

YouTube Video Summarization

Mallikarjun Reddy¹, Pentapati Charvitha², Jithendra Pappu³, Adigoppula Sai Bharadwaj⁴, Sri Sai Harsha Giduturi⁵, Anuradha Sesetti⁶

¹Student CSE, GITAM School of Technology, Visakhapatnam, India

²Student CSE, GITAM School of Technology, Visakhapatnam, India

³Student CSE, GITAM School of Technology, Visakhapatnam, India

⁴Student CSE, GITAM School of Technology, Visakhapatnam, India

⁵Student CSE, GITAM School of Technology, Visakhapatnam, India

⁶Assistant Professor CSE, GITAM School of Technology, Visakhapatnam, India

Abstract - YouTube Video Summarization tool combines machine learning and web development. This project aims to provide a reduced version of a video's documentation. Lengthy videos frequently waste the user's precious time and resources. The process of creating a concise summary of a text that encapsulates its key concepts is known as "text summarization." The generated summaries might include new words and phrases not in the original text. The number of videos accessible on web platforms is always growing. The rapid annual growth of YouTube users may directly impact the number of videos produced. The likelihood that video creators will misrepresent the video's original content in their lust for views is exceptionally high. Today, most industries rely more on online than offline resources, and few people have the time to watch lengthy recordings. A program that can sum up videos and save time is required to solve this issue. This project offers a solution to this issue by rephrasing transcripts of YouTube videos without missing the essential details using a system for creating video summaries using natural language analysis (NLP) by using Spacy and NLTK which is a framework to develop Python programmes for statistical natural language processing using real language data). To enhance user engagement even more, the video summarizer tool is being developed as a Chrome extension.

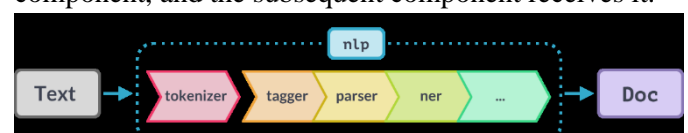
Key Words: Text Summarization, NLP, Spacy, NLTK, Chrome Extension

1.INTRODUCTION

Every day, a sizable number of videos are produced and distributed online. In 2022, there will be an average of 2.47 billion YouTube users, a significant rise each year, 300 hours of videos are posted to YouTube every single minute. Nearly one-third of YouTube users in India, according to a Google survey, spend more than 48 hours each month on the platform. It is really challenging to watch videos that are longer than you anticipated. If we are unable to locate the pertinent information we are looking for, sometimes our efforts are ineffective. Searching for videos that include the information we need can be frustrating and time-consuming due to the abundance of content on the Internet. The Second-most

visited website worldwide is YouTube. Short movies, music videos, the website offers a wide range of material from well-known YouTubers, including documentaries, live streams, vlogs, audio recordings, movie trailers, and more. YouTube receives more than one billion hours of video views each day. In order to provide an insightful and pertinent summary of the film, this project suggests using a transcription package to summarize the video's transcripts. Python includes several packages that are highly beneficial for various purposes. Currently, a Transcript API in the Python library makes it simpler to retrieve YouTube material. Using this benefit, we may easily retrieve the video's transcripts and summarize them. Two of the most well-liked Natural Language Processing (NLP) tools for Python are NLTK and spaCy. NLTK was developed by academics and researchers as a tool to aid in the development of sophisticated NLP functions. It virtually functions as an arsenal of NLP tools. As a service, on the other hand, SpaCy assists you in completing specific tasks.

The text is first tokenized with spaCy, producing a Doc object. The paper is then processed through several phases, commonly known as the processing pipeline. The trained pipelines typically use a pipeline that comprises an entity recognizer, a tagger, a lemmatizer, and a parser. The processed Doc is returned by each pipeline component, and the subsequent component receives it.



This application has also been developed as a Chrome extension, which makes it even simpler to use. The user can download the condensed texts in several different languages. For Braille users, downloadable transcripts are very useful because they may be printed out on paper by automated Braille styluses.

Real-time text-to-speech conversion is also possible for English speakers.

2. Body of Paper

2.1 Literature Review

1. The article "Natural Language Processing (NLP) based Text Summarization - A Survey" was written by Ishitva Awasthi, Kuntal Gupta, Prajbot Singh Bhojal, Anand, and Piyush Kumar and published in 2021. Abstract and extractive approaches are used for text summarizing. The advantages include the capability to determine the meanings of sentences based on linguistic and Statistical Characteristics. The disadvantage is that various situations profit from various summarizing strategies. It is impossible to determine which method is more efficient.
2. "Review of automatic text summarizing approaches & methodologies" was created by AdhikaPramita, SupriadiRustad, Abdul Shukur, and Affandy. It was published in 2020. Techniques for systematic reviews and text summaries have been used. The Fuzzy based approach has the drawback of being insufficient for semantic problems. There are various holes that need to be filled in the extractive industries' methods.
3. "Study on Abstractive Text Summarization Techniques" was created by Hardik Pradeep, Meghana Naik, and Parth Rajesh Dedhia. The publication year was 2020. Pointer Mechanism, Encoder-Decoder, and Seq2Seq have all been used. The drawback is that if several papers are provided to the model, it will not work.
4. The authors of "Abstract Summarization of video Sequences" are Anika Dilawari and Muhammad Usman Ghani Khan. They used the multi-line video description and the RCNN deep neural network model. The problem is that it only draws attention to how short the summary is. Time constraints and memory capacity are not taken into consideration.
5. "Youtube Transcript Summarizer" by Ms. K. M. Bhandare, Aishwarya A. Chigare, Utkarsha U. Patil, Shweta B. Sangle In this paper, they

proposed It was suggested to use the LSA Natural Language Processing algorithm because it needs less processing capacity and doesn't require training data.

6. "Video Transcript Summarizer" by Atluri Naga Sai Sri Vybhavi; Laggiseti Valli Saroja proposed that the built model takes user-provided video links and the necessary summary duration as input and outputs a condensed transcript. The findings show that, in comparison to other suggested methods, the final translated text was obtained in a shorter amount of time. In addition, the final text correctly and consistently conveys the main idea of the video.

2.2 Text Summarization Methods

The job of constructing a brief and fluent summary while keeping vital information and overall meaning is known as automatic text summarizing [1].

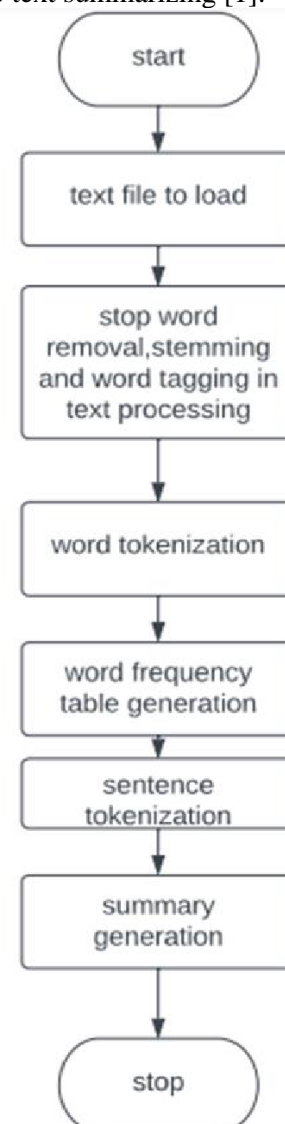


Fig 2.2.1 System Methodology for video summarization

In general, there are two methods for condensing text:

- 1) Spacy
- 2) NLTK

1) Spacy

SpaCy is a Cython-developed open-source NLP utility for Python. SpaCy was created to make it simple to develop systems for general-purpose natural language analysis or information extraction.

In Spacy we have Some Features:

i) Sentence Detection

Finding the beginning and conclusion of each sentence in each text is known as sentence detection. This makes it possible to separate a text into linguistically significant components. When processing text, these devices will be used for tasks like named-entity recognition and part-of-speech (POS) tagging.

ii) Tokenizing

Tokenizing the content is a step in the Doc container's construction. The process of tokenization separates a document into its tokens, which are represented as Token objects in spaCy.

Text with Whitespace	Is Alphanum?	Is Punctuation?	Is Stop Word?
Gus	True	False	False
Proto	True	False	False
is	True	False	True
a	True	False	True
Python	True	False	False
developer	True	False	False
currently	True	False	False
working	True	False	False
for	True	False	True
a	True	False	True
London	True	False	False
-	False	True	False
based	True	False	False
Fintech	True	False	False
company	True	False	False
.	False	True	False
He	True	False	True
is	True	False	True
interested	True	False	False
in	True	False	True
learning	True	False	False
Natural	True	False	False
Language	True	False	False
Processing	True	False	False
.	False	True	False

Fig 2.1.2

In the previous table:

- .is_punct the token's status as a punctuation sign is determined,
- .is_stop assesses the token to see if it is a stop phrase, and
- .is_alpha checks to see if the token contains alphabetic letters. We will talk about stop words later in this session.

iii) Stop Words

Typically, stop words are regarded as the expressions that are most common in a language. Stop words can be found in the English language with terms like "the," "are," "but," and "they." For most sentences to be grammatically correct and full, stop words are required. Stop words are typically eliminated when using NLP because they are superfluous and greatly skew any word frequency analysis. A collection of English stop words is kept up to date by SpaCy. In this instance, you use a list comprehension and a conditional statement to compile every word in the text that is not a stop word. Without stop words, it is impossible to know exactly what the sentence is attempting to say, but you can still deduce a lot about its broad subject.

iv) Lemmatization

The method of condensing a word's inflected forms is known as lemmatization. While still ensuring that the condensed form is linguistically acceptable. A lemma is the name for this root word or condensed version. A few examples of forms of organize are organizes, organized, and arranging. The key phrase here is arrange. You can convey various grammatical categories including tense (organized vs. organise), quantity, and more (trains vs. train), and so on, through the inflection of a word. Lemmatization is essential because it allows you to condense a word's inflected forms into a singular unit for analysis. You can normalize the content with its assistance. For instance, you will see that organizing reduces to the word arrange in its lemma form. Even though organize and organizing pertain to the same idea, the text will be counted as having different tokens if you do not lemmatize it. Lemmatization aids in avoiding redundant words that might theoretically overlap.

2) NLTK

The field of natural language processing aims to make natural human language understandable to computer programmes. (NLP). The Python software NLTK, or Natural Language Toolkit, can be used for NLP. The data that you may be analysing contains a lot of unstructured text that can be read by people. You must pre-process that data before you can automatically review it. You will get an introduction to the different text pre-processing tasks that NLTK can perform in this tutorial so that you are prepared to use them in future endeavours. Additionally, you will learn how to make graphs and perform some simple text analysis.

i) Tokenizing

Tokenizing makes it simple to divide words or sentences from the book. This will allow you to work with shorter text passages that, when read separately from the remainder of the text, are still largely coherent and understandable. It is the initial step in organising unstructured data to make it easier to assess. You will tokenize by word and tokenize by phrase when you are

analysing text. What both kinds of tokenization offer to the table is as follows:

Word by word tokenization: The basic building blocks of normal speech are words. They are the smallest important unit that is still able to exist alone. By tokenizing your text word by word, you can discover phrases that come up frequently.

Tokenizing by sentence: You can analyse word relationships by tokenizing each sentence. to one another and gain a deeper understanding of the sentence's context.

Example:

String=Muad'Dib learned rapidly, because his first training was in how to learn

Example string can be divided into phrases using `sent_tokenize ()`:

Sentences are tokenized by sentences, giving us a collection of two sentences.

- Muad'Dib learned rapidly,
- because his first training was in how to learn

Now string can be tokenized using `word_tokenize ()`:

```
["Muad'Dib",
 'learned',
 'rapidly',
 'because',
 'his',
 'first',
 'training',
 'was',
 'in',
 'how',
 'to',
 'learn',
 '.']
```

You received a collection of strings that NLTK interprets as words, including:

- "Muad'Dib"
- 'training'
- 'how'

However, the phrases listed below were also regarded as words:

- "'s"
- '.
- '.

Notice Why was "It's" divided to give you "It" and "s," but "Muad'Dib" stayed unaltered? This happened as a result of NLTK counting "It" and "'s" (a contraction of

"is") independently because it was aware that they were two distinct words. However, "Muad'Dib" was not read as two distinct terms and was instead left intact because it is not a recognised contraction like "It's."

ii) Remove Stop Words

Stop words are phrases you want to ignore, so you filter them out when processing text. Stop words are frequently used because common words like "in," "is," and "an" do not add much meaning to a document on their own.

Example:

String: Sir, I protest. I am not a merry man!

Check out the terms that made it into the `filtered_list`:

```
['Sir', ',', 'protest', ',', 'merry', 'man', '!']
```

You did remove You omitted "not," which altered the sentence's general meaning, along with some words like "am" and "a." This will not be well received by Worf. Words like "I" and "not" might seem too significant to filter out, depending on the kind of study you want to do. This is why: A pronoun, such as "I," is a context word rather than a substantive word: Content words advise you of the subjects discussed in the text or the author's feelings on those subjects. Contextual words provide insight into writing technique. In order to measure an author's writing style, you can look for patterns in the way they use context-sensitive words. Having measured their writing approach, you.

iii) Stemming

In the text processing job of stemming, you break down words to their root, or the essential component of a word. Stemming enables you to concentrate on the core meaning of a term rather than all the specifics of how it is being used. For instance, the roots of the terms "helping" and "helper" are the same. Although NLTK offers several different stemmers, you will be using the Porter stemmer.

Example:

String:" The crew of the USS Discovery discovered many discoveries. Discovering is what explorers do."

Using `word_tokenize ()` in a list comprehension, made a list:

```
['The','crew','of','the','USS','Discovery','discovered','may',
 'discoveries','.', 'Discovering','is','what','explorers','do','.',']
```

Using `stemmer.stem()` in a list comprehension, made a list of the terms in words that have been stemmed:

```
['the','crew','of','the','uss','discoveri','discov','mani','discov',
 'eri', ' ','discov','is','what','explor','do', ' '.]
```

What happened to all the terms beginning "discov" or "Discov" is as follows:

Original word	Stemmed version
'Discovery'	'discoveri'
'discovered'	'discov'
'discoveries'	'discoveri'
'Discovering'	'discov'

Those outcomes appear to be a little erratic. When "discovering" gets you "discov," why would "discovery" give you "discoveri"? There Stemming can go wrong in two different ways: understemming and overstemming. Understemming is the practise of reducing two linked words to the same stem but failing to do so. Negative mistakenly happened. when two separate phrases are incorrectly condensed to share a stem, overstemming occurs. This result is erroneous. The Porter stemming method was developed in 1979, making it somewhat dated. You can use the Snowball stemmer, also known as Porter2, in your own tasks because it is an improvement over the first one and is also offered by NLTK. It is also important to note that the Porter stemmer's

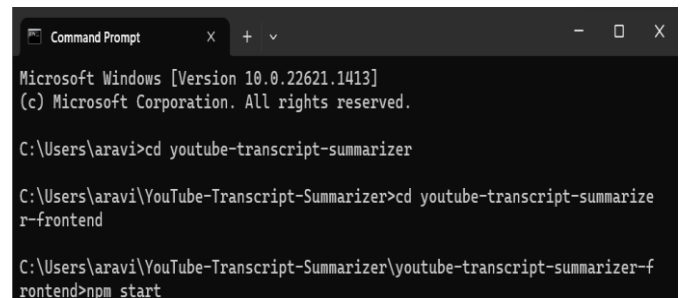
2.3 Implementation

The NLP algorithm was created in Python using spaCy and nltk. The code creates compressed text and then translates it into the desired languages using the Python translation library. The text is first preprocessed or stripped of end words and punctuation so that it can be compressed. Stop words are words that are filtered out of the stop list either before or after natural language processing because they are meaningless. By removing these words, we can make our text more focused on key information by removing low-level information. The tokenization of the words in the text is done after the pre-processing of the text. Tokenization is the process of dividing sentences in a text into separate words. Since our goal is to ensure word density in the text, this is necessary. Sentence tokenization follows. The text is divided into sentences using sentence development. Because the tokenizer is trained to use formal English text, it works well when used for literature, journalism, and official documents. Depending on the context, sentence tagging helps identify key phrases throughout the text. The sentences from the previous step are now combined into a summary paragraph. The translation package is used to translate the text into Hindi, Telugu, and Braille when it is compressed and ready for translation. The app.py file is the root of the application. All the good things about Flask come here. Here are all the paths and tasks to complete for each activity. 1. An example of a bottle framework for building web applications is bottle. So, we need to use Flask(name__) to initialize Flask. We name the piston instance by writing app = Piston(name__).2. A Python decorator called @app.route tells the app what actions to take based on the URL. 3. Requests made by the customer through the web application are handled using the request method. 4. The Flask application is run immediately after the file is executed based on the URL entered in the browser using the app.run(threaded=True) method.

2.4 Result

To deploy the application, perform these commands in the command prompt as an administrator.

- C:\Users\aravi>cd youtube-transcript-summarizer
- C:\Users\aravi\YouTube-Transcript-Summarizer>cd youtube-transcript-summarizer-frontend
- C:\Users\aravi\YouTube-Transcript-Summarizer\youtube-transcript-summarizer-frontend>npm start



```

Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aravi>cd youtube-transcript-summarizer

C:\Users\aravi\YouTube-Transcript-Summarizer>cd youtube-transcript-summarizer-frontend

C:\Users\aravi\YouTube-Transcript-Summarizer\youtube-transcript-summarizer-frontend>npm start
  
```

Fig 2.4.1 Commands

When the program is deployed, it appears to be

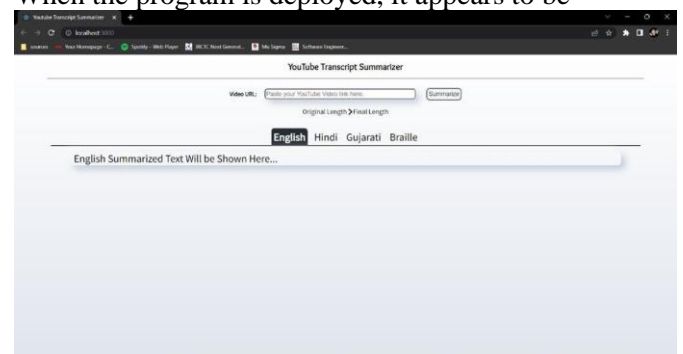


Fig 2.4.2 website

When the video URL is entered, the transcript is displayed.

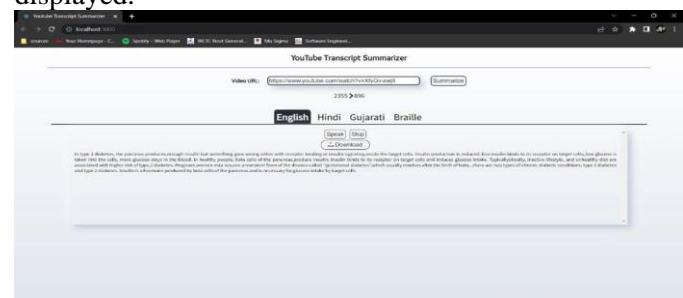


Fig 2.4.3 Summarized Text

If the URL provided is invalid, this text will be presented

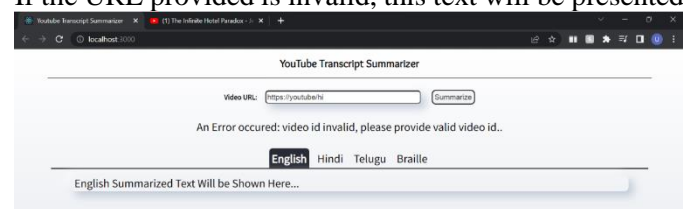


Fig 2.4.4 Error Message

This message will be displayed if the user inputs random text.

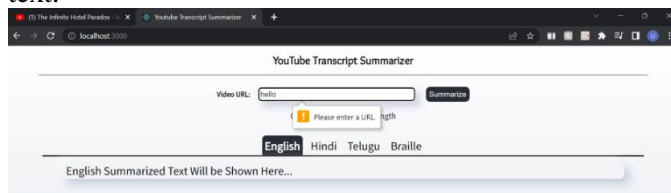


Fig 2.4.5 Warning

And Also we have the chrome extension:

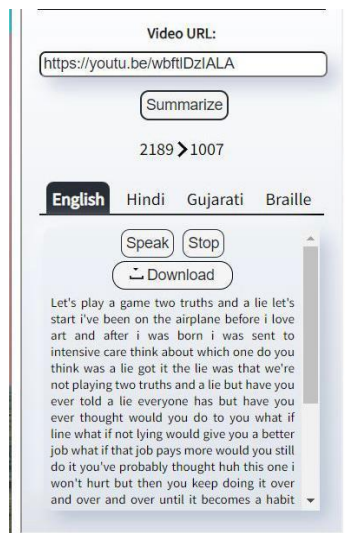


Fig 2.4.6 Chrome Extension

3. CONCLUSION

We have developed a YouTube transcript summarizer. Using a Chrome extension, users can hit the "summarize" button, and the system will pull the input YouTube video from the page and use the Python API to access its transcripts. The summary is generated using the transformers package and presented to the user on the web page for the Chrome extension. By condensing the transcript into a concise summary, our tool allows users to save time and money by quickly understanding the main points of the video without having to watch the entire thing. Additionally, it assists in identifying unusual or harmful material that may interfere with the viewing experience. Through the use of NLTK and Spacy, we were able to develop a powerful summarization tool that accurately identifies the most important information in a text. This project demonstrates the potential of leveraging multiple NLP libraries to achieve the best results in natural language processing.

4. FUTURE SCOPE

The groups of people in society who would profit most from the transcript summarization of YouTube videos are hearing-impaired people and students [1]. They would benefit if summaries were produced even for videos without easily available transcripts. The students could

choose lecture/tutorial videos based on their tastes by summarizing the YouTube videos' transcripts. Other streaming services can use the Transcript Summarization idea as well.

REFERENCES

- [1] Natural Language Processing (NLP) based Text Summarization - A Survey
https://ieeexplore.ieee.org/abstract/document/9358703?casa_token=qQ1uWklaOp0AAAAA:1Rn8FZ3ACyVuIYbYeABia6rlM7p5kFKWQpJwQChaE5K9BipORb4yHWfMumZ458nc6ypt6YJNnw
- [2] Review of automatic text summarizing approaches & methodologies
<https://www.sciencedirect.com/science/article/pii/S1319157820303712>
- [3] Study on Abstractive Text Summarization Techniques
<https://ieeexplore.ieee.org/abstract/document/9077784>
- [4] Abstract Summarization of video Sequences
<https://ieeexplore.ieee.org/abstract/document/8664480>
- [5] Youtube Transcript Summarizer
https://www.irjmetcs.com/uploadedfiles/paper/issue_3_march_2022/20210/final/fin_irjmetcs1649174683.pdf
- [6] Video Transcript Summarizer
https://ieeexplore.ieee.org/abstract/document/9751991/?casa_token=Gsm5QISQ3hEAAAAA:AFWEYta2iCrdIwD-i0rM9WoUIY66khWAgcPrLqWVcLEdKwpgm8bwouQgASQiesX7CusZQf1SRw

Extractive Automatic Text Summarization using SpaCy in Python & NLP
https://ieeexplore.ieee.org/abstract/document/9404712?casa_token=aOur0ySSJwcAAAAA:iqSO06ZQNZj_B_eMM7e4MXlaIBg7xZknHPqvETJnd1ainwcYl_G2IcYbRqdkX3PN7K8IL9ttLtTC

SpaCy Official Documentation

<https://spacy.io/api/doc>

Guide to Creating Chrome Extensions

<https://www.freecodecamp.org/news/building-chrome-extension/>

Using Google Translate API with Python

<https://codelabs.developers.google.com/codelabs/cloud-translation-python3#0>