

Predicting Likes on a Youtube Video
PreCog – TASK C



Chetan Gupta

Introduction

YouTube enables people to share videos and also interact about them. Founded in 2005, the website currently owned by Google has an alexa rank of 2 globally. As of 2014, more than 6 billion hours worth of videos have been watched on YouTube. Daily around 300k videos are uploaded and the number is steadily rising.

According to the YouTube released statistics [1] :

- YouTube has over a billion users — almost one-third of all people on the Internet — and every day people watch hundreds of millions of hours on YouTube and generate billions of views.
- YouTube overall, and even YouTube on mobile alone, reaches more 18-34 and 18-49 year-olds than any cable network in the U.S.
- More than half of YouTube views come from mobile devices.
- YouTube has launched local versions in more than 88 countries.
- You can navigate YouTube in a total of 76 different languages (covering 95% of the Internet population).

[2]The ability to socially share videos online has enabled select videos to gain a viewership of thousands in a very short period of time. Often, but not always, these videos take on a viral nature and gain tens of millions of views, while other videos only receive a fraction of the attention and viewing. These popular, viral videos also benefit from a rich-get-richer dynamic where the more popular they become, the more views they are likely to attract. Viral videos attract not only a disproportionate amount of attention, they also consume greater amounts of resource and bandwidth as well. Thus, it would be helpful to be able to predict and identify which videos are most likely to go viral for recommendation, monetization, as well as, systems performance.

What defines a YouTube Video

A YouTube video, according to the public YouTube Data API [3] has the following features :

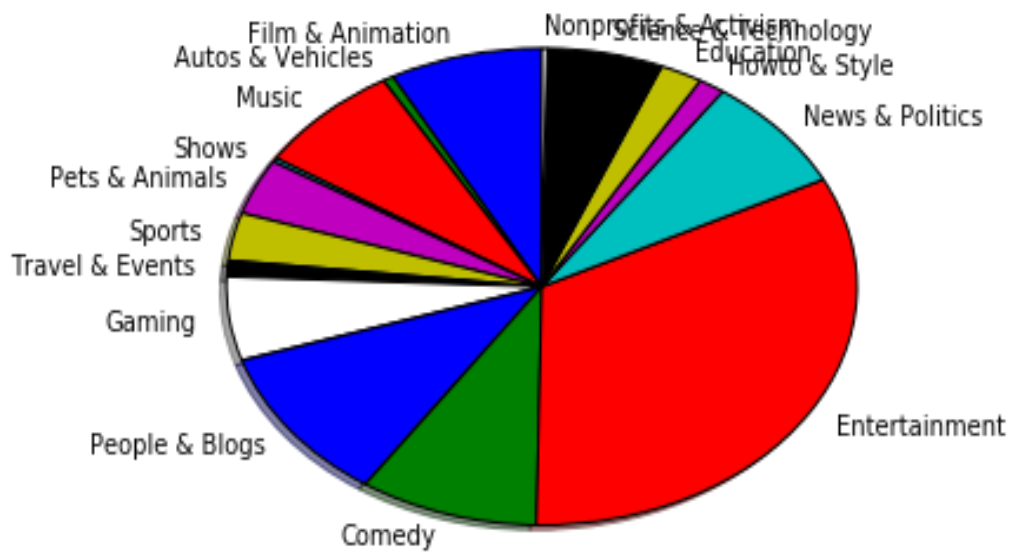
- ID - It is a unique identification key for each video uploaded on YouTube.
- Title - It is the text based title of the video.
- ChannelID - It is the unique ID identifying the uploader of the video.
- Description - It is the text based description of the video.
- CategoryID – Each video is internally given a category, by identifying its content
- PublishDate – It is the date on which a video is released.
- Thumbnail – Every video has an associated thumbnail with it.
- Duration – It is a string giving relevant details about a video's length.
- ViewsCount – The number of views on a video.
- Comment Count – The number of comments on a video.
- Comment Thread – It gives a list of relevant top level comments on a video.
- TopicIDS – It is a reference to the videos, now deprecated Freebase ID.

Using the above mentioned features, extracted from the API, I created a dataset of 420 videos by scraping the trending videos [4] page of YouTube for different countries, and stored it using MongoDB.

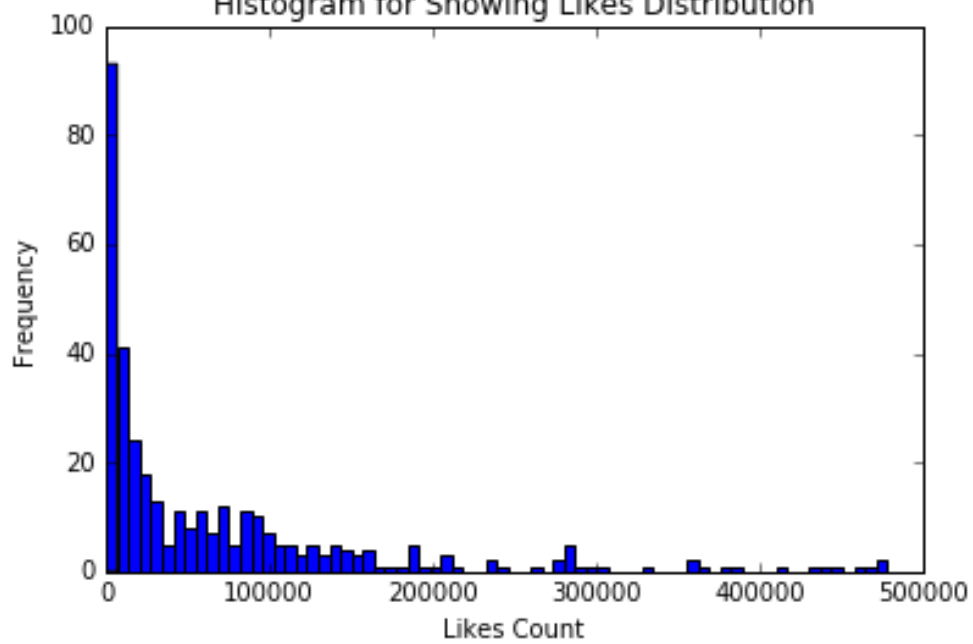
Upon simple data analysis I first removed the duplicate videos and ended up with a final raw dataset of 371 videos. Here is a sample response from YouTube API :

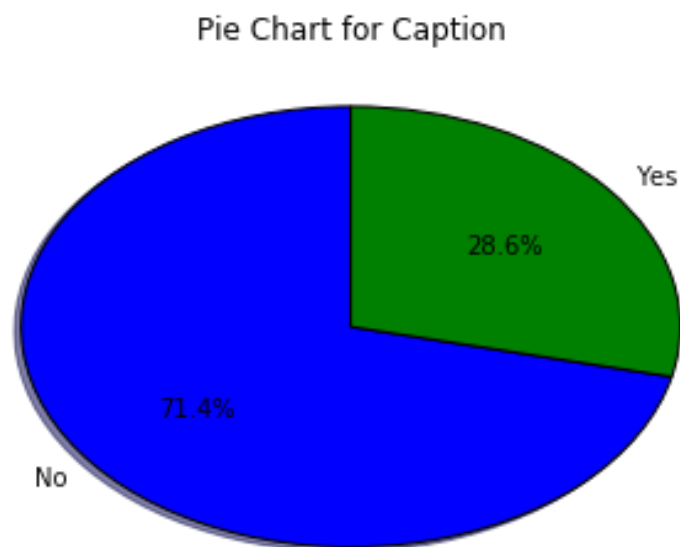
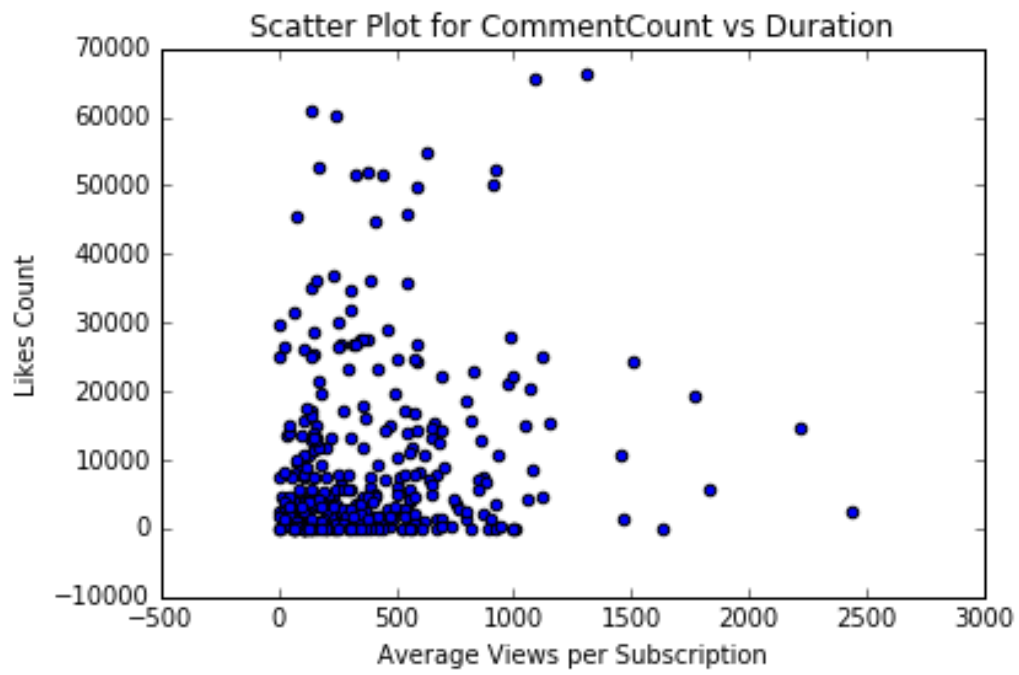


Pie Chart for Videos Categories



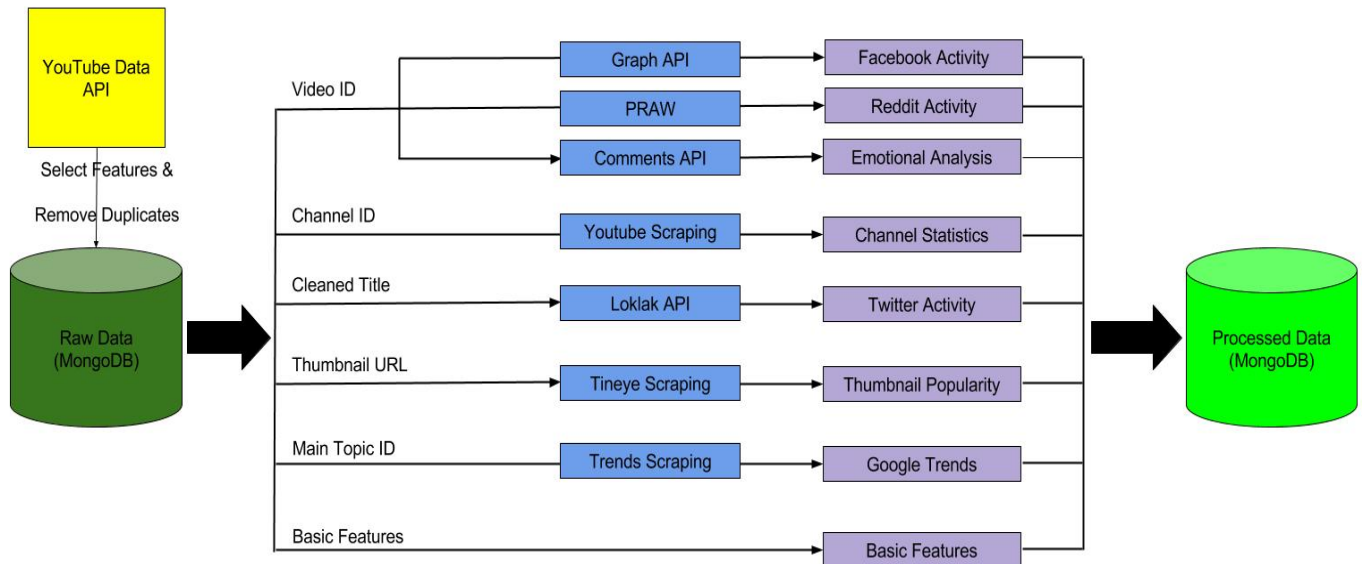
Histogram for Showing Likes Distribution





Feature Extraction from Raw Data

Now the task at hand was to extract relevant details from the raw data and also find new features identifying a video in order to predict the likes count on a video.



Description of engineered features :

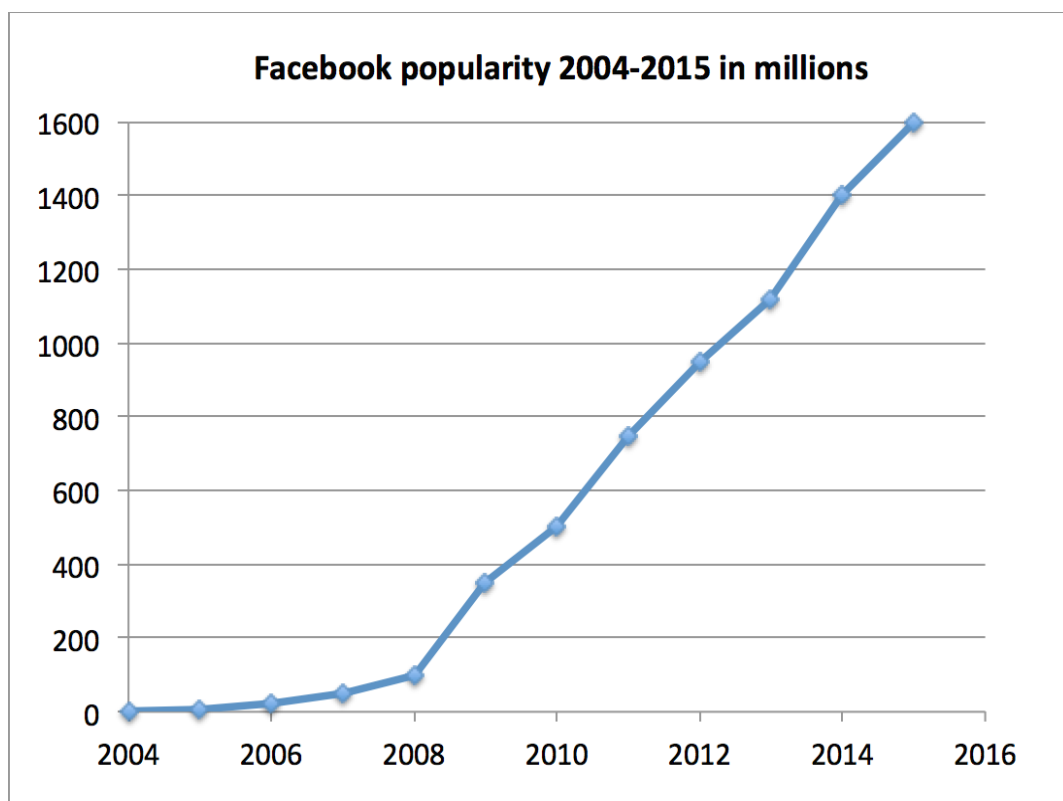
- **Channel Statistics :**
 - Each channel (uploader) on YouTube has a number of subscribers(**channelSubs**) and total number of views(**channelViews**) on all its uploaded videos. The number of subscribers is a very relevant feature, since it has been found that videos with more number of likes have been uploaded by popular channels. To extract these I wrote a web scraping solution by crawling the YouTube Page. From here I got the average views per subscriber (**viewPerSubs**). **Description :**
 - The description of video has the independence of writing anything relevant about the video/channel. Now a days every channel has its profile on top online social media platforms like Facebook, Twitter, Instagram, Snapchat etc. Popular videos have often been found to have these links in their description. Thus from the description I extracted the number of links present (**descriptionLinkCount**).
- **Caption :**
 - It is a boolean field refering to wheter the given video has closed captioning option or not. Captions can help in viewing/understanding a video originating from a different language by people not native to it. Thus helping in gathering a worldwide audience for a native languaged video.

- **Twitter Activity:**

- Twitter is an online news and social networking service where users post and interact with messages and 140 character long “tweets”. As of March 2016, Twitter had more than 310 million monthly active users. On the day of the 2016 U.S. presidential election, Twitter proved to be the largest source of breaking news, with 40 million tweets sent by 10 p.m. (Eastern Time) that day. Thus twitter as a reference for the virality of a video is a great platform. A simple twitter search using the cleaned title as a query gives relevant tweets about the video. Due to limitations of the Twitter API, for this I used a service called LokLak [5]. Loklak is a server application which is able to collect messages from various sources, including twitter. The server contains a search index and a peer-to-peer index sharing interface.
- The Loklak API returned to me a maximum of 100 tweets about the video. From this information, I extracted the features **twitterHits**(number of tweets) and **twitterCountries**(number of countries where tweets have originated).

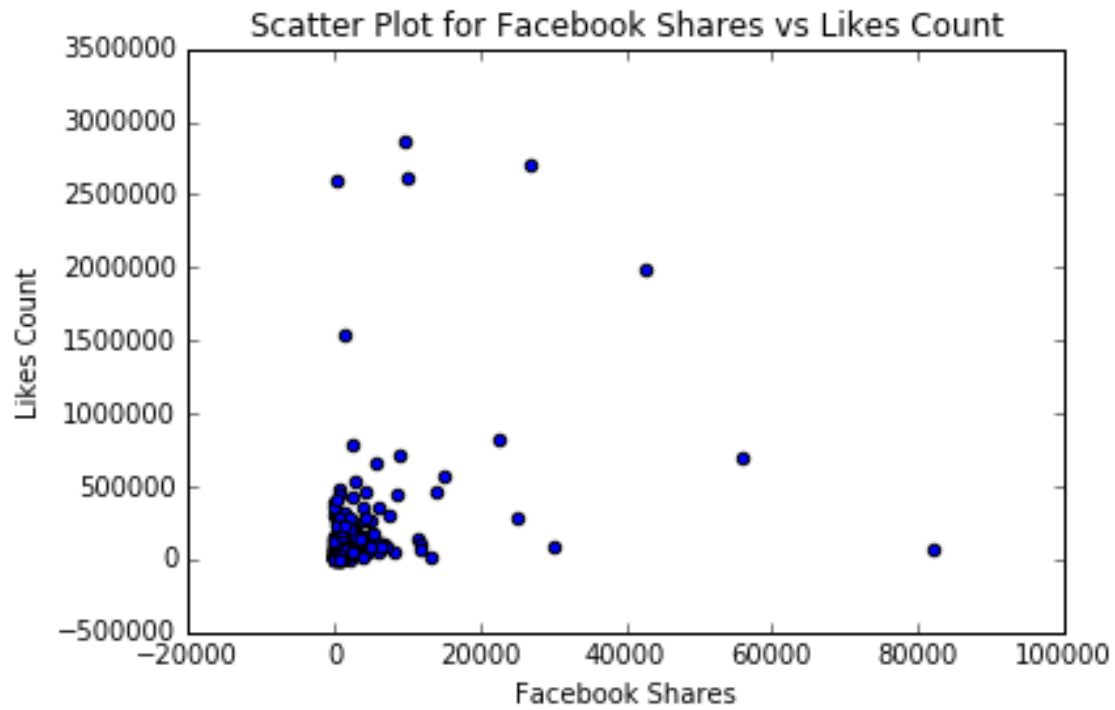
- **Facebook Activity:**

- Facebook is an American for-profit corporation and online social media and social networking service based in Menlo Park, California. It has an Alexa rank of 3. Facebook is the biggest data source when it comes to viral entities since it has a massive userbase exchanging information very rapidly. According to wikipedia :



- The Facebook Graph API [6] has an endpoint which when posted with a URL returns information like source, date originated, total number of shares etc. Owing to facebook's

popularity it is credible to say that higher number of shares of a YouTube video, more would be its views and more could be its Likes. From here thus I got the feature **facebookShares**.



- **Reddit Activity:**

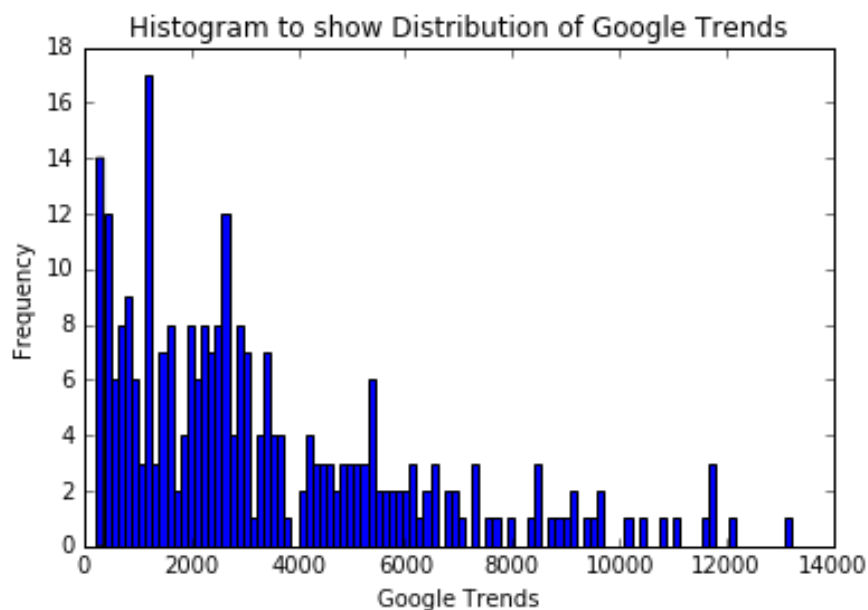
- According to Wikipedia, Reddit is a social news aggregation, web content rating, and discussion website. Reddit's registered community members can submit content, such as text posts or direct links. Registered users can then vote submissions up or down to organize the posts and determine their position on the site's pages. The submissions with the most positive votes appear on the front page or the top of a category. Content entries are organized by areas of interest called "subreddits". The subreddit topics include news, science, gaming, movies, music, books, fitness, food, and image-sharing, among many others. The site's terms of use prohibit behaviors such as harassment, and moderating and limiting harassment has taken substantial resources. It has a AlexaRank of 24, and is thus a credible source of information.
- Reddit has a system of submitting URL's to various subreddits. Users then can vote on these, comment on these. It has been found that videos with more outreach on reddit tend to become viral. Many uploaders thus also submit their video links to reddit to get more views on their content.
- The Reddit API [7] has a search endpoint which when posted with a URL gives a list of all subreddits having those and further information regarding these. From here I extracted the features **redditPosts**(Number of posts), **redditScore**(moderated number of votes), **redditComments**(Total number of comments). For this I used PRAW [8].

- **Thumbnail Popularity:**

- The YouTube API returns a list of thumbnails associated with a video. Many websites/blogs which embed youtube videos in their content, have hyperlinks to these thumbnails.
- TinEye [9] is a popular service for reverse image searching with over 17 billion images indexed and growing . By posting the website with the thumbnail URL, it returned to me the number of such websites where it found the given image.
- TinEye though did not have a free API, so I wrote a web scraping solution for getting my results. This however has the limitation of 50 searches per day per IP address. To overcome this I ran this script via multiple networks like my house, college etc. So finally this gave me the feature **thumbnailPopularity**.

- **Google Trends:**

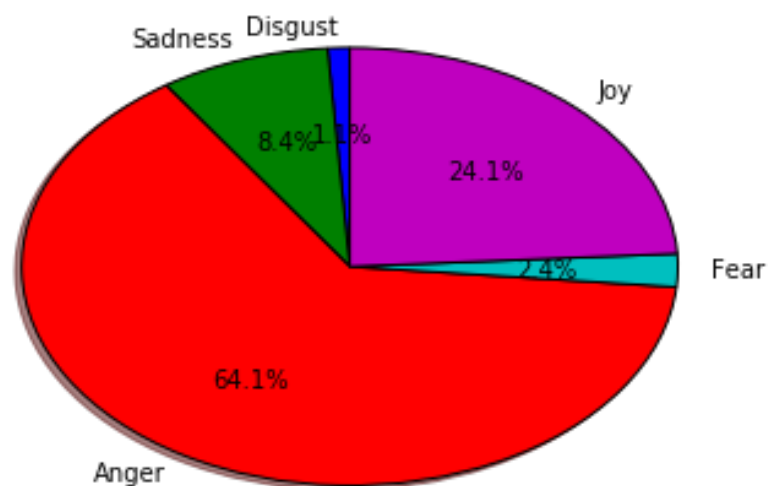
- According to Wikipedia, Google Trends [10] is a public web facility of Google Inc., based on Google Search, that shows how often a particular search-term is entered relative to the total search-volume across various regions of the world, and in various languages. The horizontal axis of the main graph represents time (starting from 2004), and the vertical is how often a term is searched for relative to the total number of searches, globally. Google Trends also allows the user to compare the volume of searches between two or more terms. An additional feature of Google Trends is in its ability to show news related to the search-term overlaid on the chart, showing how new events affect search popularity.
- For searching this I used the topicID returned by the YouTube API as the keyword for trends search. Due to unavailability of a public API, for this too I wrote a web scraping solution, with help from [11]. Once again this method had Rate Limitation per day, but unlike TinEye they were not clearly defined so had use it with trial and error on varioud networks. In the end it gave me the feature **googleTrends**.



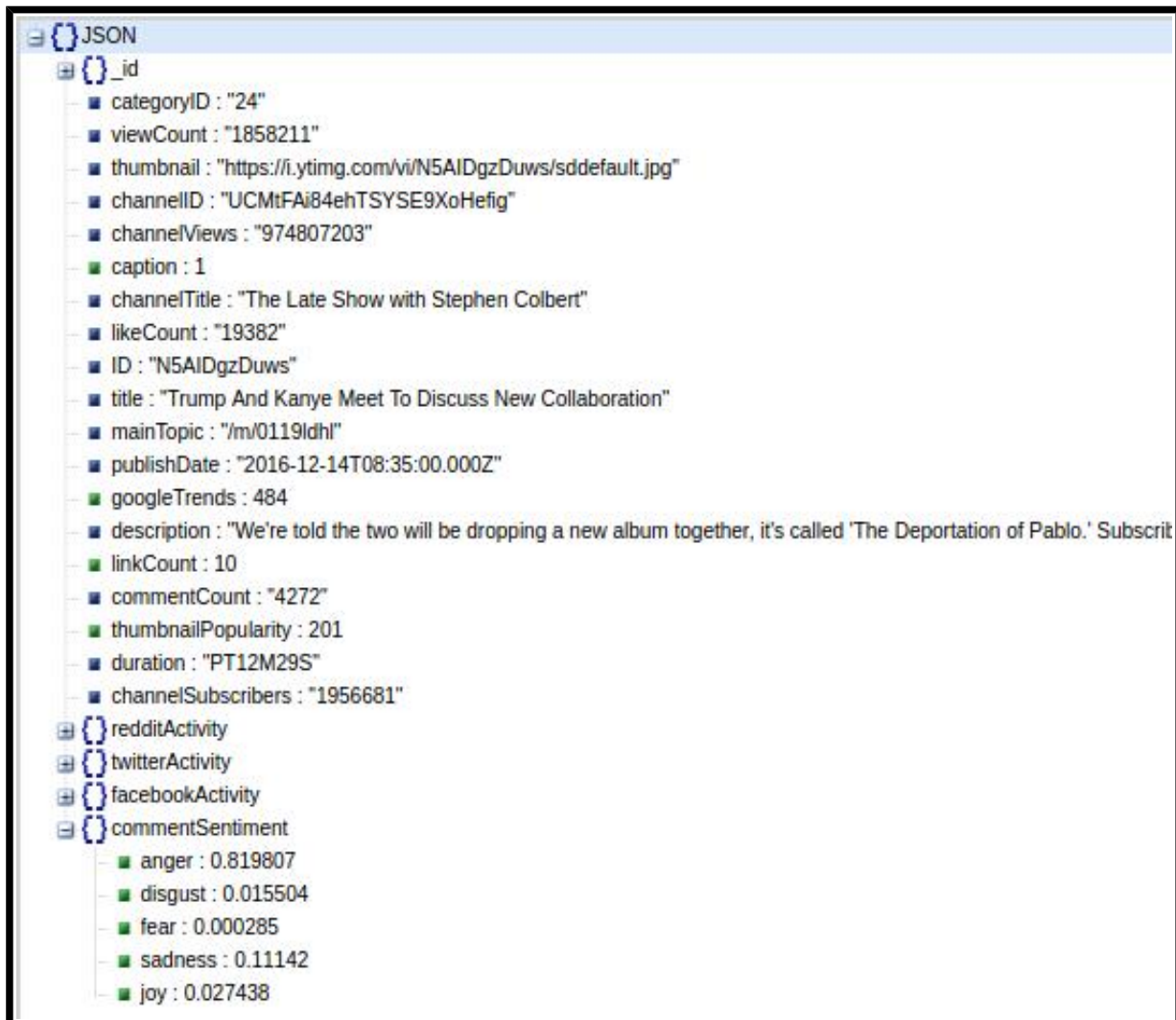
- **Comment Emotional Analysis:**

- Comments form a major form of interaction about YouTube videos. These give quite a good insight about the relevance of the topic and its reception among the masses. The YouTube Data API [3], has a separate endpoint for comment thread of a video. Analysing every single comment was not feasible to me due to rate limitations, so I got the top 50 most relevant top level comments for each video using the API, and cleaned them using simple regular expressions.
- Sentiment analysis the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. **The IBM Watson™ Tone Analyzer Service** [12] uses linguistic analysis to detect three types of tones from written text: emotions, social tendencies, and writing style. Emotions identified include things like anger, fear, joy, sadness, and disgust. Identified social tendencies include things from the Big Five personality traits used by some psychologists. These include openness, conscientiousness, extraversion, agreeableness, and emotional range. Identified writing styles include confident, analytical, and tentative.
- I used the above mentioned service to extract the emotional sentiment for the entire comment thread by merging the output of the YouTube API with the input of the Tone Analyser Service. In the end I was left with 5 features : **commentAnger**, **commentDisgust**, **commentFear**, **commentJoy**, **commentSadness**. It has been found that certain emotions trigger virality.

Pie Chart Showing Dominant Emotion in Comments



The final processed data entity looked like this :



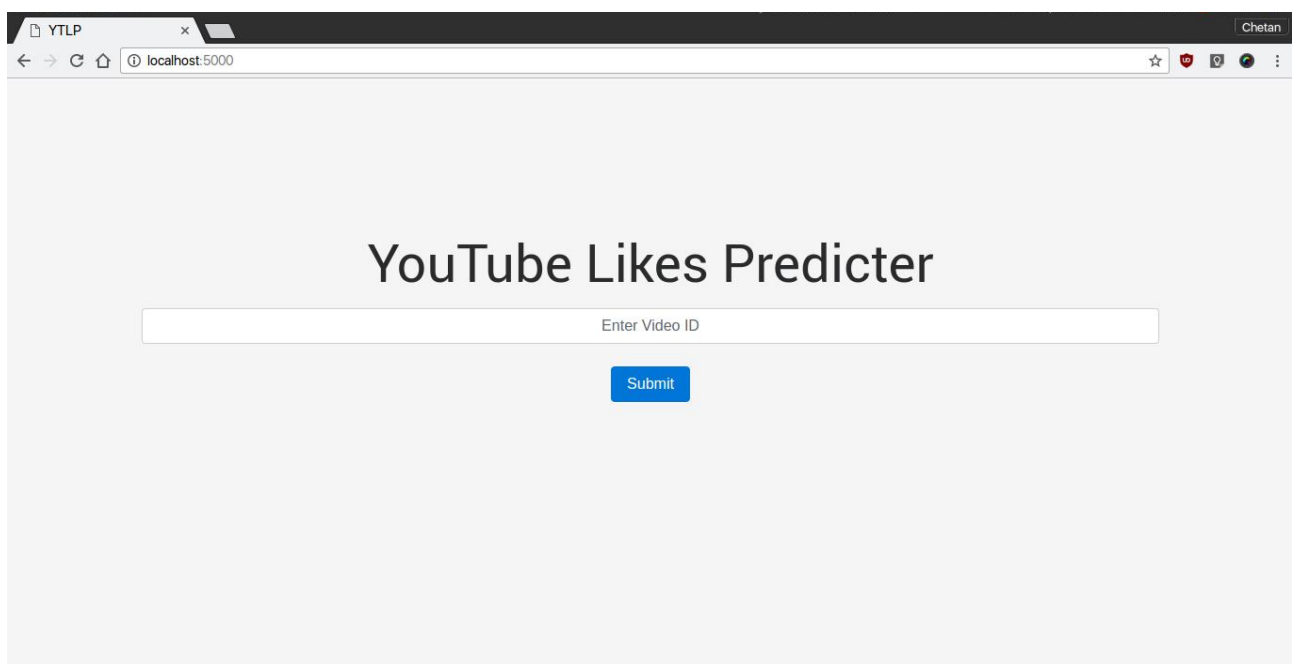
Making Predictions from Processed Data

Once we get the processed data, to make predictions regression techniques are required. From the processed MongoDB data, first I extracted numeric features into a pandas DataFrame and upon certain experimentation, selected Kneighbours Regressor. The regressor has been implemented via scikit learn.

Kneighbours Regressor [13] :

Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based the mean of the labels of its nearest neighbors. scikit-learn implements two different neighbors regressors: KNeighborsRegressor implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. RadiusNeighborsRegressor implements learning based on the neighbors within a fixed radius r of the query point, where r is a floating-point value specified by the user. The basic nearest neighbors regression uses uniform weights: that is, each point in the local neighborhood contributes uniformly to the classification of a query point. Under some circumstances, it can be advantageous to weight points such that nearby points contribute more to the regression than faraway points. This can be accomplished through the weights keyword. The default value, weights = 'uniform', assigns equal weights to all points. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied, which will be used to compute the weights.


Upon applying 10 Fold Cross Validation I got R^2 error within the range **(0.32-0.60)** due to random shuffling of data. Then for making further predictions I saved the classifier using joblib and wrote a flask based web app with Bootstrap as frontend.



YTLP

Chetan


localhost:5000/predict

<div>Actual Likes</div> <div>43221</div>	<div></div> <div>Cars 3 "Lightning Strikes" Extended Look</div> <div>Error = 49%</div>	<div>Predicted Likes</div> <div>64609</div>
--	---	---

YTLP

Chetan

localhost:5000/predict

<div>Actual Likes</div> <div>41215</div>	<div></div> <div>Be Free (Original) Pallivaalu Bhadravattakam (Vidya Mashup Cover) (ft. Vandana Iyer)</div> <div>Error = 3%</div>	<div>Predicted Likes</div> <div>39608</div>
--	--	---

Programming Guide

The following modules have been used to carry out the data gathering process :

1. **getRawDataFromAPI.py** : To get gather data form YouTube API into MongoDB
2. **extractBasicFeatures.py** : To extract basic features from raw data.
3. **getAdvancedFeatures.py**: To get engineered features from raw Data.
4. **getDataFrame.py** : To convert MongoDB instance into a pandas DataFrame.
5. **performRegression.py** : To perform regression and giving prediction.
6. **server.py** : Flask based service to do entire process.

The following modules were used to access various APIs and services :

1. **utils/channelStats.py** : For getting relevant uploader statistics.
2. **utils/tine_eye.py** : Get thumbnail popularity from TinEye.
3. **utils/lokloak_search** : Get twitter activity.
4. **utils/youtube_api.py** : Interact with the YouTube Data API
5. **utils/google_trends.py** : Get google trends data.
6. **utils/facebook_search.py** : Interact with Facebook Graph API.
7. **utils/reddit_search** : Interact with reddit API via PRAW.
8. **utils/comments_analyser.py** : Get emotional analysis using IBM Tone Analyser.

Future Work

The current work can predict YouTube video likes count with some error. It uses various sources of information like Facebook, Twitter, Reddit, Google Trends etc to model the likes count. Clearly with more data about more diverse videos, the results can be greatly improved.

References

1. <https://www.youtube.com/yt/press/statistics.html>
2. Viral Actions: Predicting Video View Counts Using Synchronous Sharing Behaviors
3. <https://developers.google.com/youtube/v3/docs/>
4. <https://www.youtube.com/feed/trending>
5. https://github.com/loklak/loklak_server
6. <https://developers.facebook.com/docs/graph-api>
7. <https://www.reddit.com/dev/api>
8. <https://github.com/praw-dev/praw>
9. <https://tinEye.com/>
10. <https://www.google.com/trends/>
11. <http://techslides.com/hacking-the-google-trends-api>
12. <https://www.ibm.com/watson/developercloud/doc/tone-analyzer/>
13. <http://scikit-learn.org/stable/modules/neighbors.html#regression>