

# **YouTube Recommender System**

Aaron Wu (pwu8), Abhishek Dutta (adutta2), Natalie Roe (nroe), Wennie Zhang (yzhang46), Preston Law (plaw)

## **I. Vision/Brief Overview**

Our vision was to build a web application that displayed the top 10 videos pertaining to a particular topic on YouTube to a user upon a search-query. Using the YouTube API, we extracted data from all videos uploaded since March 1 2017, capping the time horizon at the end of April. Using machine learning algorithms, we predicted labels for videos in our dataset based on their title, description, and user-assigned tags. From the labels that our model assigned each video, our web application returns the top videos with a particular label which are ranked based on their likes-to-dislikes ratio. In addition, sentiment analysis is shown on hover over the embedded videos, allowing a user to see what others thought of each when determining whether to watch or not. Our web application, thus, acts similar to a filter for users to YouTube videos in line with their interests, sifting out bad content at the same time.

## **II. Data**

Our complete dataset consisted of video and search metadata collected through the YouTube API via an automated grabber script implemented in Python. Specifically, we exfiltrated the video id, title, description, like count, dislike count, location longitude and latitude, and list of tags from YouTube videos uploaded from March 1st to the end of April. In addition to this, we gathered comment data from the top ten videos in order to perform sentiment analysis for some of our visualizations. Popularity scoring was calculated according to the like/dislike ratio, and videos were ranked accordingly. YouTube-8M challenge data from Kaggle was not utilized (as we intended in our final proposal) to augment and generalize our tag set from the YouTube Data API exfiltration. Our dataset, consisting of 100,000+ videos was split into

training and test samples for our final project, 90% and 10% of our cleaned video data respectively.









Two challenges with respect to extraction should be addressed, one being the difference between our theoretical assessment of the number of videos we could collect over a span of time and the actual dataset, the other being the use of YouTube-8M challenge data from Kaggle. First, the original numbers for our target video count were based on YouTube's average breakdown of uploads per day since 2012 and extrapolation of initial pulls. Our time horizon, however, that of March 1st to April 30th, represented an outlier period where upload count was at a lull. Additionally, the predicted number of hours for our grabber script to run was underestimated. As a result, our dataset turned out to consist of a much lower video count than expected. Second, the YouTube-8M challenge data didn't give us a wide enough range of tag groupings to be useful for the type of user-defined label condensing we were looking for. Although substantial coverage existed in some areas such as gaming or cosmetic tutorials, the categories were too narrow in scope for our needs. Instead, we condensed synonymous tags so as to compress the surface area of our predicted set of labels.

#### - Cleaning/Storing/Integrating:

For the title, description, and list of tags, cleaning involved stripping the strings of special characters, converting to lowercase, removing foreign languages, eliminating punctuation, and deleting links. Here, we utilized regexes as well as a natural language processing library NLTK to distinguish non-English words from English words and compare. Ultimately, we eliminated videos which made reference to foreign languages entirely. For label data only, we omitted single filler word tags such as 'the', 'and', or 'on' (which were surprisingly prevalent in the initial set). In addition and as previously mentioned above, we condensed synonymous tag data into single words or phrases using the python library Synonyms. An 'amusing' label, for instance, will be converted to 'funny' or another word of the same meaning in a text file list taken from

Pydictionary. This reduced the label set by approximately 20%, a sizable amount of compression.

Next, we map-reduced the cleaned video output to rank videos according to their popularity score (likes to dislike ratio), and for each of the top fifty videos, we exfiltrated comment data, including published text and publishing date/time. For comments as well, cleaning involved stripping the strings of special characters, converting to lowercase, removing foreign languages, eliminating punctuation, and deleting links. Comment data was separated out by date, and the most common tags present in our ranked dataset were extracted from videos with the highest popularity.

Video Metadata			
	ID	integer	
	Title	string	
	Description	string	
	Like Count	integer	
	Dislike Count	integer	
	Location	string	
	Related Tags	string	

All this was done in order to translate the initial collected data to a workable format for our visualizations and web application. We then stored this cleaned video statistics data in a MySQL database for ease of access purposes. Only video metadata relevant to our line of inquiry was processed and stored, including video id, title, description, like count, dislike count, location of post, and related tags. The database schema appears as in the figure above, where 'related tags' is a string list of user-defined labels delimited by commas.

### III. Methodology

After cleaning, integrating, and storing our collected data, the second step is to employ our model using the data to predict the tags for each of the YouTube videos we specified. This is a multilabel classification problem and to do so, we make use of multi-label learning, a model which maps inputs to binary vectors rather than to scalar outputs, as would be the case in ordinary classification problems. Similarly, in our project, video categorization almost always includes more than one tag. There are two main methods for tackling multi-label classification that exist: problem transformation methods (those which transform the entire problem set into one or more single-label classification or regression instances) and algorithm adaptation methods (those which extend current learning algorithms in order to handle multi-label data).

#### **A) Machine Learning Model:-**

We chose to do a combination of both of these approaches mentioned above. We first transform the problem set into one or more single-label classification instances. For this we transform the user defined tags which are textual in nature to a matrix with binary values where 0s indicate non-existence of a tag and 1s indicate existence of a tag. We use this as an input for our Machine Learning model.

The pipeline of this model is as follows:-

##### **1) Count Vectorizer transformation**

This transformation converts a list of strings into a matrix of counts for each token by tokenizing the strings. This leads to a sparse representation of these counts.

##### **2) TF-IDF transformation**

This transformation is very important in the context of our model. This is a statistical method which helps us to know the importance of a word in a sentence. The Term Frequency (TF) helps us know the number of times a particular term appears in a document. The Term Frequency-Inverse Document Frequency (TF-IDF) value helps us understand how frequent a particular word is in a document.

### 3) One Vs Rest Classifier with Linear SVC

The OneVsRest is our classifier with a Linear SVC estimator function.

This classifier is very useful in a multilabel classification problem. This is also known as a One Vs All classifier. This involves fitting a classifier for each class. This works well since we know that the user-defined tags are all independent of each other. Hence for each classifier, a class is fitted against all other classes to find the best fit for the data. This approach is very computationally efficient and this model is very intuitive as it does exactly what we would expect a multi-label classification model to do. This strategy is used for our model and we are able to predict multiple labels by fitting the 2-D matrix of binary values (which we discussed earlier).

The linear SVC uses a linear kernel and works well with One Vs Rest classifier. It is more flexible with respect to penalties and loss functions and also scales better to large amounts of sparse data. Hence, we have chosen this as our estimator function.

So this how we combine the 2 approaches discussed above. We first perform two transformations on our feature set which comprises of title and description (<title, description>). We then use these transformed features for our Machine Learning Model by using the One Vs Rest Classifier with Linear SVC. Since the user-defined tags are independent we have successfully reduced the dimensionality of our output space.

We have used 90% of our entire data for the training set and 10% of the entire data for the testing set.

### **B) Statistical Analysis:-**

Since many performance evaluation metrics for single-label classification fail when applied to multi-label classification, we use the following performance metrics specific to this type of model:-

- Recall:-

This computes the fraction of relevant tags that have been predicted over total number of tags present in the actual set. It find the fraction of true positives with the total of true positives and false negatives.

- Precision:-

This computes the fraction of relevant tags that have been predicted over total number of predicted tags. It find the fraction of true positives with the total of true positives and false positives.

- F1 score:-

This is computed using the precision and recall values. It performs a weighted average between precision and recall scores but we use the traditional balanced score which uses a harmonic mean of precision and recall values by multiplying the constant of 2 to scale the scores to 1 when precision and recall values are both 1.

Our training set (90%) is composed of video title and description pairs each associated with a set of labels, and the task is to predict the label sets of unseen instances with the model created by analyzing our training sample. Similarly we use 10% of the entire data for the test set.

### C) Sentiment Analysis:-

We performed two types of sentiment analysis:-

- 1) Sentiment analysis on the videos using title and description

This type of sentiment analysis was done for our web application which is a YouTube recommender system. We used Natural Language Processing technique and Natural Language Toolkit (NLTK) to perform the sentiment analysis. The feature space was the title, description and we used a

dictionary of words from NLTK to understand the context of a sentence and compute a polarity score for each sentence. We used compound scores which gives an overall average polarity score for each sentence (title, description) and then we computed the average compound score. If the score  $>0$  then it was classified as a positive video, if the score  $<0$  then it was classified as a negative video and if the score  $=0$  then it was classified as a neutral video.

2) Sentiment analysis on top ten videos using comments over a particular time span

This type of sentiment analysis was done for the visualization of our YouTube recommender system. We used Natural Language Processing technique and Natural Language Toolkit (NLTK) to perform this sentiment analysis as well. The feature space here was sparse and we used comments for a video on each day. This data is sparse due to the irregular timing of comments on videos. We used a dictionary of words from NLTK to understand the context of a comment and compute a polarity score for each comment for a particular day. We used compound scores which gives an overall average polarity score for each comment (title, description) and then we computed the average compound score for a particular day using all comments on that day. We computed this sentiment score only for the days where the videos received the comments.

The above sentiment analysis models were unsupervised learning models.

The purpose of performing these two types of sentiment analysis were:-

- For the first sentiment analysis model, the purpose is that if we find a video with a negative sentiment score  $>85\%$  then we send out a warning to the users before they watch the video. Furthermore, if we find a video with a negative sentiment score  $>95\%$  then we block the user (uploader/channel) from uploading such videos. This is very useful form

of customization and helps in a better viewing experience while watching YouTube videos.

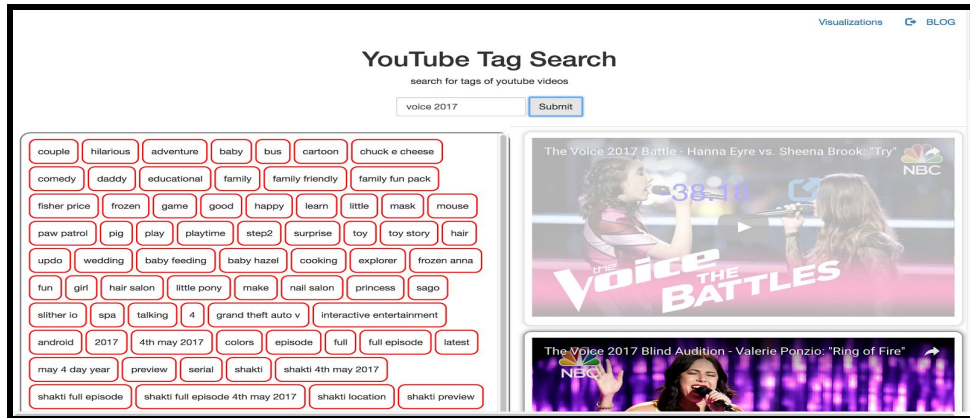
- For the second sentiment analysis model, the purpose is to analyze the sentiments of top 10 videos over a time span. We found interesting trends while performing visualizations. We found few top videos with varying sentiments which had positive sentiments initially and later they dropped towards negative sentiments. This was very helpful in finding insights of the data we had.

#### **IV. Web Application**

On top of this prediction model, we build a video recommendation web app, displaying the top 10 videos for a particular user-chosen tag. This requires Javascript, HTML, and CSS to define the UI and User Experience. As the user types within a text field, a filter is applied to a list of labels, and once a label or group of labels are clicked on, the results associated with them are embedded to the right of the user's selection. To filter tags as the user enters a query, we used Twitter's Typeahead.js library along with its suggestion engine, Bloodhound. Once the videos associated with a particular tag are loaded on the screen, the user is able to hover over the videos to view the sentiment analysis for the video. If a video is positive, the text is red. If a video is negative, the text is blue. If a video is neutral, the text is yellow. The magnitude of the numerical value for the sentiment describes how strongly the sentiment is associated with the video. This way, the user can easily see the tags that our predictive model assigned to the videos in our test set along with the sentiment associated with the video. The user is also able to click on the video to play it, and there is also a link that appears when the user hovers over the video that allows them to go to the video on YouTube if they would like to see more information about it.

To return queries for the user, we housed our data in a MySQL database. Currently, the web application can only be hosted locally but we also envisioned having a warning system in place for videos with extremely negative sentiments so it may be of consideration for us to eventually host the web application using Heroku or AWS.

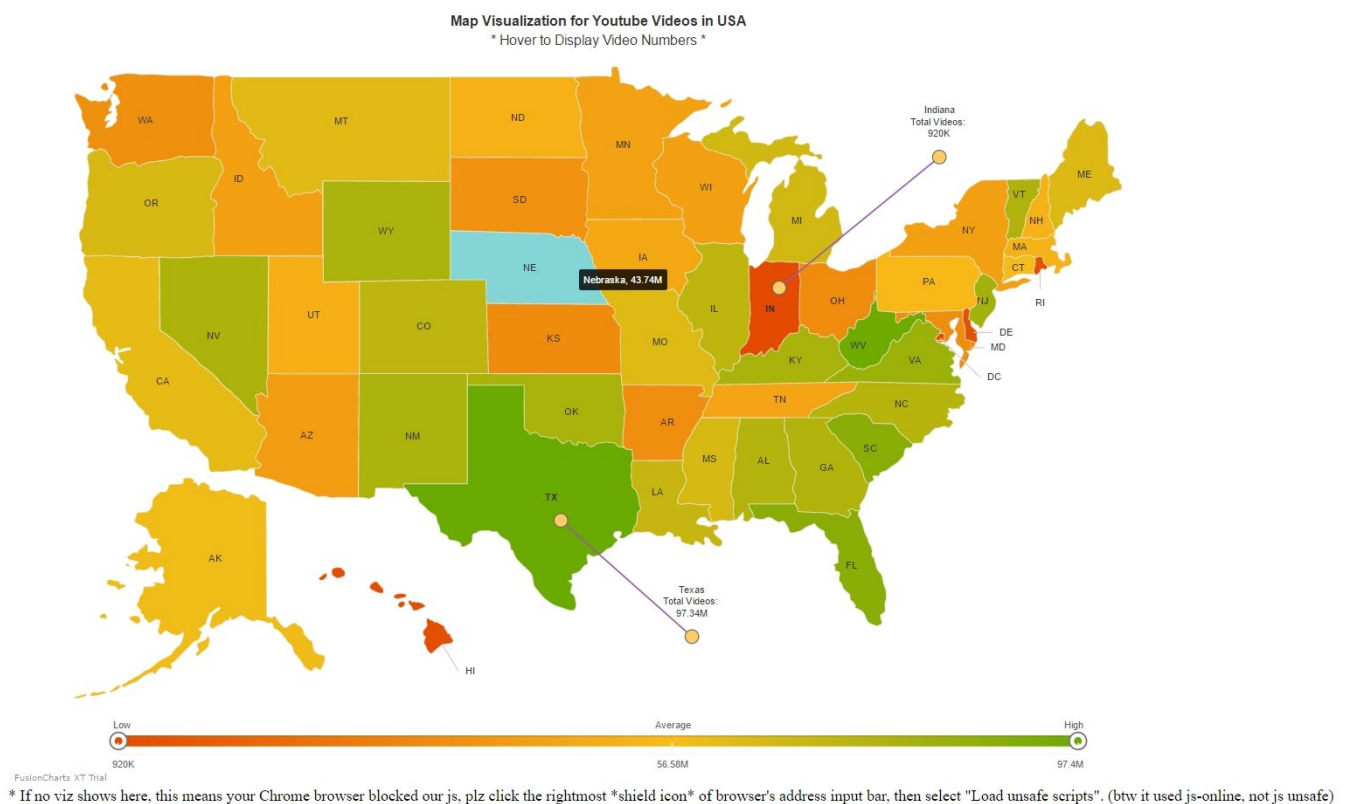




## V. Visualizations

We have done five D3 visualizations:-

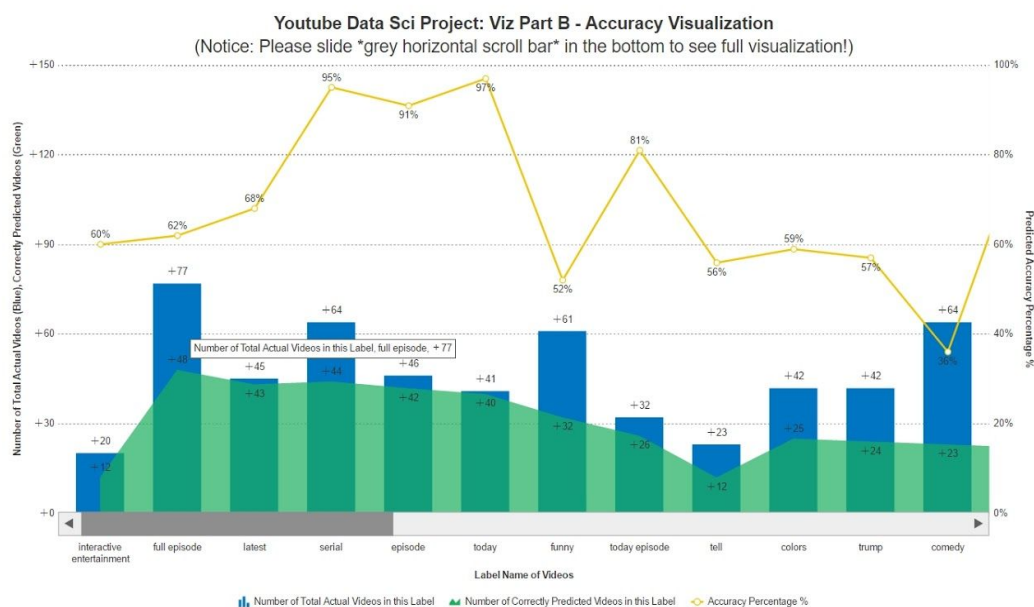
### 1. United States of America Map visualization



Overlaying a heat map of the United States of America, we show the popularity of the YouTube videos based on the number of uploads which is arranged by location, where red represents less number of uploaded videos and green represents more number of uploaded

videos. The integer value is the number of uploaded videos on YouTube which is shown on hover. The slider at the bottom is also color coded and acts like a filter for more interactivity.

- Bar chart for accuracy visualization (tags Vs actual number of videos present and number of videos present in our model)



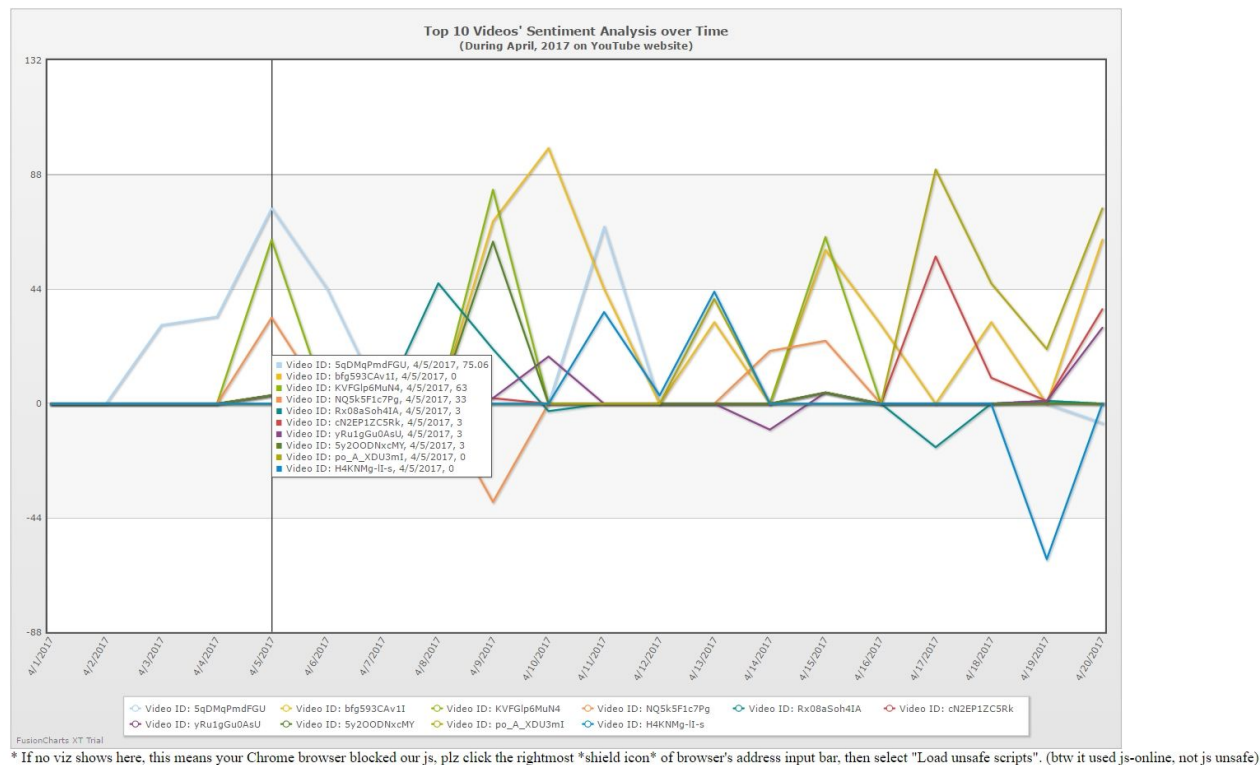
FusionCharts XT Trial

\* If no viz shows here, this means your Chrome browser blocked our js, plz click the rightmost \*shield icon\* of browser's address input bar, then select "Load unsafe scripts". (btw it used js-online, not js unsafe)

Comparison of our predicted tags to those produced by users for their own videos in the original YouTube dataset provides a performance metric visual. For each tag we have shown the actual number of videos associated with that tag Vs the number of videos which is associated with that tag in our model. This charts our deviations or lack thereof from the actual data. This would represent a good comparison between our model Vs the actual model and would show us the accuracy of our model. A slider is present to move over other tags for more interactivity and the integer value on hovering over the bars shows the actual number of videos associated with a particular tag. For further interactivity there is a filter at the bottom to see only certain parts of this visualization (actual number of videos, number of videos in our model, accuracy percentage)

### 3. Sentiment analysis visualizations (using comments):-

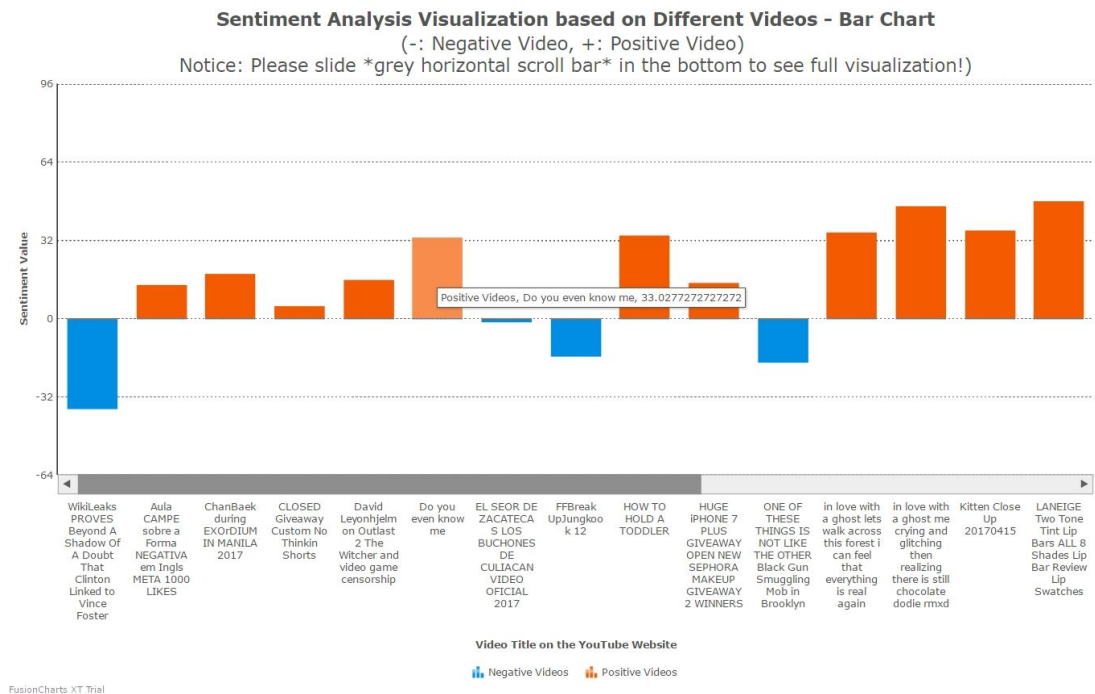
- a) Line graph to show sentiment scores for top 10 videos using comments over the days of April



Week by week beginning from the first week of April until 20th April, a line graph displays the sentiments of the videos (percentage) over time based on the comments. Interactivity is through a filter for video IDs which can be used to compare sentiments of few videos against each other. We used Natural Language Processing technique and Natural Language Toolkit to analyze the sentiment scores of comments for a particular day. The method was described above in the Methodology section under Sentiment Analysis. Thus, it shows us the daily trends of top 10 videos over time with the sentiment scores (i.e. color coded lines for each video). We also tested this model by

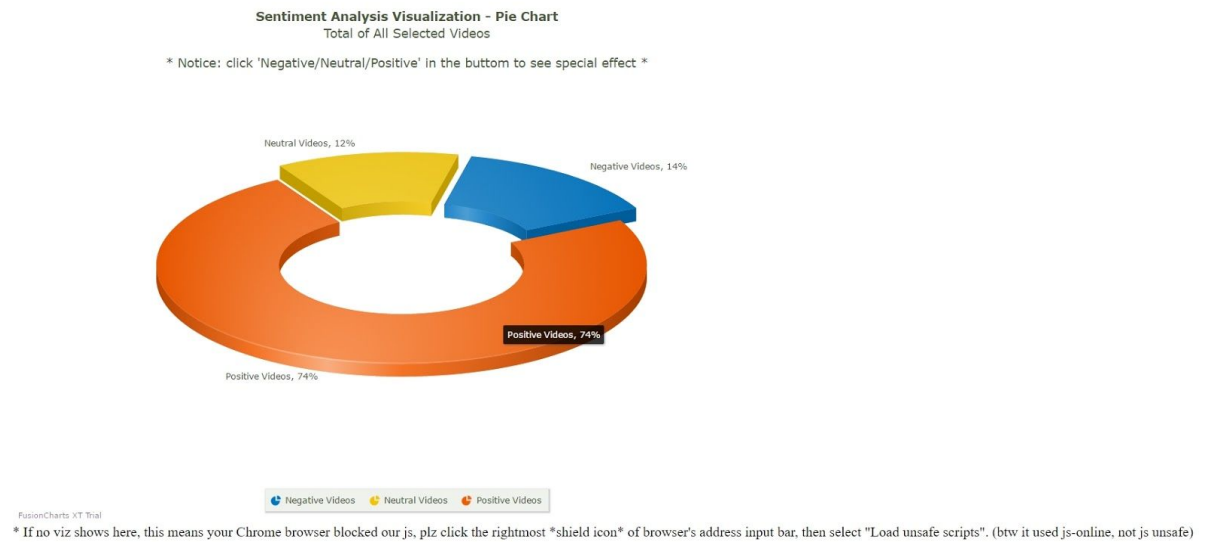
actually looking up for the video using the video ID in YouTube with a high negative sentiment score and found that our model correctly predicted this negative video. For further interactivity there is a filter at the bottom to see only particular videos with IDs.

- b) Bar chart for sentiment scores of top 50 videos using all the comments for each video [positive, negative and neutral].



We performed a similar visualization for top 50 videos using comments for each video and plotted this using a bar graph. The slider at the bottom helps in moving across for more videos with the sentiment scores (percentage). For further interactivity there is a filter at the bottom to see only positive videos or negative videos.

c) Pie chart for cumulative sentiment scores of top 50 videos using all the comments



We also found the cumulative sentiment scores for top 50 videos using the comments. We have shown this in terms of a pie chart. The interactivity is through on hover to see the percentage and on clicking the pie chart gets sliced. This pie chart can also be rotated 360 degrees. For further interactivity there is a filter at the bottom to see either positive, negative or neutral videos and the sentiment scores (percentage) by slicing the pie chart.

## VI. Impact

Since this is a customized YouTube recommendation system, it can have a lot of social impact. There are various ways in which we can have a social impact.

Here are few of them:-

- The purpose of our project was to predict tags for a video uploaded by a user. Our model achieves a precision score of 94.29% which shows that if we deploy our model then we can automate the tagging process of videos for the users.
- Our model can also be used as a recommendation system and based on the videos the users watch, we can use the tags associated with those videos and our predicted tags for unseen videos and recommend these videos to the user.

- Our model can also be used as a customization or plugin. We will give users a warning if a video has a negative sentiment score > 85% which is shown on hover in our web application. If the negative sentiment score > 95% then we would block the user from uploading videos. This will enhance the viewing experience of users in YouTube.

Hence, this Machine Learning model with Sentiment Analysis is very dynamic and flexible and can be used in different areas of tagging in YouTube.

## VII. Deliverables

As a reminder, here were our deliverables from the final proposal handin:

- 75%: This is the bare minimum that we want to accomplish for this project.

We will create a web application that returns YouTube video recommendations based on the labels that our ML algorithm assigns to videos in our dataset. The user will enter a keyword that corresponds to one of our labels, and our web app will return the top ten videos with that particular label.

- 100%: This is what we are aiming to accomplish in our project, keeping in mind realistic circumstances.

Our web application will be able to return the top ten YouTube video recommendations with Hamming Loss under 10% and average precision above 85%.

- 125%: This is what we want to be able to achieve in our project in the most ideal world where everything goes according to plan.

Our web application will be able to return the top ten YouTube video recommendations with Hamming Loss less than 2% and average precision above 95%.

## VIII. Results

We have removed the calculation of Hamming Loss as there were better performance metrics for evaluation of our model as this is a Multi-Label learning model. Apart from that we have met 100% of our goal which we had set previously. Our web application returns the top YouTube video recommendations with average precision above 85%, just shy of 95%, which was our threshold for 125% achievement.

The final results of the predicted tags for all the videos

- Recall: 70.7326%
- Precision: 94.2882%
- F1 score: 75.6736%

We tried other models for this problem and we didn't achieve better performance. This current model is also very efficient and the execution time of this model is within 10 seconds which is good with respect to the dimensionality of the data.

We tried using the Multi-Label K Nearest Neighbors model and we found it computationally inefficient and it didn't give us better results.

An alternative approach would be to use neural networks with TensorFlow package but this concept was not introduced in this course and we didn't find this as a viable option given the time constraint.

Since this was a big project with a lot of data, we had worked on separate aspects of this model as a team. Hence, we had separated out the front end source code and back end source code initially.

We have combined the 2 repositories to a new repository:-

GitHub Repository Link: [https://github.com/abhishekdutta/youtube\\_recommender\\_system](https://github.com/abhishekdutta/youtube_recommender_system)

The links to the other GitHub repositories for front end and back end source code (separately) can be found in the README.md file of the above listed GitHub repository (combined).