CS-GY 9223 (E) – MINNING MASSIVE DATASET

FINAL REPORT

Under the Supervision of Prof. Gustavo Sandoval

Project Title: Youtube Comment Analysis

By

Name	Net ID / N Number
Eashan Kaushik	ek3575 / N19320245
Rishab Singh Redhu	rs7623 / N18032325
Srijan Malhotra	sm9439 / N18390405

NEW YORK UNIVERSITY

Spring 2022

Table of Contents

1. Problem Statement and Hypothesis.

1.1. Problem Statement:

In this project, we will not only prepare our dataset of YouTube comments but also will classify these comments into three categories namely Positive, Negative and Neutral. The data will be fed into Valence Aware Dictionary for Sentiment Reasoning (Vader) algorithm this will give us the polarity of sentiment and intensity of emotion. Using these parameters we will assign each comment to the aforementioned categories. Now that we have target labels present in our dataset, we plan to train a supervised learning algorithm like LSTM on the labeled data.

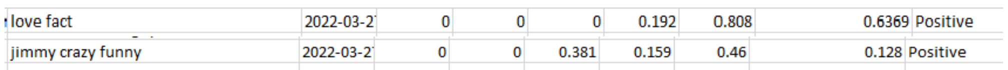
The end goal of the project would be to provide the user with a UI interface in which they can upload an URL of any YouTube video and the application will classify the comments into the above-mentioned labels, giving the user an idea of how the video might be in terms of quality.

1.2. Hypothesis:

YouTube comments can be categorized into three labels:

1. **Positive**: Comments which are meant to be in the support of the video or YouTuber will belong to this label.

Example:

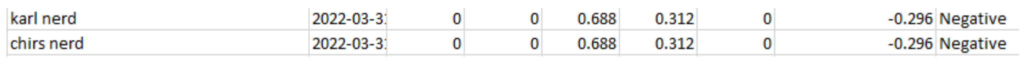


love fact	2022-03-2	0	0	0	0.192	0.808	0.6369	Positive
jimmy crazy funny	2022-03-2	0	0	0.381	0.159	0.46	0.128	Positive

Fig 1. Sample Positive Comments.

2. **Negative:** Comments which are meant to be against the video or YouTuber will belong to this label.

Example:

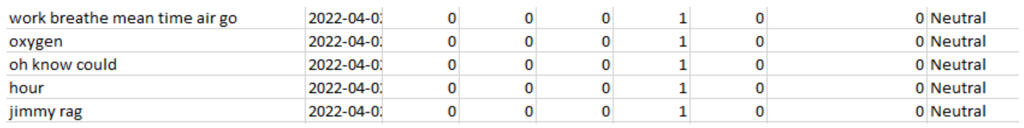


karl nerd	2022-03-3	0	0	0.688	0.312	0	-0.296	Negative
chirs nerd	2022-03-3	0	0	0.688	0.312	0	-0.296	Negative

Fig 2. Sample Negative Comments.

3. **Neutral:** Many viewers leave comments which are unrelated to the video or classify the video as 'okay'. All these comments will belong to this label.

Example:



work breathe mean time air go	2022-04-0	0	0	0	1	0	0	Neutral
oxygen	2022-04-0	0	0	0	1	0	0	Neutral
oh know could	2022-04-0	0	0	0	1	0	0	Neutral
hour	2022-04-0	0	0	0	1	0	0	Neutral
jimmy rag	2022-04-0	0	0	0	1	0	0	Neutral

Fig 3. Sample Neutral Comments.

The initial hypothesis is that any YouTube video can be easily classified into three of these categories and, we have enough evidence to classify our comments into these categories. Our ML algorithms like LSTM, SVM, and ensemble algorithms should be able to successfully classify the sentiments. We expect to see the best results in LSTM networks, as they are state of the art and have been industry-proven to perform well in NLP tasks.

2. Detailed Description of the Dataset.

2.1. Description of the Dataset and how it was obtained:

The dataset we are using was scrapped by us. This data consists of 5 columns Name of the Author of the Comment, the Comment itself, the Time at which the comment was posted, likes on the Comment, and, Reply to the comment. These comments are exclusively from the most popular videos on Mr. Beast Channel and Mr. Beast Gaming Channel.

To get the dataset for this project we developed a pipeline using YouTube API deployed on AWS for comment mining. This code is responsible for data scrapping on the server-side with no overhead to the client. Till now we have managed to mine over 1 million comments. Most of the comments in the video are neutral and positive, and not many negatives.

This data is stored in a private S3 bucket; therefore, nobody would be able to view the scrapped comments from S3. We have provided a Kaggle link for the Dataset, Dataset.

Steps to get scrap data: -

- Get the code of the video you want to scrap.
- Enter the code in the URL Tab.
- Copy the request ID and go to the request tab.
- Enter the request and hit submit.
- If the data is scrapped it will show data is scrapped or else the request is pending.

Pipeline for obtaining the data: -

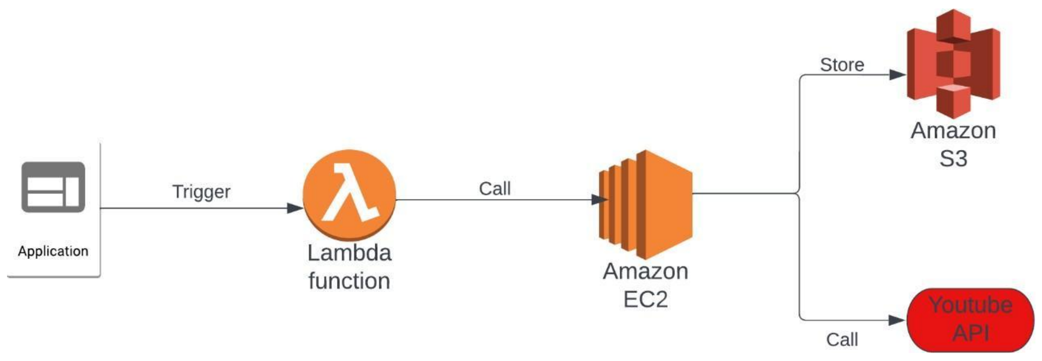


Fig 4. Structure for Scrapping Data.

2.2. Basic Statistics:

We have scrapped over 1,507,306 comments for our problem statement. These comments are exclusively from the most popular videos on Mr. Beast Channel and Mr. Beast Gaming Channel. Each row in the dataset constitutes 5 columns namely the Name of the Author of the Comment, the Comment itself, Time at which the comment was posted, likes on the Comment, and Reply to the comment.

As you can see, we do not have the target variable for our problem statement, the sentiment of the comment at this time. And if we choose to solve the given problem, we can use unsupervised learning algorithms like Clustering for classifying the comments into Positive, Negative, and Neutral. However, in the following paragraphs, we have discussed that clustering did not give us adequate results. Thus to assign labels to the dataset we have used an NLP algorithm called Vader. In the following graph, you can see the distribution of comments among the target variables.

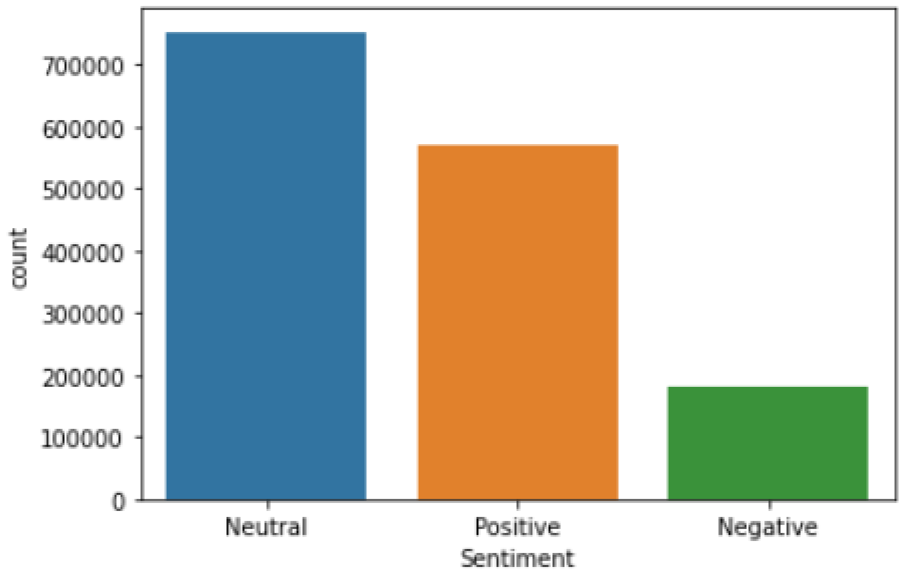


Fig 5. Dataset Features.

This type of distribution is expected. The majority of the comments on any YouTube Videos are meaningless, and most of the time are not talking about the Video itself. For example, comments like an advertisement for other YouTube channels, comments like ‘Hi’, ‘First’, etc. . Since Mr. Beast is one of the most popular YouTubers and most of the videos are about helping individuals, the sentiment around his videos is Positive. However, that being said, some viewers are of the mindset that giving people an insane amount of money might be degrading sometimes. All these claims are proven in Figure 5.

2.3. Initial Exploratory Data Analysis (EDA) phase.

On our first exposure, we realized that the data we have is very vast which is a good thing since it will give us a better overall result but it would also require a lot of cleaning as it is in its raw form. There are several stop words, repetitions, and gibberish which would require a lot of preprocessing to clear and get meaning out of which in turn would give us the polarity score we are looking to calculate. The cleaning step is mentioned in the pre-processing step.

Word Cloud

In this section we performed EDA to get better insights into our data. Initially, we developed three-word clouds for three sentiments we are trying to classify. These word clouds can be seen as follows:

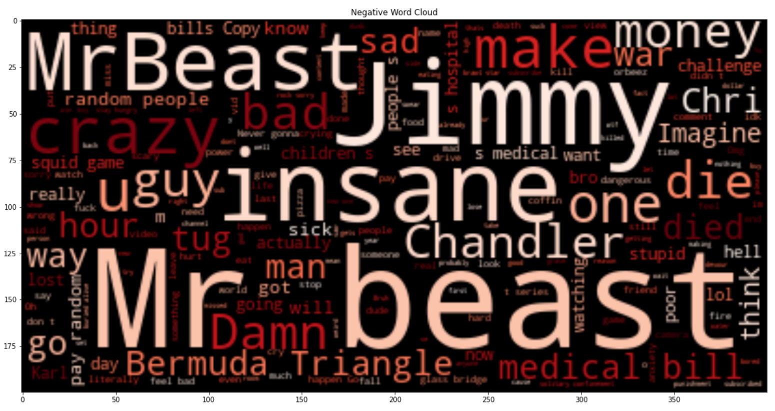


Fig 6. Word Cloud for Negative Comments

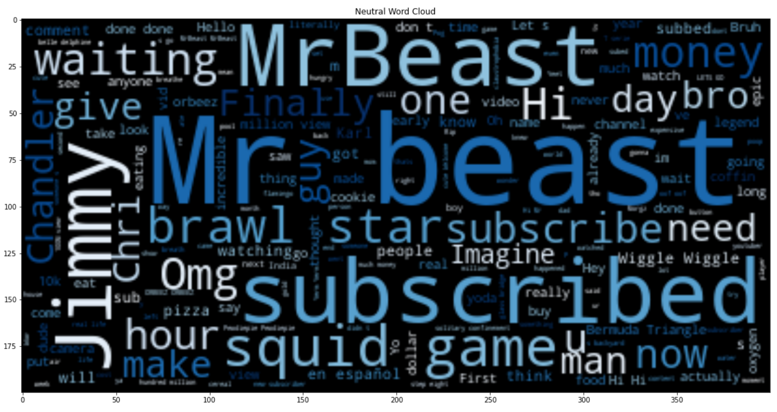


Fig 7. Word Cloud for Neutral Comments

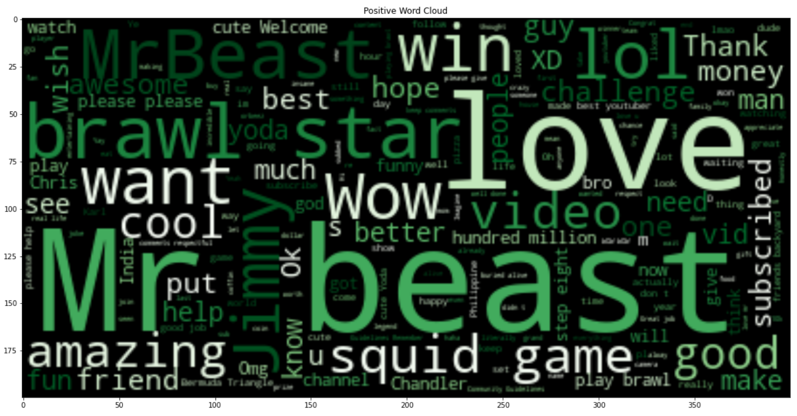


Fig 8. Word Cloud for Positive Comments

Comparison between Mr. Beast Channel vs Mr. Beast Gaming Channel

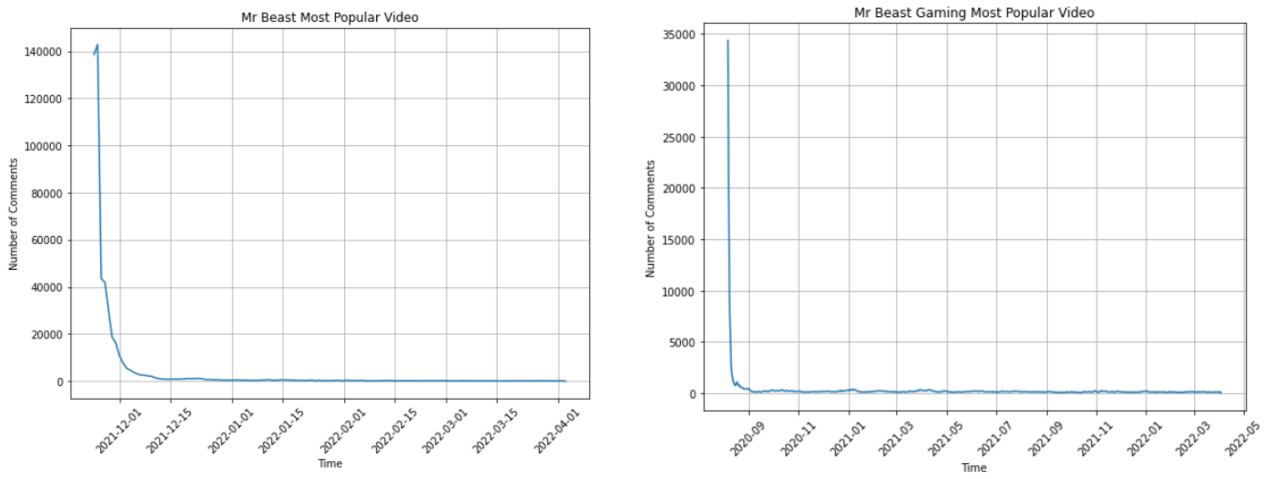


Fig 9. (Left) Mr. Beast Channel Most Popular Video View Chart (Right) Mr. Beast Gaming Channel Most Popular Video View Chart.

Mr. Beast Channel Most Popular Video '$456,000 Squid Game In Real Life!' has almost 247 million views and around 600,000 comments. This Video was posted on November 24th, 2021. On the other hand, Mr. Beast Gaming Channel Most Popular Video 'Whatever You Build, I'll Pay For!' has almost 88 million views and around 192,000 comments. This Video was posted on August 6th, 2020. You can see in the view chart (Figure 9), that the squid game video not only did better than the gaming channel video in terms of the number of views and number of comments but on average had more comments per day in the next 30 days of the video being posted. This might be because gaming videos have a lot smaller fan base than a general video, thus attracting more viewers constantly.

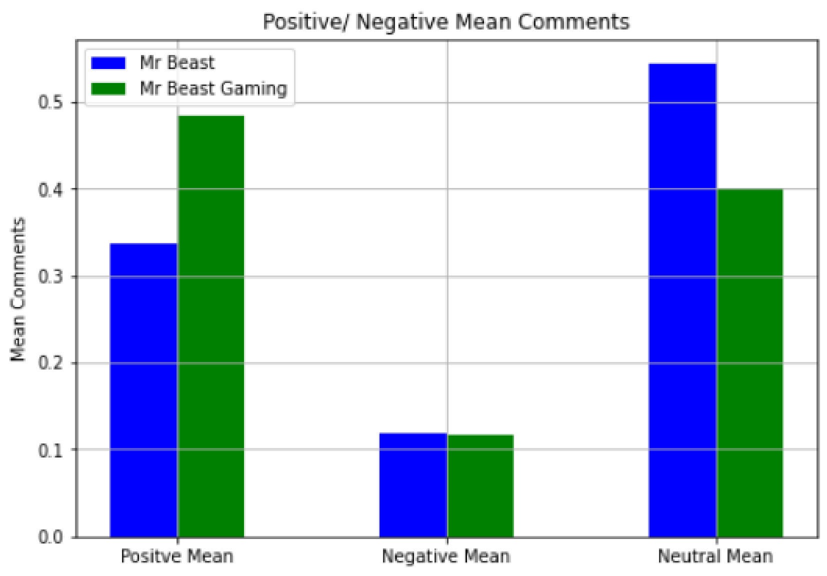


Fig 10. Sentiment Polarity Mr. Beast Channel vs Mr. Beast Gaming Channel.

All of the stats provided by YouTube tell us that Mr. Beast Channel does better than Mr. Beast Gaming channel in terms of subscribers, views, and revenue generation. However, in graph 10 we have plotted the mean polarity of Sentiments of the videos we scrapped from both of the channels. Mr. Beast Gaming channel has much higher positive comments than Mr. Beast channel.

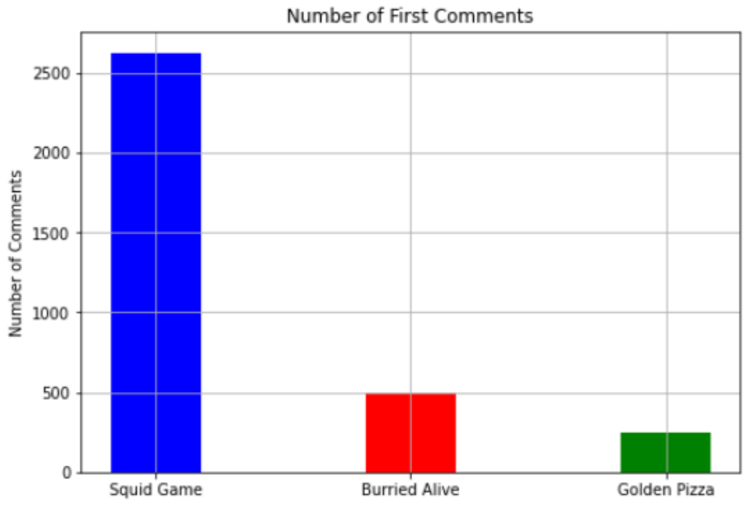


Fig 11. "First" Comment.

We discovered a phenomenon in each of the Mr. Beast videos. The first few comments are always viewers commenting on the word "First". In figure 11 you can see the number of times we saw only the first word in a comment. While classifying the data into Sentiment we expect these comments to belong to Neutral Category.

2.4. Features to use in the analysis phase.

We have used Vader for generating labels Negative, Positive, and Neutral. After we have extracted these labels, we see that we have an imbalanced dataset as seen in figure 5. To address this issue we have taken around 200,000 comments from the Positive and Neutral classes and 182,000 from the Negative class. The newly distributed dataset can be viewed here: Dataset.

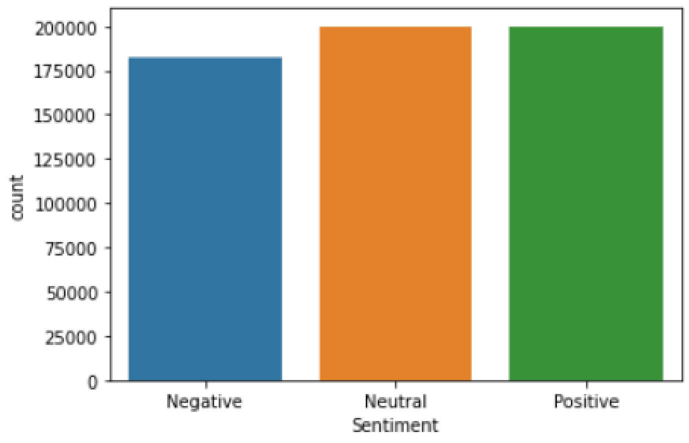


Fig 12. New class distribution.

2.5. Description of pre-processing steps.

Preprocessing is a must, to clean the data for optimal output. Most of the cleaning is common but very essential. YouTube comments are of very bad quality, and if not preprocessed properly will result in bad models. We need to keep in mind that comments might have special characters, emojis, many other languages, digits, currency symbols, etc.

To achieve the same, we have also resorted to preprocess which is as follows:

- Unicode errors were resolved
- Transliterate to the closest ASCII representation
- Convert everything in lowercase
- All line breaks were removed
- URLs were replaced with special tokens
- Emails replaced with special tokens
- Phone Numbers replaced with special tokens
- Numbers replaced with special tokens
- Digits replaced with special tokens
- Currency replaced with special tokens
- Punctuations, Emoji have been removed
- Lemmatization of comments

Cleaned dataset can be viewed here: Cleaned Dataset

3. Detailed Description of the Models used.

3.1. Description of what kinds of statistical methods and machine learning algorithms we used

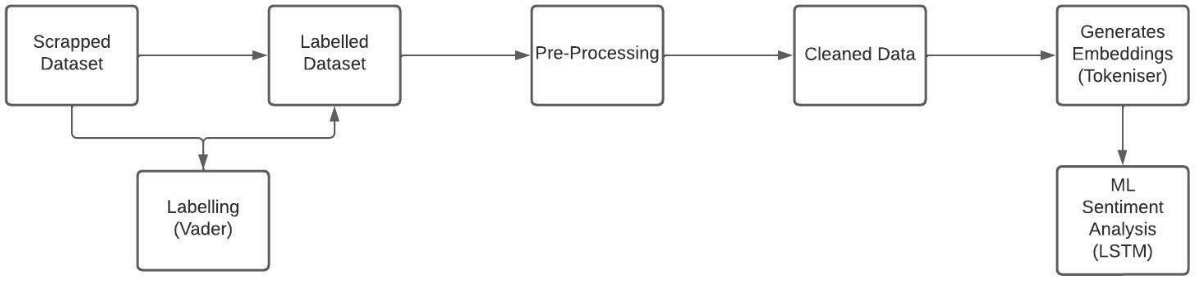


Fig 13. Model Architecture for Sentiment Analysis.

To begin with, we needed to label our dataset. For this, we tried various approaches such as K Means Clustering (Cluster size = 3, for negative, positive, and neutral) TextBlob, and Vader. Once we have a labeled dataset, we performed data pre-processing for cleaning our data. To generate embeddings we have used TF-IDF, Tokenizer, and Word2Vec Model. Finally, if we can successfully generate embeddings we have trained our model on LSTM, SVM, Random Forest, Logistic Regression, Gaussian Naive Bayes, and Bernoulli Naive Bayes.

To generate labels we tried the following approaches:

- Word2Vec for generating embeddings and K-Means Clustering for labeling dataset
- TF-IDF for generating embeddings and K-Means Clustering for labeling dataset
- TextBlob
- Vader

TF-IDF is one of the earlier methods that was used to vectorize the textual data. Although it is a simple and easy-to-use model, it cannot account for the semantic meaning in the texts. Statistical-based models such as the N-gram model, and latent Dirichlet Allocation (LDA), suffers from performance when it comes to dealing with huge datasets.

To minimize the time complexity, researchers from Google gave out **Word2Vec**. As the name suggests, it converts words/texts into their vectorized form which helps machines to learn the human language. This model draws two new architectures, namely a continuous bag of words, and a continuous skip-gram model, from previous work in feedforward NNLM and CBOW respectively. The continuous bag of word approach starts from the context to predict a word, while the continuous skip-gram starts from the single word to predict the context. Although this model can work efficiently, it couldn’t give us accurate results (Silhouette score) and thus we resorted to the advances made in the field of deep learning.

TextBlob is one of the popular libraries used in the field of Natural Language Processing that uses the natural language toolkit (NLTK). We have used TextBlob as a sentiment analyzer that gives us two key properties of a sentence for our project, i.e positive /negative polarity and subjectivity. The polarity is measured in the range of [-1,1] where the value closer to -1 indicates a negative comment and +1 indicates a positive comment. Subjectivity is measured within a range [0,1] and refers to opinion, emotions, and judgments. TextBlob works on rule-based sentiment analysis. This approach is quite simple and doesn't need any learning as compared to machine learning.

Another rule-based approach is using **VADER**. Vader stands for Valence aware dictionary for sentiment reasoning. This approach uses a list of lexical features to determine the polarity, and intensity of the emotion depicted in a sentence. The lexical approach maps the words of a sentence to a dictionary of sentiment. This dictionary of sentiments {negative, positive, and neutral} is used to generate labels on our dataset. Apart from rule-based learning, we also have a machine learning-based approach. This approach works by training a model using previously labeled data and predicting the new dataset. As we did not have labels in our dataset, this approach was deemed void. Hence, we proceeded with Vader. Vader is special in our case as it is optimized for textual data based on social media like Facebook, Twitter, etc.

In dealing with youtube comments, users tend to use a lot of emoticons like :), acronyms like DLDW which stands for dark like deep water, and slang like 'meh'. This is where Vader gains an edge as it maps these colloquialisms to their intensity values. Along with the lexical features, other parts of a sentence affect the sentiment associated with it. Contextual information like capitalization, modifiers, and punctuations also adds emotions to a sentence. Vader utilizes five ways to deal with this contextual information in a sequential manner. It starts with punctuation. For example " this video is a bomb" and "this video is the bomb!!" Vader assigns a higher sentiment score to the latter with the help of an empirically obtained value. Similar to this, it again adds extra value to a sentence that indicated higher emotions in its second step, i.e capitalization. For example "this video is the bomb" and "THIS VIDEO IS BOMB". The capitalization of words also indicates the projection of strong emotions, be it negative or positive. Thus, Vader adds empirical values to the sentiment analysis score to account for this. Following this, Vader also accounts for the shift in polarity. The use of clauses in a sentence can shift the meaning of a sentence. For example, " this video is the bomb, but it is not something I like". The first part of the sentence indicates a positive sentiment, but the second part adds more meaning to the overall sentence and indicates the overall sentiment to be negative. Vader accounts for this shift in polarity as well.

To generate a labeled dataset, we see that Vader will give us the most adequate results. We have finally used Vader for this step in our project. To generate embeddings we have used Keras Tokenizer API. For training, we have embeddings of shape (456694, 2100) and for testing, we have embeddings of shape (114174, 2100).

The next step in our project is to train a machine learning model on this labeled dataset for the classification of Sentiment Class. For this, we tried working with different ML models as follows:

- **SVM**: SVM uses something called a support vector kernel which converts the lower-dimensional data to a higher dimensional data so that we can classify the nonlinear data using a support vector classifier. This representation of data in the higher dimensional space is what helps it to classify the comments into positive, negative, and neutral labels using a hyperplane this model draws. We used the RBF kernel that helps to mathematically project the aforementioned hyperplane. Although this model didn't fail to classify, its inference time is what made it an inefficient algorithm.

- **Gaussian Naïve-Bayes:** This is a supervised model that comes from the family of probabilistic-based classification. This model draws probabilities using the Bayes probability theorem. One of the key assumptions we made while using this model is the bag of words. This means that we care more about the word and its frequency. Although this model is computationally fast, and trains context to some degree, it relies on a complete and representative dataset, and independence assumption.

- **Random Forest:** Since, we are trying to train labels like positive, negative, and neutral, ensemble algorithms like Random Forest will outperform algorithms like Decision Trees which are more prone to overfitting. XGBoost might give much better results.

- **Bernoulli Naive Bayes:** This model is based on the principles of Bayes' theorem. As the data contains user comment data, the probability distribution for a user to respond follows a binomial distribution. Hence, we decided to test Bernoulli NB Classifier. It turns out that the algorithm is faster to train but the accuracy didn't result in satisfactory results.

- **LSTM:**

  Recurrent Neural networks can represent complex patterns compared to normal feed-forward neural networks. The key point in this type of network is that it connects its hidden layer back to itself. Thus, using a time-delayed connection that helps in creating a short-term memory. In short, RNN is a generalization of feedforward neural networks with memory. Here the term recurrent indicates how the network works in the background. It takes input one at a time and the output generated is later utilized for the next input. RNNs use this as their internal state to process the subsequent inputs. This makes them good for sequential data such as ours. The major problem that these neural networks suffer from is vanishing and exploding gradients. The gradient value calculated during the backpropagation step either becomes too small or too big and doesn't contribute to the learning. Thus it fails to converge the function and leads to inaccurate results.

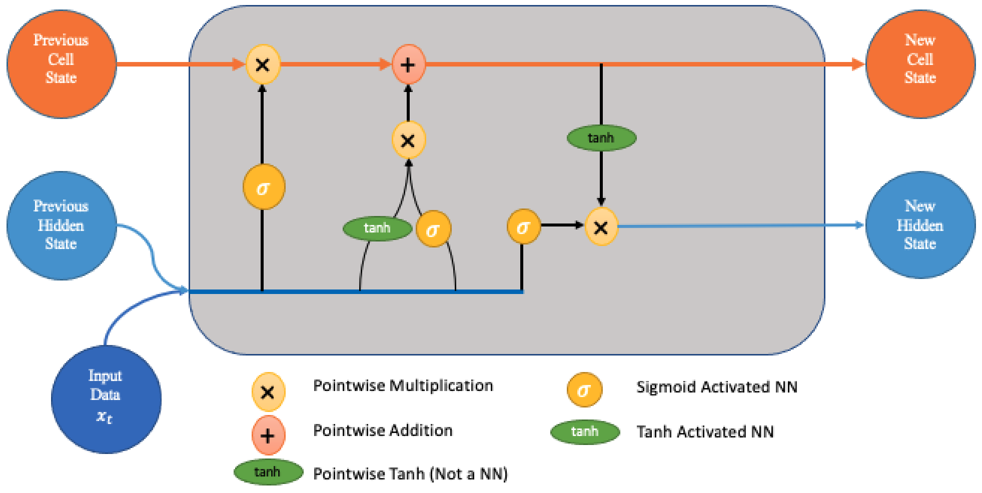


Fig 14. LSTM Architecture for Sentiment Analysis.

A modified version of this network is Long Short Term Memory (LSTM). This architecture modifies the backend of the algorithm by adding three gates in its structure, namely the input gate, forget gate, and output gate. These gates improve the efficiency of RNNs. The memory cell helps to keep track of the previous state, thus helping in removing the vanishing gradient problem. The forget gate helps to keep track of what input states should be considered with the help of a sigmoid function. The value that tends to zero is disregarded. The input gates help us to determine the importance of the information in the input. This is done by passing the feature vector through sigmoid and tanh functions. Following this, a cell state is obtained using the output from the other two gates. First, it multiplies with the output from the forget gate and then makes a pointwise addition to the output from the input gate. Finally, the output gate passes the new cell state through a tanh function and is multiplied with a sigmoid function to decide the information to be carried by the hidden state. LSTM is what we have used to classify sentiments.

3.2. Why LSTM performs better than Vader for Sentiment Analysis

VADER is a lexicon and rule-based tool for sentiment analysis. This pre-trained model is easy to set up and the results are easy to interpret. However, faces some limitations like the model ignoring the context of the text, and certain suitability issues across domains. VADER lexicon does not necessarily understand irony, jargon, grammatical errors, and sarcasm. LSTM on the other hand despite being difficult to train and set up is industry-proven to perform well in NLP tasks. LSTM does take into account the context of the text due to its long and short-term memory.

3.3. Model Validation

We have considered two ways to validate our models. The first of them is splitting the data into an 80/20 split of training and testing datasets. This is because then we can make use of most of the data for training purposes and then use just a small portion for testing. This in turn would give us an idea of how well our machine learning model performs.

Secondly, we have calculated the validation accuracy of our LSTM model which tells us how efficiently our model has worked using the test data. In our case, we ran a total of three epochs which gave us a validation accuracy of 0.9814, 0.9822, 0.9833 respectively, and training accuracy of 0.9577, 0.9861, and 0.9901. This is a good sign that our model is not overfitting or underfitting the dataset and is generalizing well to unseen data.

3.4. Other models, we considered and why didn't we proceed with those?

For this project, we considered a lot of models but not all of them could be used because of some of the other constraints. Following are the models and the reasons we could not use them for our project.

Model	Accuracy on Test Set
SVM	Model did not train, SVM training time complexity is $O(n_{features} \times n^2_{observations})$
Random Forest	0.611
Logistic Regression	0.416
Gaussian Naïve Bayes	0.342
Bernoulli Naïve Bayes	0.407

Table 1. Model Accuracy Table

This is a classification task and we used the above-mentioned classification Algorithms. Due to the size of our training data SVM was not able to finish training, our model trained for around 6 hours and finally gave used out of memory error (configuration used - Colab Pro High RAM 27.3 GB). Logistic regression performed better than what was expected since Logistic Regression rarely performs well in NLP tasks. Naive Bayes algorithms are better when it comes to NLP tasks however we saw that Gaussian Naive Bayes and Bernoulli Naive Bayes performed worse than Logistic Regression. We did not use Decision trees because of their nature to overfit the data. We also tried Random Forest Classifier, which gave us the best accuracy out of the other discussed algorithms.

As you can see, all of these algorithms fail to generalize well to unseen data. Also since the sklearn implementation of these algorithms does not have GPU support. LSTM on the other hand gave us high accuracy on train and test dataset and have TensorFlow GPU support for fast training and predictions.

4. Business Applications

The number of comments on a popular YouTube video is huge. It is difficult to make sense of it by the viewer or even the Youtuber. We plan to develop a fully functional website, in which the user needs to enter the YouTube video code. After this, we will scrap the data and provide the user with a brief analysis of comments on the video. This will give the users another metric to rate the video. Also, this will help the Youtuber to know the public consensus around their video.

This application was developed using Django Python Framework. The code for the web application can be downloaded here, GitHub. See figure 4 to understand the scrapping phase of the project. The data is stored in an S3 bucket and our application uses python boto3 to get data from the bucket. For production purposes, we have stored our trained LSTM model and other necessary files in a public S3 bucket. Necessary files can be downloaded here, tokenizer, labelEncoder, LSTM model, and LSTM weights.

The application demo can be seen here: Demo. This demo is on a local server, as due to the large size of the files required, Heroku is not able to support the application. However, this application can be easily replicated. Steps to replicate are given in the readme.MD file of GitHub repository. Moreover, since we are using paid technologies like AWS and Youtube API, we did not make this application publicly available.

5. Important Links:

5.1. Github: https://github.com/EashanKaushik/youtube-comment-analysis

5.2. Colab Notebook:

- TextBlob:https://drive.google.com/file/d/1b6oYOWlSl_iIg3sHZos7ltpg61SrYEwb/view?usp=sharing

- Vedar:https://drive.google.com/file/d/19m1stfsVklcP1IUsl8ApLD0tTR6sPFpg/view?usp=sharing

- Word2Vec:https://drive.google.com/file/d/16C6lSoFOdmTFzI7BtmaMk-3iMpE_dzgH/view?usp=sharing

- TF-IDF:https://drive.google.com/file/d/1b3LGU_NukRIusi_P3LbbOxRPfl29iqkG/view?usp=sharing

- LSTM:https://drive.google.com/file/d/1FHXmgcLsjpRmk9SpaZF7ELV5EEzXVXpP/view?usp=sharing

- Models (random forest, logistic regression, Naive Bayes):

https://drive.google.com/file/d/1vnInA6SAsT8YhVBi30ui3a_BwCgRsT6X/view?usp=sharing

- Exploratory Data Analysis (EDA): https://drive.google.com/file/d/1XxYpw-kVlqczTpSDOAN3zaVFL7tgCZeF/view?usp=sharing

5.3. Dataset: https://www.kaggle.com/datasets/eashankaushik/youtube-scrapped-data

5.4. Embeddings: https://drive.google.com/drive/folders/183pFYwahAP6oP7qKidCXuC-kauF1W9En?usp=sharing

5.5. Demo Video: https://www.youtube.com/watch?v=BNPqXxj8aqg

References

[1] https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

[2] https://pypi.org/project/clean-text/

[3]https://towardsdatascience.com/machine-learning-word-embedding-sentiment-classification-using-keras-b83c28087456

[4]https://towardsdatascience.com/sentiment-analysis-comparing-3-common-approaches-naive-bayes-lstm-and-vader-ab561f834f89