

Syracuse University

Engineering and Computer Science Department

“YouTube Sentiment Analysis”

Shyamal Shah SUID:

Rushang Shah SUID:

Rushi Shah SUID:

Meet Doshi SUID: 744032688

Syracuse University, NY, 13244

1. **Abstract:**

The main objective of the project is to analyze and perform data mining on YouTube. This system takes data from a YouTube channel and does data analysis on that data. Our main goal is to analyze whether a YouTube channel will grow positively or not. We take the few latest videos of the channel and analyze the comments of the viewers. This will give us a rough idea whether the viewers are liking the videos or not. This will help the creators to make improvement and can easily improve themselves, they can make more videos which get most positive response. We have used several data classifying techniques to build a trained classifier and then classify our test data. Using the classified test data, we predict the sentiment of people on the latest videos.

1. **Introduction:**

YouTube has a very large library of videos and users. YouTube has 23 million users, 300 hours of videos are uploaded every minute, almost 5 billion videos and gets over 30 million users per day. This makes us analyze the sentiment of people who give their opinion on the videos by commenting on it. This will give us an idea about the future scope of the channel. We analyze the positivity and negativity in the comments by the users on videos of a channel and find out the overall impact of videos on the users. This will predict whether a channel will grow or not. Our classifier classifies into three categories: Positive, Negative and Neutral.

Given a group of sentences or paragraphs, which was used as a comment by a user in an online platform, our task is to classify them in one or more of the following categories – Positive, Negative or Neutral.

1. **Problem Statement**

Today there a lot of creators on YouTube right now, but the problem is that they don’t know how to become more efficient. Most of them are confused on why their videos are not getting views or why are they getting negative response. Our goal is to analyze the videos and give output on what is getting a positive response, and which one is getting the negative. By which they can improve their content accordingly.

Given a group of sentences or paragraphs, which was used as a comment by a user in an online platform, our task is to classify it to belong to one or more of the following categories – Positive, Negative or Neutral. Comments are highly unstructured and also non grammatical and some of them are out of vocabulary words, Lexical variation and extensive use of acronyms.

The challenges that we face are :

1. Comments are highly unstructured and also non-grammatical.

eg.

$$$get free Galaxy8$$$

>>>http://bit.ly/2qrKmku<<<

2. Some of them are out of vocabulary words.

eg.

"I want to start ahs, but ointb but the mentalist ,hmmm"

3. Lexical variation and extensive use of acronyms.

eg.

"Trying to brace maself coz tmrw is result. Come asap"

"Lol"

1. **Process Flow**

YouTube Dataset

Pre-Processing

Classifier training

Feature Extraction

Classification

1. **Analysis:**

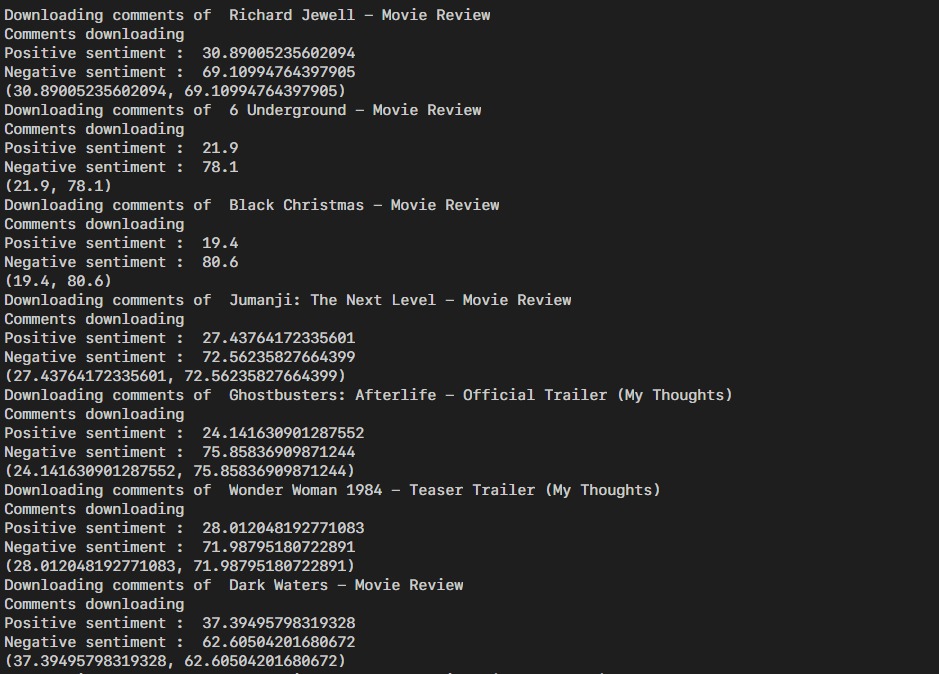
We will start with the dataset and analyze different attributes linked to the data

* 1. **Dataset:**
* We use YouTube data v3 search data API to fetch 50 most recently uploaded video’s meta data set for the channel.
* The YouTube Data API (v3) lets you incorporate YouTube functionality into your own application. You can use the API to fetch search results and to retrieve, insert, update, and delete resources like videos or playlists.
* In conjunction with the YouTube Player APIs and the YouTube Analytics API, the API lets your application provide a full-fledged YouTube experience that includes search and discovery, content creation, video playback, account management, and viewer statistics.
* We have used the get request

GET/activities - Returns a list of channel activity events that match the request criteria. For example, you can retrieve events associated with a particular channel or with the user's own channel

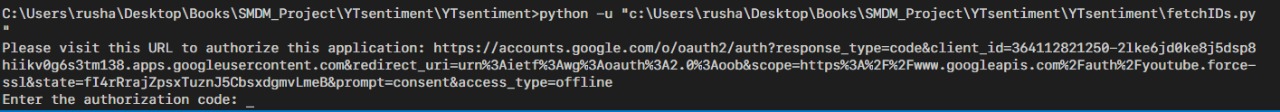


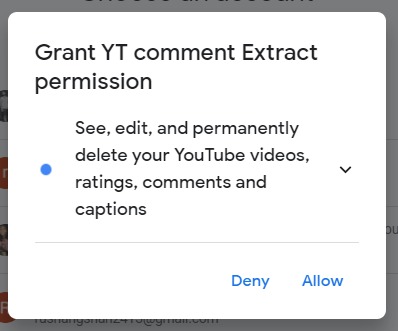
YouTube Dataset

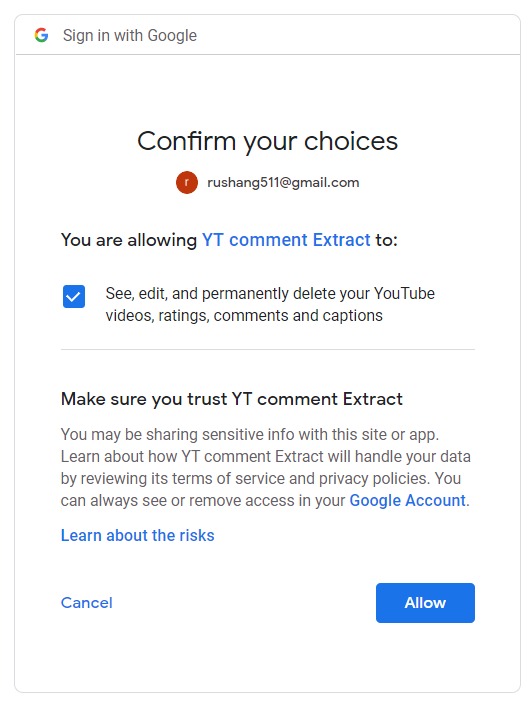
Comment Dataset

* + 1. **Authentication Steps for YouTube Data collection**

As we run the file to fetch the ID of the videos from YouTube it asks for an authentication code which is received, when we give permission on our Google account to allow us to use YouTube API for extraction of Id’s for videos and then to retrieve the comments of the videos.







**5.2 Sentiment Analysis**

Sentiment analysis is used to identify the polarity of text and classify it as positive, negative or neutral. This work is done based on the polarity score of a text; a score of zero identifies a neutral text, positive score corresponds to a positive text and negative score represents a negative text. Sentiment Analysis consists of various steps to generate useful insight. The very first step involves collecting valid data. This step is followed by performing some preprocessing on the text like removing numbers, stop words, punctuations, etc. We basically try to eliminate all the words that will contribute almost nothing in analyzing the sentiments of the text. The next step is to decide a process to follow to classify sentiments. The common use cases involved in analyzing sentiments are machine learning based, lexicon-based and a hybrid use case that includes both machine learning in conjunction with lexicon-based approach. After we decide upon the approach, we are left with using the approach to classify the text based on the previously selected approach.

1. **Pre-Processing and Classifier training**

* We tokenize every comment using NLTK Word Tokenizer.
* The NLTK is a Python library written for working and modeling text.
* It provides good tools for loading and cleaning text that we can use to get our data ready for working with machine learning and deep learning algorithms.
* Cleaning the data
* Split into Sentences - NLTK provides the sent\_tokenize() function to split text into sentences.
* Split into Words - NLTK provides a function called word\_tokenize() for splitting strings into tokens.
* Filter out punctuation - This can be done by iterating over all tokens and only keeping those tokens that are all alphabetic. Python has the function isalpha that can be used.
* Filter out Stop Words - NLTK provides a list of commonly agreed upon stop words for a variety of languages, such as English.
* Naïve Bayes classifiers are a family of simple "[probabilistic classifiers](https://en.wikipedia.org/wiki/Probabilistic_classification)" based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with strong (naïve) [independence](https://en.wikipedia.org/wiki/Statistical_independence) assumptions between the features. They are among the simplest [Bayesian network](https://en.wikipedia.org/wiki/Bayesian_network) models.
* Naïve Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the [text retrieval](https://en.wikipedia.org/wiki/Information_retrieval) community in the early 1960s. and remains a popular (baseline) method for [text categorization](https://en.wikipedia.org/wiki/Text_categorization), the problem of judging documents as belonging to one category or the other (such as [spam or legitimate](https://en.wikipedia.org/wiki/Spam_filtering), sports or politics, etc.) with [word frequencies](https://en.wikipedia.org/wiki/Bag_of_words) as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine). It also finds application in automatic [medical diagnosis](https://en.wikipedia.org/wiki/Medical_diagnosis).
* Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. [Maximum-likelihood](https://en.wikipedia.org/wiki/Maximum-likelihood_estimation) training can be done by evaluating a [closed-form expression](https://en.wikipedia.org/wiki/Closed-form_expression), which takes [linear time](https://en.wikipedia.org/wiki/Linear_time), rather than by expensive [iterative approximation](https://en.wikipedia.org/wiki/Iterative_method) as used for many other types of classifiers.
* In the [statistics](https://en.wikipedia.org/wiki/Statistics) and [computer science](https://en.wikipedia.org/wiki/Computer_science) literature, naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes. All these names reference the use of Bayes' theorem in the classifier's decision rule, but naïve Bayes is not (necessarily) a [Bayesian](https://en.wikipedia.org/wiki/Bayesian_probability) method.

1. **Feature Extraction**

* We create NLTKs BigramAssocMeasures and BigramCollocationFinder to create Bigrams. Bigrams are a pair of consecutive written units such as letters, syllables, or words.
* Finding collocations requires first calculating the frequencies of words and their appearance in the context of other words. Often the collection of words will then be requiring filtering to only retain useful content terms. Each ngram of words may then be scored according to some association measure, in order to determine the relative likelihood of each ngram being a collocation.
* The ``BigramCollocationFinder`` class provide these functionalities, dependent on being provided a function which scores a ngram given appropriate frequency counts. A number of standard association measures are provided in bigram\_measures.
* Now we check if any classifier is already trained for the particular bigram. If there are no pre-trained classifiers, then we create a new classifier via out training data set.

1. **Classification**

* First, the collected data in form of statements are divided into bigrams.
* Then we classify these bigrams into majorly three categories: positive, negative or neutral, using trained classifier.
* Once all the bigrams are classified, we use the vote classifier to judge the overall sentimental of each comment. This takes each comment of the video and gives the average sentiment of people on the videos.
* The EnsembleVoteClassifier is a meta-classifier for combining similar or conceptually different machine learning classifiers for classification via majority or plurality voting. (For simplicity, we will refer to both majority and plurality voting as majority voting.). The EnsembleVoteClassifier implements "hard" and "soft" voting. In hard voting, we predict the final class label as the class label that has been predicted most frequently by the classification models. In soft voting, we predict the class labels by averaging the class-probabilities (only recommended if the classifiers are well-calibrated).
* Then each comment of a video is classified using the naïve Bayes classifier as positive and negative and then the average sentiment is calculated.