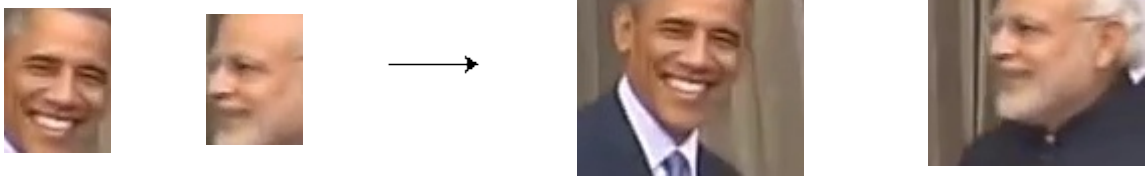
Variables and constants in 1st cell

num:

num controls how many frames per second of the video we are extracting so basically idea is that it is very computationally expensive and time consuming for long videos to perform frame by frame detection and processing. Hence we can periodically skip some frames. This is controlled by num variable. Setting num=1 would mean extract all frames. num = 2 means skip 1 frame and then extract the second and so on. num = 4 would mean skip 3 frames and then extract 1 and then again skip 3 and so on. If you have a short video and you want to process all frames, set num=1 and it should work fine.

offset:

Our emotion classifier needs a picture with a single face. Hence for multiple people in a frame, we first detect all faces and then extract only that portion of the frame which contains face. But this face portion from YOLOv3 may be too small to feed to classifier. Hence it is better to have some more pixels around the face but it should not be too much because for a crowded scnario, faces would overlap. Below pictures show the images for offset = 15. This means there would be 30 pixels more on width and 30 pixels more on the height.



Original faces from YOLOv3 Face images after offset

emotion_code_mapping:

Dictionary that defines which emotions are being detected and what are theeir labels. It will be used further in “mod-et-emo-002”.

output_FPS:

The code in “mod-et-emo-002” is generating output video that labels emotions and emotion score for each face. This is just for the purpose of easy visualization. I set it low so that it is easy to see the emotion for every frame. You can set it to high value like 20 or 25.

sentiment_threshold:

It controls boundary between “slightly” and “highly”. I worked a lot for emotion classifier from

text and the best and easiest method I found for our case is SentimentAnalyzer class from nltk.sentiment.vader module. It returns a dictionary which contains 0-1 score for the three types of emotions i.e negative, neutral and positive. A hypothetical value is as follows:

d = {“neg”: 0.18, “neut”: 0.82, “pos”: 0.0}

I observed one very important thing. Mostly the analyzer predicts neutral as 1.0 and neg and pos as 0. Only if there is some sad or happy word in the text, value of “neut” is less than 1. Hence, in my function named “calculate_polarity_and_intensity”, when “neut” value is 1 it is considered “Neutral”. For the above example, it would be classified as slightly negative although the score for neutral is highest.

But question is why not set it to 0.5 which should be logical threshold between slightly and highly. But from my observation, it gives low values (0.3, 0.4) even for harsh words like crisis in a sentence.

I thought about doing the anlyzing word by word and then combine but that would give us so many neutrals that it did not seem a good idea to me.

num_period:

This variable is for “mod-et-emo-005” and it controls the time period in milliseconds.

time_period = int(total_duration/num_period)

Hence if you want to analyze the setiment for short periods then increase num_period but if you want to analyze longer sentences, decrease num_period.

calculate_polarity_and_intensity()

You can find this method in 2nd cell which will be used in 4th and 5th project. I have defined its rules and everything from my experience to fit requirements. There is no reference of any paper or repository for this. If neutral is 1 then I say it is neutral. If I have two non-zero values let us say

d = {“neg”: 0.0, “neut”: 0.82, “pos”: 0.18} it is slightly positive.

If all three values are non-zero values like

d = {“neg”: 0.3, “neut”: 0.52, “pos”: 0.18} it will be neutral.

Because for 3 non-zero values I am finding the maximum value to decide negative, neutral or positive.

clean_words()

Here I am breaking the sentence first into words. Then I remove the all characters from each word which is not alphabet because they do not convey important information from sentiment analysis perspective.

Few important notes:

- For classification of text, I have used inspiration from this repository

  https://github.com/ian-nai/Simple-Sentiment-Analysis

- They have given 3 ways for text classification all based on NLTK library. I tried all of them and found vader analyzer most suitable and good performer for our task.
- 4th project code breaks the the whole subtitle file according to the time stamps provided in srt file. The method “clean_up” gets rid of all non-text lines and returns those lines that have text.
- 5th project calculates the total duration and time period in milliseconds and divides the whole subtitle file into ***num_period*** number of parts and length of each part is ***time_period***. Then it calculates sentiment for all these periods.
- In 6th project, Ranking is being performed based upon how many times an emotion appears. Most frequent will get rank 1 and least frequent will get rank 5.