Beyond Likes and Dislikes: Unveiling YouTube Comment Sentiment

Jasdeep Singh Jhajj
School of Information
University of Arizona
jasdeepjhajj@arizona.edu

1 Introduction

The sudden and exponential growth of Youtube has led to the creation of active communities centered around content created by users. Comments on Youtube, specifically, provide a valuable source of information showcasing viewer thoughts, interaction frequency and general feelings towards a video. Examining these feelings can be extremely beneficial for creators, growing companies and researchers.

Understanding emotions from YouTube comments, which is a casual online text expression way is tricky because people might use some informal ways of text, emojis, or some common social media slang to express themselves. It is quite a tedious job if we have to build a team to manually analyze each and every comment, as every second thousands of videos get uploaded and also it can be prone to biasing and inconsistency in results. Using a mixture of machine learning techniques from natural language processing and supervised learning[1] the model can handle the informality of the comments and even slangs being used in a phrase and analyze large volumes of comment data at once and categorize it into positive, negative, or neutral sentiment. One of the commendable work done in this field by Makhmudah et al. [2](2019), the authors used the support vector machine (SVM) algorithm for analysing sentiment of Tweets related to homosexuality in Indonesia, and achieved an accuracy around 99

This project focuses on building a model that can precisely categorize the sentiment of Youtube comments. Firstly, the Youtube Data API[3] is used to gather comments from selected videos. As there will be use of casual language in Youtube comments, it can be tackled with proper data preprocessing to include text normalization, stop word removal, tokenization, and lemmatization[4]. Studies like Khyani and B S (2021)[5] delve into lemmatization and stemming, enriching our comprehension of text preprocessing in Natural Language Processing.As the data scraped will not be pre labeled to achieve data labeling I will then use Valence Aware Dictionary and Sentiment Reasoner(VADER)[6] to label the comments as positive, negative, or neutral for training purposes. Once trained, the evaluation parameter for the model can be based on accuracy, precision, and recall will be carefully assessed to evaluate its performance.

With a successfully trained sentiment analysis model, it can contribute to growth of small-scale brands, content creators to understand their target audience sentiment better, how their audience views their content, and eventually improve their content strategies. Hence, the project will be able to create a more meaningful and impactful online experience for creators and viewers.

2 Methods

2.1 Data Handling

2.1.1 Data Collection

The data collection for the Comments analysis is performed using the YouTube Data API v3[3] using googleapiclient package. Google Cloud Console[7] helps with the API key which was required to fetch the required information from the Youtube from a specific YouTube video ID, relevant comment data, including author name, comment text, publication time, and likes count were extracted from the API. The user is provided with interface to input a YouTube Video URL, from which the comments data is collected at runtime. The model trained and tested on the videos which have 9000 comments so that the model is properly trained without any kind of underfitting or overfitting.

2.1.2 Data Preprocessing

Data preprocessing is performed on the raw extracted comments to ensure data cleanliness and uniformity. The Youtube comments column in our dataset is cleaned by removing any non english character like special characters, numerical values, emojis/emoticons, converting to lowercase, and duplicate values and missing values are dropped appropriately. Finally, the comments are tokenized into separate words and lemmatization to make the analysis of feelings more efficient.

2.1.3 Dataset Description

The scraped dataset contains information about comments on a YouTube video. Here's a brief description of each column:

Author_Name(<str>): Usernames of individuals who posted the comments, starting with "@" followed by the username.
Comment_Text(<str>): It contains the actual text content of the comments posted by the users.
Updated_Time(<str Timestamp>): Timestamps indicating when the comments were posted in the format "YYYY-MM-DDTHH:MM:SSZ".
Likes_Count(<int>): Number of likes received by each comment.

Table 1: Scrapped YouTube Comments Dataset

Index	Author Name	Comment Text	Updated Time	Likes
1	@spartanshivam2837	Nyce	2024-04-11T06:25:59Z	0
2	@devsharma070	Delious	2024-04-11T04:49:49Z	0
3	@abhi9800	Wow.. best foods that I can think of. Great to...	2024-04-11T04:41:35Z	0
4	@Noobert	watched this while eating korma thinking about...	2024-04-07T23:33:25Z	0

2.1.4 Data Labeling for Sentiment Analysis

As the data extracted from Youtube API is not labelled, to analyse the sentiment we use the VADER[6] sentiment analysis tool from the NLTK library, function used to evaluate the comments polarity is *polarity_scores*. VADER sentiment analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores. The compound polarity score is calculated by summing the valence scores of each word in the lexicon. The threshold based from the research of Hutto and Gilbert [8](2015). If the compound score >= 0.05 then the text is labeled positive sentiment, value of compound score <= -0.05 is considered negative sentiment and others are neutral sentiment. By running this tool on

every processed comment, we determine sentiment scores such as positive, negative, neutral, and compound. We categorize each comment's sentiment as good, negative, or neutral based on a compound score.

2.2 Model & Algorithm Description

The model is created using scikit-learn's Pipeline class which includes two steps TfidfVectorizer and SVC(Support Vector Classifier). The first step of vectorizer, converts the text data using TF-IDF representation. It converts a collection of text documents into a matrix of TF-IDF features. The sentiment classification is done using the Support Vector Machine (SVM) algorithm[9]. The choice to use SVM was based on its ability to handle high-dimensional data and non-linear relationships useful for text classification. SVM finds an optimal hyperplane which can separate data points in a high-dimensional space into different classes. The transformation of input data to higher-dimensional space is achieved by kernel function, which can be of different type like linear, polynomial, radial-basis-function. The SVM model is trained on tokenized comments converted into numerical feature vectors, enabling it to recognize patterns in text and categorize comments into positive, negative, or neutral sentiments.

The dataset collected from Youtube comments is unbalanced in most of the cases, so one of the class becomes a minority class and the model is not able to learn that class properly, which leads to misclassifications in this case. So, to balance out all the classes a RandomOverSampler[6] is use to over-sample the minority class by picking samples randomly.

Algorithm 1 TrainSVM

procedure TrainSVM(x_{train}, y_{train}, C_{values}, kernel_types, num_{folds})
    Initialize empty list to store the performance metrics of each model
    for each C **in** C_{values} **do**
        for each kernel_type **in** kernel_types **do**
            Initialize SVM model with parameters C and kernel_type
            Initialize empty list to store performance metrics of each fold
            for fold = 1 **to** num_{folds} **do**
                Split training data into training/validation sets using k-fold CV
                Train the SVM model on training set (x_{train}, y_{train})
                Evaluate the model on the validation set and calculate performance metrics
                Add the performance metrics to the list of fold metrics
            end for
            Calculate current model's average performance metrics over all folds
            Add the average performance metrics to the list of model metrics
        end for
    end for
    Find the model with the best average performance metrics
    Train the SVM model with the best hyperparameters on the entire training data
    return Trained SVM model with the best hyperparameters
end procedure

2.3 Evaluation Procedure

The performance evaluation for the sentiment analysis model is performed by splitting the dataset into training, training a Support Vector Machine (SVM) classifier on the TF-IDF vectorized comments[10] and testing sets and testing our model on unseen testing data then measuring metrics like, accuracy, precision, recall, and F1-score.Accuracy can be explained as a ratio of correctly predicted instances to the total instances. Precision is calculated as the ratio of predictions that should be positive to the total positive predictions made. Recall is used to check the correctness of predicted positive instances out of all positives. F1-score

combines the precision and recall into a single metric and is calculated as harmonic mean of precision and recall.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.4 Hyperparameter Tuning

The accuracy of the Sentiment analysis SVM trained model can be maximized using the Hyperparameter tuning which requires optimizing the regularization parameter and the kernel coefficient. Grid search[11] helps us search through a parameter grid, evaluating multiple possible combinations using cross-validation to find the optimal set of hyperparameters that maximizes the model's performance.

3 Results

Labeled Data Distribution After preprocessing the extracted comments from the API and labeling them, we have a distribution of sentiments in the labeled data as shown in the plot below which signifies distribution of Negative comments is 18.2%, Neutral comments 35.2% and rest of the comments are Positive comments 46.6%. Sentiment distribution plot implies that for the given Youtube Video most of the comments belong to Positive sentiment, which is followed by Neutral and Negative sentiments.

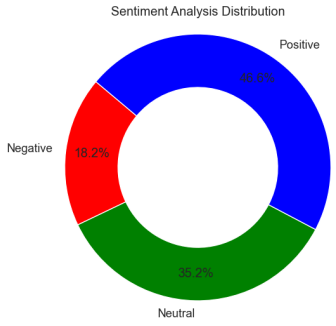
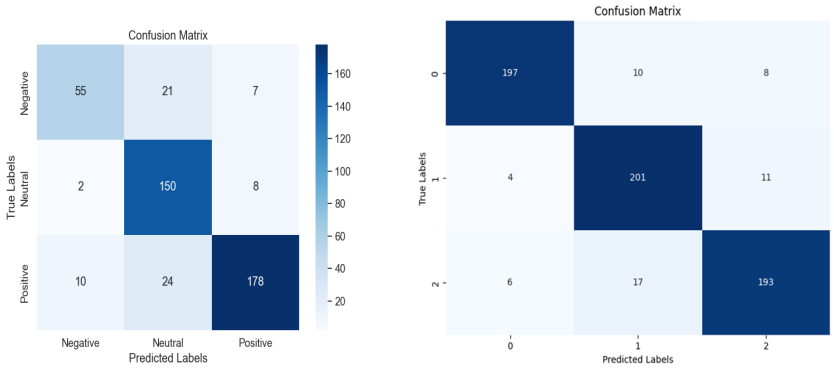


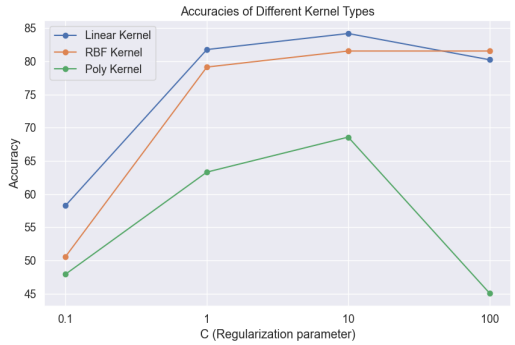
Figure 1: Sentiment Distribution

Model Performance: The Support Vector Machine (SVM) model trained on the labeled data is evaluated on parameters like accuracy, precision[12], F1-score[13]. The model achieved an accuracy of approximately 84.17% on the test set which can be seen on the Result dashboard. This indicates that the well trained SVM model can effectively classify the sentiment of YouTube comments into one of the three categories: Negative, Neutral, or Positive with high reliability.



(a) Confusion Matrix

(b) Confusion Matrix



(c) Kernel Accuracies

Figure 2: Confusion Matrix and Different Kernel Accuracies

Confusion Matrix Analysis: The confusion matrix provides insights into the model's performance across different sentiment categories. It shows how many instances of each sentiment category were correctly or incorrectly classified. The diagonal element with higher values indicates correct classifications whereas Off-diagonal elements represent misclassifications. From fig2(a), the comments that were correctly classified as Negative(True Negative) were 55 whereas there were some comments which were misclassified into Negative(21 Neutral comments and 7 Positive comments) which are considered False Positive for Negative Sentiment case. Similarly for the Neutral Sentiment we observe 150 comments were correctly classified as Neutral comment with a barely minimum False Negative in this case(2 Negative comments and 8 Positive comments). At last, for the Positive Sentiment, comments that are classified appropriately were 178 with a False Negative(10 Negative comments and 24 Neutral comments) being misclassified. Looking at figure 2(b), now that an oversampler is encorporated, the trained model will be learning equally about each class so it is verified by the diagonal elements being of almost equal range and the misclassification is reduced by large margin and diagonal elements have increased.. The confusion matrix plot provides a visual representation of the model's performance in terms of correct and incorrect classifications for each sentiment category. It helps in identifying which sentiments are being confused with others and where the model's strengths and weaknesses lie. For instance if we look at the Negative Comment are misclassified by a significant amount like 21 Neutral comments and 7 Positive comments are classified as Negative.

Accuracy for Different Kernel Types and Regularization Parameters: This plot shows accuracy for the SVM model when using different kernel types (linear, RBF, polynomial) across different values of the regularization parameter which is symbolized by C[14] in this case. The regularization parameter (C) is plotted on x-axis and accuracy of each kernel type model plotted on y-axis. It helps to understand the relationship between regularization

strength and model accuracy for three kernels. It helps in determining the optimal value of C that maximizes the model's performance.

Classification report

Table 2: Classification Report for SVM model

Class	Precision	Recall	F1-Score	Support
Negative	0.70	0.76	0.73	94
Neutral	0.86	0.92	0.88	213
Positive	0.93	0.84	0.88	225
Macro Avg	0.83	0.91	0.81	532
Weighted Avg	0.86	0.92	0.84	532

The classification report reveals the model's precision, recall, and F1-scores across negative, neutral, and positive sentiment categories. With an overall accuracy of 84%, the model performs well in classifying comments, Precision values of 0.70, 0.86, and 0.93 for negative, neutral, and positive sentiments, respectively, indicate the model's accuracy in classifying sentiments within each category. However, the classification for Negative comments seems to require an improvement as the precision of 70% indicates the model includes many false positives. Recall is particularly high for Neutral comments (92%), indicating the model effectively captures most actual Neutral comments. For Negative comments, the recall of 76% suggests there might be more false negatives compared to other classes which aligns with the discussion on Precision in classifying Negative comments. F1-score provides a more balanced perspective on the model's performance for each sentiment class. All F1-scores hover around 0.87, indicating a good equilibrium between precision and recall. The Support values, which represent the number of instances in each class, we observe 94 instances of negative sentiments, 213 instances of neutral sentiments, and 225 instances of positive sentiments.

4 Discussion

The sentiment analysis model based on Support Vector Machine (SVM) results in terms of accuracy and precision and F1-scores signify the performance in categorizing sentiments expressed in YouTube comments. Although the effectiveness of model, the confusion matrix highlights specific sentiment misclassifications, offering valuable insights for refining the model. The value of Precision denotes that the model is well trained and able to determine the sentiment with an accuracy of 84%. Although the precision for Negative is not commendable, but for the Neutral and Positive ones it is quite effective. Observing the accuracy curve for various kernel types, we observe consistent performance with the linear kernel across different regularization parameter values (C). Linear kernel performs well in most of the cases with the accuracy peak at parameter value of 10. However, the RBF and polynomial kernels exhibit fluctuations in accuracy. The accuracy vs parameter plot serves as a deciding tool for fine-tuning the SVM model by identifying the optimal regularization parameter. In conclusion, the results and analysis display the effectiveness of the SVM model for the sentiment classification on YouTube comments with a possibility of additional improvements especially those targeting distinct misclassification patterns.

Results Dashboard

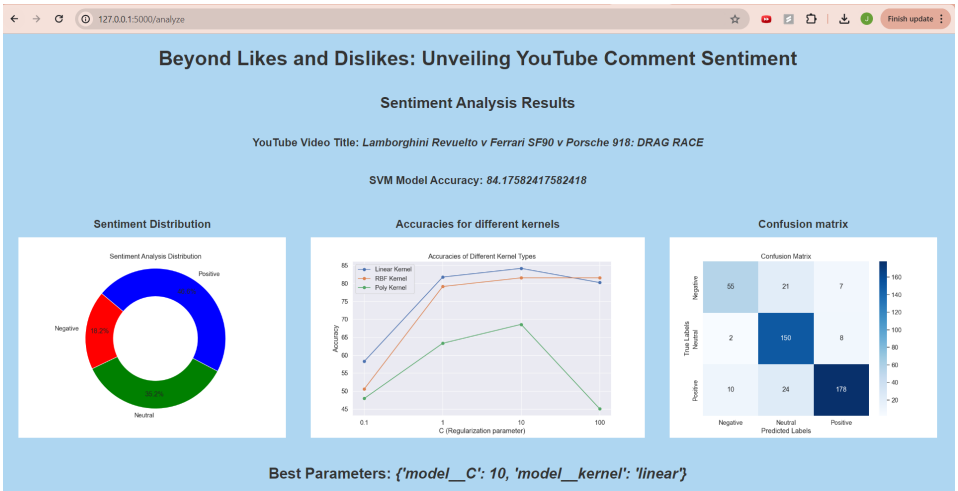


Figure 3: Youtube Comment Analysis Results Dashboard

.

References

[1] Ameen Abdullah Qaid Aqlan, Manjula, and Lakshman Naik. A Study of Sentiment Analysis: Concepts, Techniques, and Challenges. In: *Proceedings of International Conference on Computational Intelligence and Data Engineering*. Ed. by Nabendu Chaki, Nagaraju Devarakonda, Anirban Sarkar, and Narayan C. Debnath. Singapore: Springer Singapore, 2019, pp. 147–162.

[2] Umroh Makhmudah, Saiful Bukhori, Januar Putra, and Bratasena Yudha. Sentiment Analysis Of Indonesian Homosexual Tweets Using Support Vector Machine Method. In: Oct. 2019, pp. 183–186.

[3] Sanchhaya Education Pvt. Ltd GeeksforGeeks. *Youtube Data API*. URL: https://www.geeksforgeeks.org/youtube-data-api-set-1/.

[4] Yuxing Qi and Zahratu Shabrina. Sentiment analysis using Twitter data: a comparative application of lexicon- and machine-learning-based approach. In: Singapore: Springer Singapore, 2023, pp. 147–162.

[5] Divya Khyani and Siddhartha B S. An Interpretation of Lemmatization and Stemming in Natural Language Processing. In: *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology* 22 (Jan. 2021), pp. 350–357.

[6] Sandeep Panchal Published in Analytics Vidhya. *Sentiment Analysis with VADER- Label the Unlabelled Data*. URL: https://medium.com/analytics-vidhya/sentiment-analysis-with-vader-label-the-unlabeled-data-8dd785225166.

[7] Google for Developers. *YouTube Data API Overview*. URL: https://developers.google.com/youtube/v3/getting-started.

[8] C.J. Hutto and Eric Gilbert. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In: Jan. 2015.

[9] Ankit Soni. Text Classification Feature extraction using SVM. In: *International Journal of Innovative Research in Computer and Communication Engineering* 7 (July 2019), pp. 3563–3569.

[10] Sevilay Kilmen and Okan Bulut. *Okan Bulut: Text Vectorization Using Python: TF-IDF*. 2022.

[11] Nik - datagy. *Hyper-Parameter Tuning using Grid Search*. URL: https://datagy.io/sklearn-gridsearchcv/.

[12] Analytics Vidhya. *Precision and Recall in Machine Learning*. URL: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/.

[13] Nirajan Acharya. *Understanding Precision, Recall, F1-score, and Support in ML Evaluation*. URL: https://medium.com/@nirajan.acharya666/understanding-precision-recall-f1-score-and-support-in-machine-learning-evaluation-7ec935e8512e#:~:text=F1%20score%20combines%20precision%20and,false%20positives%20and%20false%20negatives..

[14] Michal Aibin. *C Parameter in Support Vector Machines*. URL: https://www.baeldung.com/cs/ml-svm-c-parameter.