

## Compute part

The Idea of the project is to process the video recommendation tags for online YouTube videos.

## Working of tag recommendation system

### Steps

1. Pull a sample video from the YouTube API
2. Extract the audio from the Video
3. Convert audio to transcript (Text)
4. Use NLP libraries over the video transcript and extract the recommended tags for the videos
5. Compare the Tags with the title/ tags of the videos and show the analysis

Analysis could be any of the information that is useful for the video owner.

Following will serve as the examples for analysis

1. Compare the video description and NLP recommended tags, show how far the title, video tags are deviated from the NLP recommended tags/ key words.
2. Looking at the deviation from step 1, system recommends additional tags/ keywords which upon inclusion into the tags will improve the video when searched by user.

## Example working demonstration of the Idea

### Step 1.

Hit the YouTube API and collect a list of sample videos. For example I pulled most popular videos in US at this particular time

The screenshot shows the YouTube Data API Explorer interface. The URL bar displays `https://developers.google.com/youtube/v3/docs/videos/list`. The breadcrumb navigation shows `YouTube > Data API`. The left sidebar contains a list of API endpoints, with `Videos > list` selected. The main area contains input fields for `videoCategoryId`, `regionCode` (set to `US`), `chart` (set to `mostPopular`), and `part` (set to `snippet,contentDetails,statistics`). An `EXECUTE` button is visible. Below the inputs, a green status bar indicates `200 (OK)`. The response is a JSON object representing a list of videos, with the first video being "THE THINNING: NEW WORLD ORDER - Official Trailer".

```
* API response
*/
{
  "kind": "youtube#videoListResponse",
  "etag": "\"XI7nbFXuLYBIpL0ayR_gDh3eu1k/eF5T-OeZNYbkFqdCL9zbE7t66gE\"",
  "nextPageToken": "CAUQAA",
  "pageInfo": {
    "totalResults": 200,
    "resultsPerPage": 5
  },
  "items": [
    {
      "kind": "youtube#video",
      "etag": "\"XI7nbFXuLYBIpL0ayR_gDh3eu1k/pgcufx1Ik9GgHR3vZ1nVbmy_Hvg\"",
      "id": "W0eHsQ7q3nw",
      "snippet": {
        "publishedAt": "2018-10-17T04:00:02.000Z",
        "channelId": "UCG8rbF3g2AMX70yOd8vqIZg",
        "title": "THE THINNING: NEW WORLD ORDER - Official Trailer",
        "description": "Enjoy the trailer for THE THINNING: NEW WORLD ORDER, starring me, Peyton List, Lia M
```

Step 2.

Pull a video ID from the API and fetch the video from youtube.

Step 3.

Extract the audio from the Video

Download the mp3 file from the YouTube API command as shown below

```
skatta3@vm18-71:~$ youtube-dl --extract-audio --audio-format mp3 https://www.youtube.com/watch?v=W0eHsQ7q3nw
[youtube] W0eHsQ7q3nw: Downloading webpage
[youtube] W0eHsQ7q3nw: Downloading video info webpage
[download] Destination: THE THINNING - NEW WORLD ORDER - Official Trailer-W0eHsQ7q3nw.webm
[download] 100% of 2.35MiB in 00:00
[ffmpeg] Destination: THE THINNING - NEW WORLD ORDER - Official Trailer-W0eHsQ7q3nw.mp3
Deleting original file THE THINNING - NEW WORLD ORDER - Official Trailer-W0eHsQ7q3nw.webm (pass -k to keep)
skatta3@vm18-71:~$ ls
examples.desktop  THE THINNING - NEW WORLD ORDER - Official Trailer-W0eHsQ7q3nw.mp3
```

Update the mp3 file to a intermediate convertible file to consume by the speech to text converter later

```
skatta3@vm18-71:~$ ls
examples.desktop  speech.txt  temp.log
skatta3@vm18-71:~$ ffmpeg -i THE\ THINNING\ -\ NEW\ WORLD\ ORDER\ -\ Official\ Trailer\ W0eHsQ7q3nw.mp3 -ar 16000 -ac 1 file.wav
ffmpeg version 2.8.15-0ubuntu0.16.04.1 Copyright (c) 2000-2018 the FFmpeg developers
  built with gcc 5.4.0 (Ubuntu 5.4.0-6ubuntu1~16.04.10) 20160609
  configuration: --prefix=/usr --extra-version=0ubuntu0.16.04.1 --build-suffix=ffmpeg --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --cc=cc --cxx=g++ --enable-gpl --enable-shared --disable-stripping --disable-decoder=libopenjpeg --disable-decoder=libschroedinger --enable-gnutls --enable-ladspa --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable-libfribidi --enable-libgme --enable-libgsm --enable-libmodplug --enable-libmp3lame --enable-libopenjpeg --enable-libopus --enable-librtmp --enable-libshine --enable-libsnappy --enable-libsoxr --enable-lbspeex --enable-libssh --enable-libtheora --enable-libtesseract --enable-libvorbis --enable-libvpx --enable-libx264 --enable-libx265 --enable-libxvid --enable-libzmq --enable-libzpng --enable-frei0r --enable-libx264 --enable-libopenh264
  libavutil      54. 31.100 / 54. 31.100
  libavcodec     56. 60.100 / 56. 60.100
  libavformat    56. 40.101 / 56. 40.101
  libavdevice    56.  4.100 / 56.  4.100
  libavfilter     5. 40.101 /  5. 40.101
  libavresample   2.  1.  0 /  2.  1.  0
  libswscale     3.  1.101 /  3.  1.101
  libswresample   1.  2.101 /  1.  2.101
  libpostproc    53.  3.100 / 53.  3.100
mp3 @ 0x1f75440] Skipping 0 bytes of junk at 237.
Input #0, mp3, from 'THE THINNING - NEW WORLD ORDER - Official Trailer-W0eHsQ7q3nw.mp3':
  Metadata:
    encoder           : Lavf56.40.101
  Duration: 00:02:42.22, start: 0.023021, bitrate: 103 kb/s
    Stream #0:0: Audio: mp3, 48000 Hz, stereo, s16p, 103 kb/s
    Metadata:
      encoder         : Lavc56.60
Output #0, wav, to 'file.wav':
  Metadata:
    ISFT              : Lavf56.40.101
    Stream #0:0: Audio: pcm_s16le ([1][0][0][0] / 0x0001), 16000 Hz, mono, s16, 256 kb/s
    Metadata:
      encoder         : Lavc56.60.100 pcm_s16le
Stream mapping:
  Stream #0:0 -> #0:0 (mp3 (native) -> pcm_s16le (native))
Press [q] to stop, [?] for help
Size= 5069kB time=00:02:42.19 bitrate= 256.0kbits/s
video:0kB audio:5069kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.001503%
skatta3@vm18-71:~$ sudo pocketsphinx_continuous -infile file.wav > temp.log > speech.txt &
```

Step3.

As shown in above screenshot the last command converts the speech from the intermediate file to the text file with a log file in between.

We can see the output diff of the file while generated

```
skatta3@vm18-71:~$ cat speech.txt | tail -10
and
where
this
we are live in the memorial service that light reading all that out this governor dean reading lo
andpoint high school
skatta3@vm18-71:~$ cat speech.txt | tail -10
and
where
this
we are live in the memorial service that light reading all that out this governor dean reading lo
andpoint high school
smartphones the it known for you and try to ruin anything oversight committee for
i'm not going to be increased for haven't gotten around not working for him or not but the gun
the difference
```

The output file from here needs to be fed into the NLP module to note the diffs generated.

### **Data Intensive part**

The above compute process is repeated for a huge number of videos and the results to be displayed consolidated in a dashboard.

The backend part of streaming the data is processed from Kafka engine and the data final text, NLP intermediate files are stored on Hadoop cluster