



TOPIC: Audible Summary of Long-form YouTube Content

COLLABORATORS: Isha Pendharkar- 19070122069,

Maanav Bhavsar - 19070122098

SUBJECT: Artificial Intelligence – Natural Language Processing

Mini Lab Project

UNDER THE GUIDANCE OF

PROF. POOJA KAMAT

Associate Professor

Symbiosis Institute of Technology, Pune

Introduction

Artificial Intelligence (AI) is the term to describe a machine's learning, logic, reasoning, perception and creativity which were once considered unique to humans but now replicated by technology and use in every industry.

Artificial Intelligence is the use of computer science programming to imitate human thought and action by analyzing data and surroundings, solving or anticipating problems, learning of self-teaching or adapting to a variety of tasks. AI can relieve humans of various repetitive tasks. The technology can learn work once and repeat it, as many times as desired by its human programmer. AI makes it possible for machines to learn from experience, adjust to new inputs and perform human-like tasks, from chess-playing computers to self-driving cars, which rely heavily on deep learning and natural language processing.

The basic objective of Artificial Intelligence, or the stimulation of cognitive behavior, is to enable computers to perform such intellectual tasks as decision making, problem solving, perception, understanding human communication in any language and translating among them.

Artificial Intelligence is now being used extensively in the domain of Natural Language Processing. It is a very wide domain. Natural Language Processing is a domain where people or artificial intelligence practitioners build models to predict, analyze and conclude out sentiments, emotions as well as generate the text based on the previously used words in the sentences.

In this project, we learnt how to use hugging face transformers, speech recognition, pipeline, etc to make a project that takes a YouTube video URL and takes its captions using YouTube transcripts or generates its captions using speech recogniser and summarizes the entire captions, converts it into an audio file making it easy for the user to go through a long video or long lectures by just listening to the summarized audio.

Literature survey

Automatic text summarization is the process of producing a summary of one or more text documents. The summary should retain the most important points of the original text document. A good text summarizer should also take into account variables such as length, writing style and syntax of the original document.

Current approaches to automatic summarization fall into two broad categories: extraction and abstraction. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In contrast, abstractive methods first build an internal semantic representation and then use natural language generation techniques to

create a summary. Such a summary might contain words not explicitly present in the original document.

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. Abstraction based summarization is inherently more difficult and is an active area of research.

Big Data Analytics by Venkat N. Gudivada, ... Vijay V. Raghavan, in Handbook of Statistics, 2015

Methodology

1. Select any long-form Youtube video (like speech, podcast or even educational content) Contrary to popular belief that YouTuber users have the attention span of a goldfish, more than a third of all YouTube view time is for videos that are 20 minutes long or over. Not only is long form a serious draw for viewers but a third of all searches on the site are news and current affairs related and 80% of all YouTube traffic now comes from outside the U.S. YouTube has spent the last couple of years encouraging original, broadcast quality programming. They have invested in a number of creators and opened up the site for monetization via paid channels for those who want to distribute their work that way. More live features have been added and YouTube are offering state of the art production facilities in L.A, London and Tokyo for creators who want to produce the best video content they can. With the general move away from traditional TV viewing and on to second screen devices and Connected TVs, YouTube need to keep encouraging this professionally produced, long form content in order to compete with Cable TV and other outlets. This is the reason we decided to choose transcripts from Youtube videos as the raw data for our project because the dataset potentially becomes endless. We can practically pick any video available on the site and our project pipeline will work on it.

2. Extract the video transcript (using API or by converting video to audio and then to text) It's tedious work to transcribe an audio file. In fact, it takes the average person anywhere from 4-8 hours to transcribe a video that is one hour in length. For most busy professionals, those are hours that could be better spent. When it comes to transcribing a Youtube video, there are two options available: if the video has "closed captions" included then we can extract these captions and convert them into a text file/list/string, or there is another more complicated way which involves converting the video into an audio file and then transcribing the audio chunks.

METHOD 1: EXTRACTING CAPTIONS

Python provides a large set of APIs for the developer to choose from. Each and every service provided by Google has an associated API. Being one of them, YouTube Transcript API is very simple to use and provides various features. The subtitles can be auto-generated by YouTube or can be manually added by the mentor. So to extract these subtitles we are using the module- `youtube_transcript_api`. This is a python API which allows you to get the transcript/subtitles for a given YouTube video. It also works for automatically generated subtitles, supports translating subtitles and it does not require a headless browser, like other selenium based solutions do. It is recommended to install this module by using pip.

The easiest way to get a transcript for a given video is to execute:

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

❏ [{"duration": 9.78,
  "start": 30.65,
  "text": "President Faust, Board of Overseers, faculty,\n alumni, friends, proud parents, members of'},
{"duration": 14.93,
  "start": 40.43,
  "text": "the ad board, and graduates of the greatest\n university in the world,'},
{"duration": 5.3,
  "start": 55.36,
  "text": "I'm honored to be with you today because,\n let's face it, you accomplished something"},
{"duration": 3.53, "start": 60.66, "text": "I never could.'},
{"duration": 5.64,
  "start": 64.19,
  "text": "If I get through this speech, it'll be the\n first time I actually finish something at"},
{"duration": 1.43, "start": 69.83, "text": "Harvard.'},
{"duration": 5.31, "start": 71.26, "text": "Class of 2017, congratulations!'},
{"duration": 10.08,
  "start": 76.57,
  "text": "I'm an unlikely speaker, not just because\n I dropped out, but because we're technically"},
{"duration": 2.75, "start": 86.65, "text": "in the same generation.'}].
```

The output is in the form of a json object so we applied a for-loop to get the required captions out of it.

METHOD 2: CONVERTING TO AUDIO FILE

We have used a package called `youtube_dl` which we will be installing using the pip command.

This package mainly allows us to download videos from YouTube and a few other websites.

Now to convert the downloaded video into an audio file, we used `subprocess`.

Now that we have an audio file, the module of `speech_recognition` is used to convert it into pure text. Speech Recognition is an important feature in several applications used such as home automation, artificial intelligence, etc. This article aims to provide an introduction on how to make use of the SpeechRecognition library of Python. This is useful as it can be used on microcontrollers such as Raspberri Pis with the help of an external microphone. It is a Library for performing speech recognition, with support for several engines and APIs, online and offline.

Due to the large size of the audio file, we had to define a function to split the file into manageable chunks. Then we applied speech recognition on these created chunks.

The output is in the form of pure text and can be used for the next step.

3. Summarizing the extracted transcript

Text summarization is a very useful and important part of Natural Language Processing (NLP). First let us talk about what text summarization is. Suppose we have too many lines of text data in any form, such as from articles or magazines or on social media. We have time scarcity so we want only a nutshell report of that text. We can summarize our text in a few lines by removing unimportant text and converting the same text into smaller semantic text form.

Text summarization in NLP is the process of summarizing the information in large texts for quicker consumption. It is essential for the summary to be a fluent, continuous and depict the significant. In fact, the google news, the inshorts app and various other news aggregator apps take advantage of text summarization algorithms.

Text summarization methods can be grouped into two main categories: **Extractive** and **Abstractive methods**. **Extractive Text Summarization** It is the traditional method developed first. The main objective is to identify the significant sentences of the text and add them to the summary. The summary obtained contains exact sentences from the original text. **Abstractive Text Summarization** It is a more advanced method, many advancements keep coming out frequently. The approach is to identify the important sections, interpret the context and reproduce in a new way.

We have used 2 extractive methods in this project:

METHOD 1: SUMMARIZATION PIPELINE

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, including Named Entity Recognition, Masked Language Modeling, Sentiment Analysis, Feature Extraction and Question Answering.

This summarizing pipeline can currently be loaded from pipeline() using the following task identifier: "summarization".

The models that this pipeline can use are models that have been fine-tuned on a summarization task, which is currently, 'bart-large-cnn', 't5-small', 't5-base', 't5-large', 't5-3b', 't5-11b'.

Each result comes as a dictionary with the following keys:

- `summary_text` (str, present when `return_text=True`) — The summary of the corresponding input.
- `summary_token_ids` (torch.Tensor or tf.Tensor, present when `return_tensors=True`) — The token ids of the summary.

Initializing the pipeline for summarization and saving the model.

By default this pipeline will make use of BART model

```
[ ] summarize = pipeline('summarization')
```

No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 (<https://huggingface.co/sshleifer/distilbart-cnn-12-6>)

Downloading: 100%  1.76k/1.76k [00:00<00:00, 43.4kB/s]

Downloading: 100%  1.14G/1.14G [00:34<00:00, 35.2MB/s]

Downloading: 100%  26.0/26.0 [00:00<00:00, 650B/s]

Downloading: 100%  878k/878k [00:00<00:00, 2.26MB/s]

Downloading: 100%  446k/446k [00:00<00:00, 694kB/s]

METHOD 2: Tf-IDF Algorithm.

TF-IDF algorithm is made of 2 algorithms multiplied together.

Term Frequency: Term frequency (TF) is how often a word appears in a document, divided by how many words there are.

$TF(t) = (\text{Number of times term } t \text{ appears in a document}) / (\text{Total number of terms in the document})$

Inverse document frequency: Term frequency is how common a word is, inverse document frequency (IDF) is how unique or rare a word is.

Example,

Consider a document containing 100 words wherein the word apple appears 5 times. The term frequency (i.e., TF) for apple is then $(5 / 100) = 0.05$.

Now, assume we have 10 million documents and the word apple appears in one thousand of these. Then, the inverse document frequency (i.e., IDF) is calculated as $\log(10,000,000 / 1,000) = 4$.

Thus, the TF-IDF weight is the product of these quantities: $0.05 * 4 = 0.20$.

Steps:

a. Text cleaning

The first step is to make the text a little less lengthy by removing the useless stopwords. Stop word removal is one of the most commonly used preprocessing steps across different NLP applications. The idea is simply removing the words that occur commonly across all the documents in the corpus. Typically, articles and pronouns are generally classified as stop words. These words have no significance in some of the NLP tasks like information retrieval and classification, which means these words are not very discriminative. On the contrary, in some NLP applications stop word removal will have very little impact. Most of the time, the stop word list for the given language is a well

hand-curated list of words that occur most commonly across corpuses. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages. You can find them in the nltk_data directory.

We also removed unnecessary punctuations from the text to make it more manageable.

- b. Create the Frequency matrix of the words in each sentence.

We calculate the frequency of words in each sentence.

The result would be something like this:

```
[ ] word_frequencies
    'American': 1,
    'Aznar': 1,
    'Belltower': 1,
    'Beyonce': 1,
    'Board': 1,
    'Boys': 2,
    'Chan': 1,
    'Change': 1,
    'Church': 1,
    'City': 1,
    'Civilization': 1,
    'Class': 4,
    'Club': 2,
    'Computer': 1,
    'Congratulations': 1,
    'David': 2,
    'Deal': 1,
    'Ec10': 1,
    'F': 1,
    'Facebook': 7,
```

Here, each sentence is the key and the value is a dictionary of word frequency.

- c. Calculate TermFrequency and generate a matrix

Now the resultant matrix would look something like this:

```

[] word_frequencies

{ ' ': 0.037037037037037035,
  '--': 0.5555555555555556,
  '10': 0.037037037037037035,
  '12': 0.037037037037037035,
  '121': 0.037037037037037035,
  '17': 0.037037037037037035,
  '2017': 0.1111111111111111,
  '22': 0.037037037037037035,
  '300,000': 0.037037037037037035,
  '50x': 0.037037037037037035,
  'Actually': 0.037037037037037035,
  'Agnes': 0.07407407407407407,
  'American': 0.037037037037037035,
  'Aznar': 0.037037037037037035,
  'Belltower': 0.037037037037037035,
  'Beyonce': 0.037037037037037035,
  'Board': 0.037037037037037035,
  'Boys': 0.07407407407407407,
  'Chan': 0.037037037037037035,
  'Change': 0.037037037037037035,
  'Church': 0.037037037037037035,
  'City': 0.037037037037037035,
  'Civilization': 0.037037037037037035,
  'Class': 0.14814814814814814,
  'Club': 0.07407407407407407,

```

If we compare this table with the table we've generated in step 2, you will see the words having the same frequency are having the similar TF score.

d. Score the sentences

Scoring a sentence differs with different algorithms. Here, we are using Tf-IDF score of words in a sentence to give weight to the paragraph.

```

▶ sentence_scores

{ President Faust, Board of Overseers, faculty,
  alumni, friends, proud parents, members of the ad board, and graduates of the greatest
  university in the world: 1.7407407407407405,
  , I'm honored to be with you today: 0.37037037037037035,
  because,
  let's face it, you accomplished something I never could.: 0.3333333333333333,
  If I get through this speech, it'll be the
  first time I actually finish something at Harvard.: 0.7037037037037035,
  Class of 2017, congratulations!: 0.25925925925925924,
  I'm an unlikely speaker, not just because
  I dropped out, but because we're technically in the same generation.: 0.6296296296296295,
  We walked this yard less than a decade apart,
  studied the same ideas and slept through the same Ec10 lectures.: 0.5185185185185185,
  We may have taken different paths to get here,
  especially if you came all the way from the Quad, but today I want to share what I've
  learned about our generation and the world we're building together.: 2.6296296296296293,
  But first, the last couple of days have brought
  back a lot of good memories.: 0.5925925925925926,
  How many of you remember exactly what you
  were doing when you got that email telling you that you got into Harvard?: 0.5925925925925926,
  I was playing Civilization and I ran downstairs,

```

e. Summarization

Now for the final step, we define a summarization function. In this function we will be able to define the length of the summary as we want it by adjusting the percentage in the function.

```
select_length = int(len(sentence_tokens)*0.3)
select_length
```

84

4. Text-to-Speech

Sometimes we prefer listening to the content instead of reading. We can do multitasking while listening to the critical file data. Python provides many APIs to convert text to speech. The Google Text to Speech API is popular and commonly known as the gTTS API. It is very easy to use the tool and provides many built-in functions which used to save the text file as an mp3 file.

We don't need to use a neural network and train the model to convert the file into speech, as it is also hard to achieve. Instead, we will use these APIs to complete a task. The gTTS API provides the facility to convert text files into different languages such as English, Hindi, German, Tamil, French, and many more. We can also play the audio speech in fast or slow mode.

However, as its latest update we cannot change the speech file; it will be generated by the system and not changeable.



To convert text files into, we will use another offline library called pyttsx3.

In the code, we have imported the API and use the gTTS function.

We saved this file as mark.mp3, which can be accessible anytime, and then we can use the playsound() function to listen to the audio file at runtime.

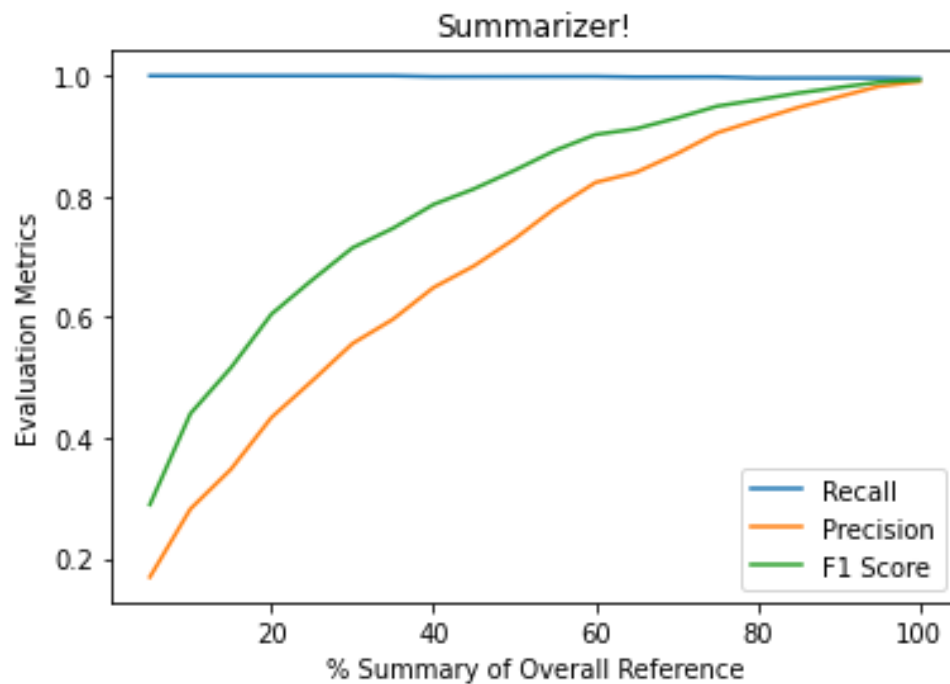
Outcomes

The final output of our NLP project is in the form of an audio file which gets downloaded immediately after the code cell is executed. This audio file contains the summary of the entire video transcript extracted from the video.

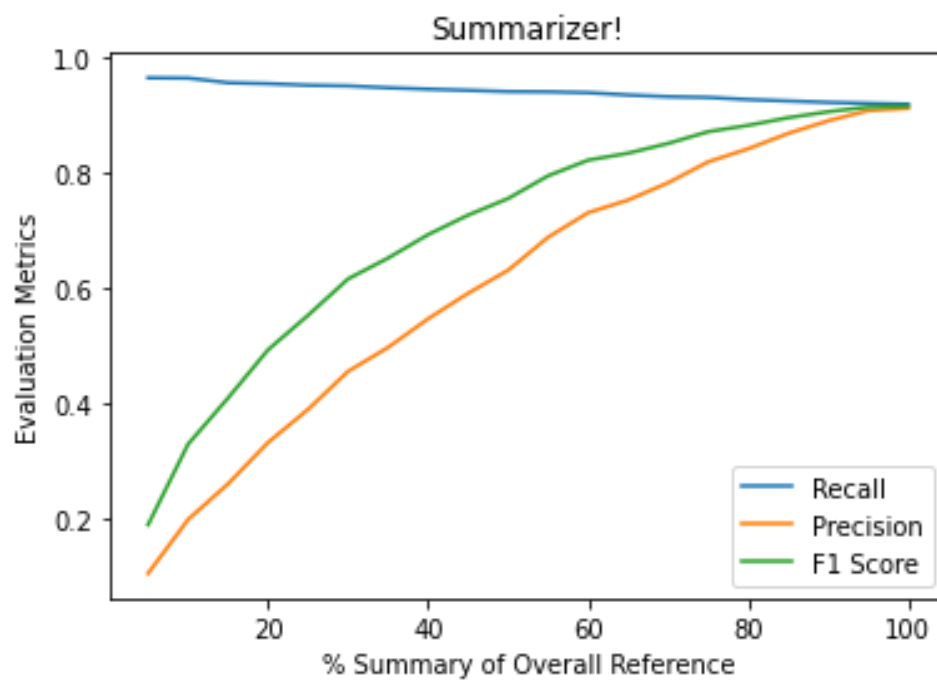
 mark (1)	20-04-2022 09:46	MP3 File	1,170 KB
 welcome	20-04-2022 09:30	MP3 File	11 KB

Result discussion with graphs

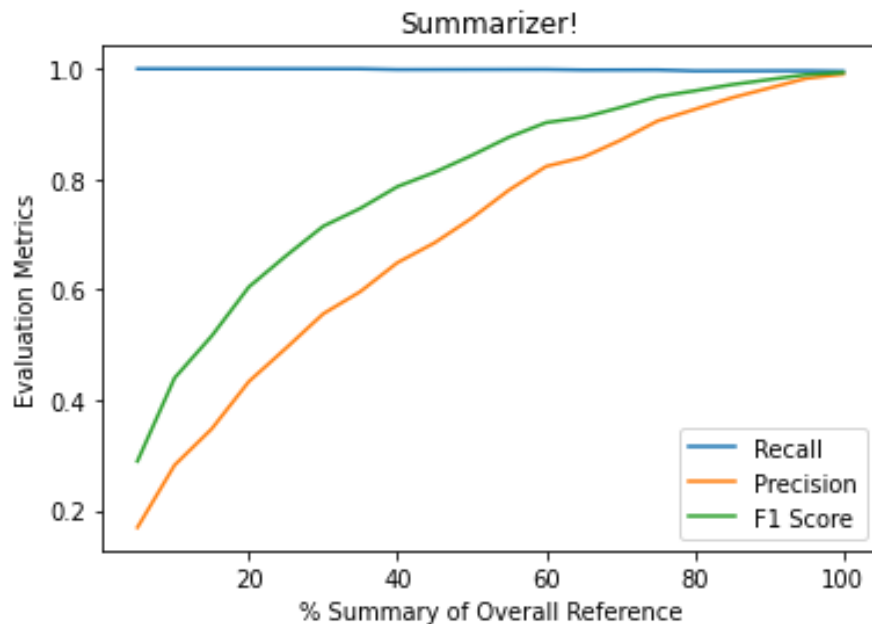
Rouge-1: refers to overlap of unigrams between the system summary and reference summary.



Rouge-2: refers to the overlap of bigrams between the system and reference summaries.



Rouge-L: measures longest matching sequence of words using LCS. An advantage of using LCS is that it does not require consecutive matches but in sequence matches that reflect sentence level word order. Since it automatically includes longest in-sequence common n-grams, you don't need a predefined n-gram length.



Limitations / gaps

- The more we try to summarize the given transcript, the more we lose our f1 score.
- Both the methods of summarization used are extractive ones.
- The audio file method takes too long to run, as the entire video has to be downloaded to the system.

Future directions

With some further finetuning, this feature can be directly integrated into the YouTube website itself. People can use this to listen to the summaries of long videos and save their time. By adding a translating model to this, it will become accessible to more people who don't have English as their first language making the content useful to more people. This project model can have a widespread use in the education field as students will be able to extract all the important parts from the lectures. Now with the increase in online learning this can be very useful.

Conclusion

Through the implementation of this project we have understood regarding NLP and how we can apply NLP for various types of projects and operations. We hope this project helps students to reduce their study time on YouTube lectures and assists them in situations when they do not have much time in hand.

References

1. Big Data Analytics Venkat N. Gudivada, ... Vijay V. Raghavan, in Handbook of Statistics, 2015
2. Blog referred for NLTK summarization techniques
<https://towardsdatascience.com/text-summarization-using-tf-idf-e64a0644ace3>
3. Rouge documentation
<https://pypi.org/project/rouge-score/>
4. Youtube video referred for Rouge evaluation theory
<https://www.youtube.com/watch?v=TMshhnrEXlg>
5. <https://www.codetd.com/en/article/13076550>
6. Blog that discusses and explains about the 3 types of rouge evaluation:
https://blog.csdn.net/qq_25222361/article/details/78694617