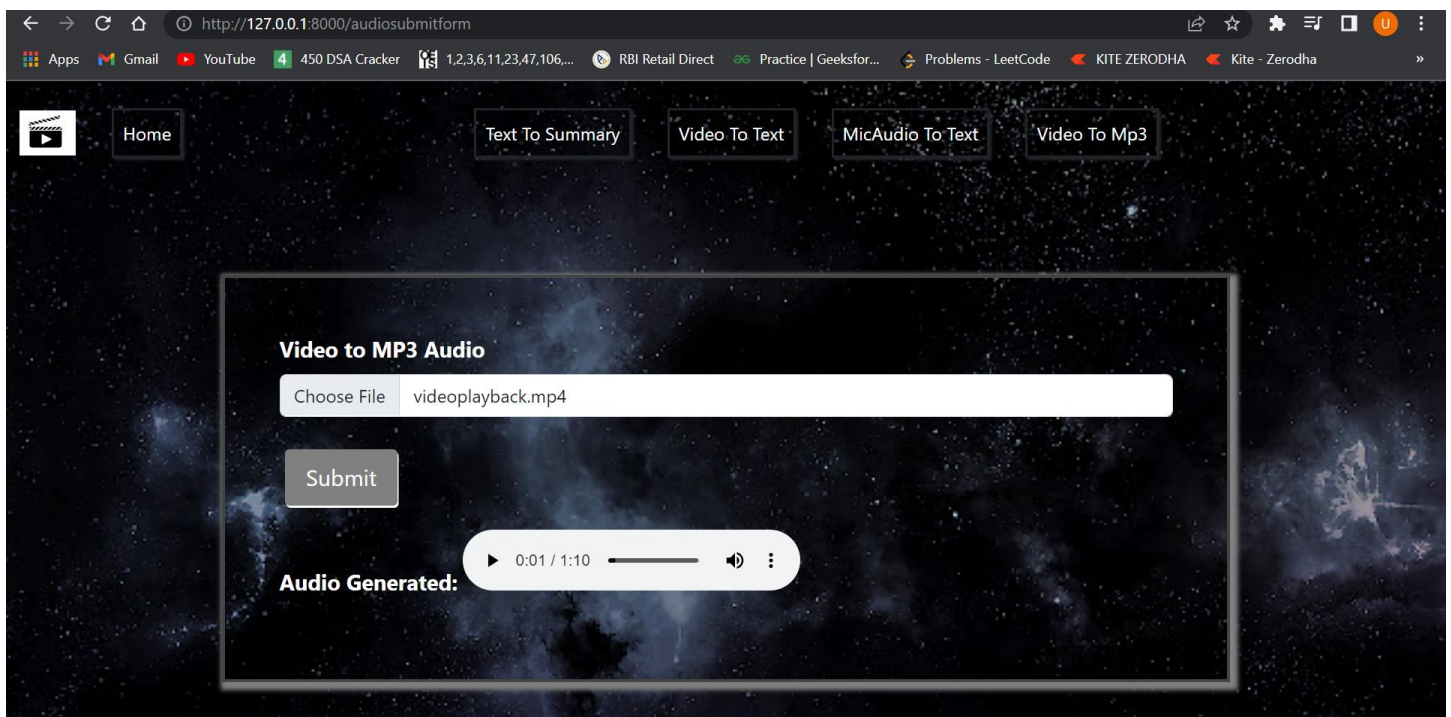
🎬 Home Text To Summary Video To Text MicAudio To Text Video To Mp3

Video to Text

Submit

Text Generated:

Video to MP3 Audio



← → ↻ 🏠 ⓘ http://127.0.0.1:8000/audiosubmitform

Apps Gmail YouTube 4 450 DSA Cracker 1,2,3,6,11,23,47,106... RBI Retail Direct Practice | Geeksfor... Problems - LeetCode KITE ZERODHA Kite - Zerodha »

🎬 Home Text To Summary Video To Text MicAudio To Text Video To Mp3

Video to MP3 Audio

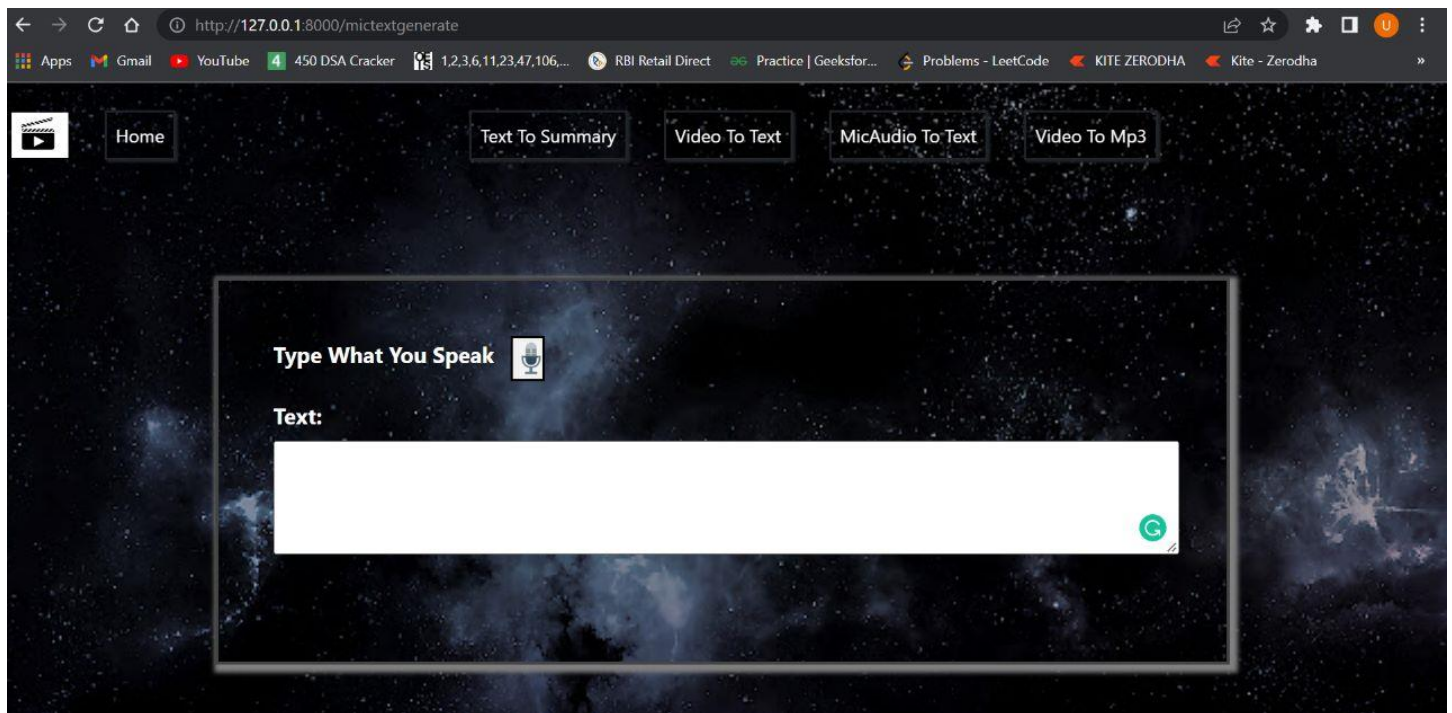
Choose File

Submit

Audio Generated:

▶ 0:01 / 1:10 🔊 ⋮

Micaudio To Text:



PYTHON CODE SCREENSHOTS:

```
home > views.py > ...
1  from django.shortcuts import render,HttpResponse
2  import requests
3  #video summarizer
4  from transformers import pipeline
5  from youtube_transcript_api import YouTubeTranscriptApi
6
7  #video to mp3
8  import moviepy.editor as mp
9  from moviepy.editor import *
10 from tkinter.filedialog import *
11
12 from IPython.display import YouTubeVideo
13
14 #mic to text
15 import speech_recognition
16 #text to summary
17 from heapq import nlargest
18 import spacy
19 from spacy.lang.en.stop_words import STOP_WORDS
20
21 # import tkinter
22 # from tkinter import messagebox
23
24
25 def index(request):
26
27     return render(request,'index.html')
28
29 def showssummary(request):
```



```

home > views.py > ...
29 def showssummary(request):
30
31     youtube_video=request.POST.get('videourl')
32     video_id = youtube_video.split("=")[1]
33
34     YouTubeVideo(video_id)
35     transcript= YouTubeTranscriptApi.get_transcript(video_id)
36
37     result=""
38     for i in transcript:
39         result+= ' '+i['text']
40
41     summarizer=pipeline('summarization')
42     num_iters=int(len(result)/1000)
43     summarized_text=[]
44     for i in range(0,num_iters+1):
45         start=0
46         start=i*1000
47         end=(i+1)*1000
48         out=summarizer(result[start:end])
49         out=out[0]
50         out=out['summary_text']
51         summarized_text.append(out)
52
53     return render(request,"index.html",{ 'output':summarized_text})
54
55
56 def mictotext(request):
57     return render(request,'mictotext.html')
58

```

```

home > views.py > ...
59 def mictextgenerate(request):
60     UserVoiceRecognizer = speech_recognition.Recognizer()
61
62     while(1):
63         try:
64
65             with speech_recognition.Microphone() as UserVoiceInputSource:
66
67                 UserVoiceRecognizer.adjust_for_ambient_noise(UserVoiceInputSource, duration=0.5)
68
69                 # The Program Listens to the user voice input.
70                 UserVoiceInput = UserVoiceRecognizer.listen(UserVoiceInputSource)
71                 UserVoiceInput_converted_to_Text = UserVoiceRecognizer.recognize_google(UserVoiceInput)
72                 UserVoiceInput_converted_to_Text = UserVoiceInput_converted_to_Text.lower()
73
74                 return render(request,"mictotext.html",{ 'micoutput':UserVoiceInput_converted_to_Text})
75
76         except KeyboardInterrupt:
77             exit(0)
78
79         except speech_recognition.UnknownValueError:
80             return render(request,"mictotext.html",{ 'micoutput':"No User Voice detected OR unintelligible noises"})
81
82
83 def audiosubmitform(request):
84
85     # print("hellooooooooooooooooooooo")
86     vid = request.POST.get('audioinp')
87     str = "static"

```

```

home > views.py > ...
87     vid = str + "/" + vid
88     # print(vid)
89
90     mp3file="static/demo.mp3"
91     videoclip= mp.VideoFileClip(vid)
92     audioclip=videoclip.audio
93     audioclip.write_audiofile(mp3file)
94     audioclip.close()
95     return render(request,"videotomp3.html",{ 'audiofile':mp3file})
96
97
98 def videotomp3(request):
99     return render(request,'videotomp3.html')
100
101
102
103
104 def textgenerate(request):
105     youtube_video=request.POST.get('videotextinp')
106
107     video_id = youtube_video.split("=")[1]
108     YouTubeVideo(video_id)
109     transcript= YouTubeTranscriptApi.get_transcript(video_id)
110     result=""
111     for i in transcript:
112         result+= ' '+i['text']
113     print(result)
114     return render(request,"videototext.html",{ 'videotext':result})
115
116

```

```

    return render(request,"videototext.html",{ 'videotext':result})

def videototext(request):
    return render(request,'videototext.html')

def texttosummary(request):
    return render(request,'texttosummary.html')

def textsummary(request):
    from string import punctuation
    text =request.POST.get('textinp');
    stopwords = list(STOP_WORDS)
    # create nlp model
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(text)
    tokens = [token.text for token in doc]
    # print(tokens) # all words of text printed
    # print(punctuation) #all punctuations are stored in this like , . / ! etc
    punctuation = punctuation + '\n'

    # now text cleaning removing punct , and stop words
    word_frequencies = {}
    for word in doc:
        if word.text.lower() not in stopwords:
            if word.text.lower() not in punctuation:
                if word.text not in word_frequencies.keys():
                    word_frequencies[word.text]=1 # if seeing word for first time , freq of that word is 1 ,
                else:

```

```

home > views.py > ...
141         if word.text not in word_frequencies.keys():
142             word_frequencies[word.text]=1 # if seeing word for first time , freq of that word is 1 ,
143         else:
144             word_frequencies[word.text] += 1
145         # print(word_frequencies)
146         max_frequency = max(word_frequencies.values())
147         for word in word_frequencies.keys():
148             word_frequencies[word] = word_frequencies[word]/max_frequency
149
150         # print(word_frequencies)
151         # sentence tokenization, take each sentence till fullstop
152         sentence_tokens = [sent for sent in doc.sents]
153         # print(sentence_tokens)
154         sentence_scores={}
155         for sent in sentence_tokens:
156             for word in sent:
157                 if word.text.lower() in word_frequencies.keys():
158                     if sent not in sentence_scores.keys():
159                         sentence_scores[sent] = word_frequencies[word.text.lower()]
160                     else:
161                         sentence_scores[sent] += word_frequencies[word.text.lower()]
162         # print(sentence_scores)
163         # now we want 30% of text of sentence with max score
164
165
166         select_length = int(len(sentence_tokens)*0.3)
167         summary = nlargest(select_length,sentence_scores,key=sentence_scores.get)
168         final_summary = [word.text for word in summary]
169         summary = ' '.join(final_summary)
170         # print(summary)

```

LIMITATIONS

A major problem with automatically-produced summaries in general, and extracts in particular, is that the output text often lacks textual coherence. Our goal is to improve the textual coherence of automatically produced extracts. We developed and implemented an algorithm which builds an initial extract composed solely of topic sentences, and then recursively fills in the lacunae by providing linking material from the original text between semantically dissimilar sentences.

CONCLUSION AND FUTURE SCOPE

We have successfully built a project where you can pass any youtube video's url, and then that project will generate a summary out of the video and make it easy for you to get information out of that video easily and in a faster way.

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.

In addition, we used HTML and CSS to develop Website and used Django to connect it with python.

This project has proposed a YouTube Transcript summarizer. The system takes the input YouTube video from the user. When the user clicks on the summarize button on the web page and accesses the transcripts of that video with the help of python API.

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

RESULT

Machine learning-based algorithms require high processing power. Summarizing a video based on its subtitle is the fastest way of generating a video summary because dealing with text is easier and faster compared to training various videos using machine learning models. This paper presents the users with a predominant advantage of producing summaries of YouTube videos. The fundamental goal of this challenge is to develop software that produces summaries of YouTube videos automatically to get the gist of the entire video before watching it.

STUDY OF REFERENCE PAPERS

- <https://www.ijarcce.com/upload/2016/march-16/IJARCCE%2040.pdf>:

This reference paper describes different techniques used for text summarization. It also describes the different methods for abstractive and extractive summarization techniques.

- <https://american-cse.org/sites/csci2020proc/pdfs/CSCI2020-6SccvdzjqC7bKupZxFmCoA/762400b503/762400b503.pdf>

It not only describes the techniques for text summarization , but also shows their advantages and disadvantages.

- <https://deliverypdf.ssrn.com/delivery.php?ID=427100007071002106029115030081107086060033061010095011108113100076068004111083120120049058116059030048032001076127086070027102050083035032000083079068100091124126069032021050091103095027064022073075102075102080125093116076069064075101116100016106085102&EXT=pdf&INDEX=TRUE>

This research paper shows NLP, and voice recognition. It is a detailed description of the NLP process.

- <https://arxiv.org/pdf/1707.02268.pdf>

This research paper shows the different applications of summarization.

REFERENCES

- <https://www.ijarcce.com/upload/2016/march-16/IJARCCE%2040.pdf>
- <https://americancse.org/sites/csci2020proc/pdfs/CSCI20206SccvdzjqC7bKupZxFmCoA/762400b503/762400b503.pdf>
- <https://deliverypdf.ssrn.com/delivery.php?ID=9991100130200021080911210920701130650280750370510190601020060820281140840061131141011240130530291060450020871180910780230881150070101075005054067113088030017006099031037085049114120089123114123074085002016006069119097021030077075074120081015016092078122&EXT=pdf&INDEX=TRUE>
- <https://arxiv.org/pdf/1707.02268.pdf>
- <https://spacy.io/usage/spacy-101>