Generate Summary of Videos Using Python

In this project, we’ll develop a YouTube video transcript summarizer that automatically extracts video transcripts from YouTube and generates concise summaries using the Natural Language Toolkit (NLTK) and sentence tokenization techniques.

To accomplish this, we’ll utilize the YouTube API to fetch the video based on the provided URL or video ID. Once the video is obtained, we’ll get the text of the video using the transcript.

With the video transcript in hand, we’ll leverage the powerful features of NLTK, a widely used natural language processing library, to tokenize the transcript into individual sentences. This sentence tokenization step allows us to break down the transcript into smaller units for analysis.

To generate the summary, we’ll utilize NLTK’s summarization algorithms. By applying techniques like extractive summarization, we’ll identify the most significant sentences and construct a condensed summary that captures the key points and main ideas of the video.

Project Tasks

1 Video-to-Text Conversion

In this project, you’ll use Python modules to generate the summary of a YouTube video. A Jupyter Notebook has been provided in the /usercode directory. You can open the Solution.ipynb file from the directory tree of VS Code.

Every task in the project has one or more associated cells in the notebook with a heading.

Note: Press “Shift+Enter” to run a single cell in the notebook. You can also click the “Play” button on the side of the cell to run individual cells.

Task 0: Get Started

In this task, you’ll import the necessary modules for this project. To begin generating a summary from a video, import the following modules:

pytube: This module will be used to interact with YouTube using the video’s URL.

youtube\_transcript\_api: This module will be used to get the transcript of the video.

spacy: This module will be used to build the NLP model.

heapq: This module will be used to generate a summary from the tokenized sentences.

Use the following structure to import modules:

from <module> import <library>

**Task 1: Import Modules**

In this task, you’ll import the necessary modules for this project. To begin generating a summary from a video, import the following modules:

* pytube: This module will be used to interact with YouTube using the video’s URL.
* youtube\_transcript\_api: This module will be used to get the transcript of the video.
* spacy: This module will be used to build the NLP model.
* heapq: This module will be used to generate a summary from the tokenized sentences.
* from pytube import extract  
  from heapq import nlargest  
  from youtube\_transcript\_api import YouTubeTranscriptApi  
  import spacy  
  from spacy.lang.en.stop\_words import STOP\_WORDS  
  from string import punctuation

**Task 2: Get the ID of the YouTube Video**

After importing all of the necessary modules, obtain the ID of the YouTube video.

Use the extract library available in the pytube module to get the ID of the YouTube video using its URL.

url = 'https://www.youtube.com/watch?v=EpipswT-LuE&ab\_channel=TED'

video\_id = extract.video\_id(url)

video\_id

**Task 3: Get a Transcript of Video**

After getting the video’s ID, obtain the transcript of the video. To complete this task, perform the following steps:

1. Get the transcript of the YouTube video using YouTubeTranscriptAPI. This will return a list of dictionary values containing a timeline and text.
2. Retrieve all the text into a new variable
3. transcript = YouTubeTranscriptApi.get\_transcript(video\_id)  
   text = ""  
   for elem in transcript:  
       text = text + " " + elem["text"]  
   text

2 Text to Summary Conversion

Task 4: Get All Available Sentences

After successfully converting the video to text, break all the text into all available sentences. To complete this task, perform the following steps:

Load the en\_core\_web\_sm model from spaCy.

Note: This model has been downloaded and installed for you.

Get all sentences using natural language processing.

nlp = spacy.load('en\_core\_web\_sm')

document = nlp(text)

for sentence in document.sents:

    print(sentence.text)

Task 5: Get All Tokens from the Document

In this task, obtain all the available tokens in the document. To complete this task, use a loop to iterate through the document and add all the tokens to a list.

tokens = [token.text for token in document]  
tokens

Task 6: Calculate the Frequency of Tokens

After obtaining all the tokens from the document, calculate the frequency of each token available in the document. To complete this task, perform the following steps:

1. Create a dictionary containing the tokens as keys and frequencies as values against each key.
2. Use a loop to iterate through all the tokens from the document.
3. If the token is not a punctuation or stop word, then increase its frequency count.

Use the lower() method to convert the word to lower case.

Use STOP\_WORDS and punctuation to check if the word is a stop word or punctuation.

word\_frequencies = {}  
for word in document:  
    text = word.text.lower()  
    if text not in list(STOP\_WORDS) and text not in punctuation:  
        if word.text not in word\_frequencies.keys():  
            word\_frequencies[word.text] = 1  
        else:  
            word\_frequencies[word.text] += 1  
word\_frequencies

Task 7: Normalize the Frequency of Tokens

After getting the frequency of each token, normalize the frequencies for better accuracy. To complete this task, perform the following steps:

1. Get the word with the maximum frequency in the document.
2. Divide each frequency with the maximum frequency to normalize the frequencies.
3. max\_frequency = max(word\_frequencies.values())  
   for word in word\_frequencies.keys():  
       word\_frequencies[word] = word\_frequencies[word]/max\_frequency  
   word\_frequencies

Task 8: Calculate the Score of Sentences

After normalizing the frequencies of each word, calculate the score of each sentence available in the document. To complete this task, perform the following steps:

1. Get all sentences from the document.
2. Create a dictionary to store the sentences as keys and scores as values.
3. Iterate through all of the sentences in the document and perform the following steps:
   1. Iterate through all words of the sentence.
   2. If the word is available in the word\_frequencies, add the frequency of that word to the sentence

Use word\_frequencies.keys() to check if the word is available.

sentence\_tokens = [sentence for sentence in document.sents]  
sentence\_score = {}  
for sentence in sentence\_tokens:  
    for word in sentence:  
        if word.text.lower() in word\_frequencies.keys():  
            if sentence not in sentence\_score.keys():  
                sentence\_score[sentence] = word\_frequencies[word.text.lower()]  
            else:  
                sentence\_score[sentence] += word\_frequencies[word.text.lower()]  
sentence\_score

Task 9: Generate the Summary

Generate Summary

After obtaining the normalized score of each sentence, generate the summary of the actual document. To complete this task, perform the following steps:

1. Get the 30% sentences with the maximum score.
2. Use these 30% sentences to get the summary of the actual text. Note: These will be the most important sentences in the document.
3. Combine all the sentences to get the summary of the document.

Use len(sentence\_tokens) to get the total number of sentences.

Use the nlargest() method from heapq to get the sentences with the maximum score

select\_length = int(len(sentence\_tokens) \* 0.3)  
summary = nlargest(select\_length, sentence\_score, key = sentence\_score.get)  
final\_summary = [word.text for word in summary]  
summary = ' '.join(final\_summary)  
summary