Sentiment Analysis of Youtube Live Chats using Machine Learning Methods

Aniket Malik
IIIT Delhi
aniket21231@iiitd.ac.in

Piyush Gautum
IIIT Delhi
piyush21549@iiitd.ac.in

Mayank Jha
IIIT Delhi
mayank20521@iiitd.ac.in

Abstract

In today's digital age, live streaming and real-time chat features have become an integral part of online entertainment. However, the ease of anonymous communication in live chats has given rise to various forms of toxicity, including hate speech and harassment, creating a hostile environment that adversely affects user experiences. While platforms often rate video content by age, live chat content remains largely unregulated.

This research is motivated by the pressing need to address this issue. Toxicity in live chats not only harms individuals but also tarnishes the reputation and community health of streaming platforms. Some platforms have taken steps to employ content moderation, but there is room for improvement. More refined machine learning models, trained on extensive chat data, can help detect nuanced forms of toxicity that elude traditional filters.

By enhancing toxicity detection algorithms and implementing real-time monitoring, streaming platforms can cultivate a safer and more inclusive online community for users of all ages. This proactive approach can mitigate the adverse effects of toxic behaviour, fostering a more positive online environment.

1. Introduction

YouTube Live chats have transformed the way viewers engage with content creators in real time. These live chats provide an interactive space where viewers can communicate with each other and the streamer, sharing thoughts, questions, and reactions as events unfold. This dynamic feature has significantly contributed to the popularity of live streaming, making it a central aspect of online entertainment, educational webinars, and more.

However, the open and dynamic nature of YouTube Live chats also brings forth a critical concern - the prevalence of toxic behavior and harmful interactions within these spaces. Toxicity in live chats can manifest as hate speech, offensive comments, harassment, and other forms of disruptive communication. This not only creates a hostile environment but also detracts from the overall experience for users, potentially leading to serious emotional and psychological consequences.

Recognizing the gravity of this issue, our research focuses on Sentiment Analysis of YouTube Live Chats using Machine Learning Methods, with the primary objective of classifying chat messages as either safe or toxic. By doing so, we aim to foster a healthy and safe environment for live chat interactions among viewers and between viewers and streamers alike.

This problem holds immense significance, extending its use cases beyond the realm of entertainment. It can be applied to the moderation of online classroom chats, ensuring that students have a productive and respectful learning environment. Moreover, it can be utilized in various other scenarios where real-time chat interactions occur, ranging from live Q&A

sessions to community forums, enhancing the quality of online interactions and fostering a more positive and inclusive digital space.
In this paper, we will delve into the methodologies and machine learning techniques used to develop a model capable of accurately detecting and classifying toxic content in YouTube Live chats. We will explore the challenges and opportunities in this domain, with the ultimate goal of contributing to a safer and more welcoming online community**.**

2. Literature Survey

We look at two research papers that try to perform sentiment analysis or text mining. Following papers helped us better understand the use and difficulties that could be incurred while performing the analysis.

2.1 Surjuse, V., Dharne, A., &Lade, H. (2019). SENTIMENT ANALYSIS OF CHAT BASED APPLICATION USING R. Journal of Engineering and Technology Innovation Research.

https://www.jetir.org/papers/JETIR1903556.pdf

This paper aims to learn about the sentiments of an individual involved in the text conversation.
This task has several complexities:-
-Informal language and short forms
-Colloquialisms and slags.
The research is vital for several reasons:-
-Understand the effect of social media
-Study rising anxiety and depression due to social media usage.
-Study and curb the trends of cybercrime and online bullying
Therefore, this research has implications in fields like management and social sciences, not just computer science.
One approach to sentiment analysis is,
Lexical approach = It uses literal keywords like 'happy,' 'sad,' 'afraid,' etc and their dictionary synonyms. It is simple and naive but does not understand negation. It also does not understand the subtext and relies on surface keywords. Therefore, it is prone to error.
Machine Learning-based approach uses three stages: Data collection, Pre-processing, Training data, Classification and plotting results.

2.2 Chouhan,A.,Halgekar,A.,Rao,A.,Khankhoje, D. & Narvekar,M.(2021).Sentiment Analysis of Twitch.tv Livestream Messages using Machine Learning Methods.Dwarkadas J. Sanghvi College of Engineering.

https://ieeexplore.ieee.org/document/9616932

This paper offers a methodology for sentiment analysis with ML models on live stream messages.
The task presented several challenges:
A large number of messages were involved.
Streamer-specific slang and emoticons made the language context-dependent.
The language contained numerous abbreviations, repetitions, and deliberate grammatical mistakes.
The use of emojis added complexity to the problem.
The following workflow was employed:
-They had a labeled dataset.
-Pre-processing was performed, and the dataset was split into three sets for training, validation, and testing (60:20:20).
-The pre-processing pipeline comprised three steps: tokenization, removal of stop words, and lemmatization.
-Stop words, which are common words with little to no meaning in text analysis, were removed.
-Data was then split.
-Subsequently, the text was converted into vector form using Count Vectorization and TF-IDF Vectorization.
Different ML models were used and their accuracies were compared.
Support Vector Classifier 70.4%
Logistic Regression 69.2%
Decision Tree Classifier 67.2%
Random Forest Classifier 66.4%
Multinomial Naïve Bayes 65.8%
SVM produced the best accuracy of SVM 70.4% suggesting that it might be the best model for us.

3. Dataset

The following dataset was used-
uetchy. (2021). Sensai Dataset. Kaggle. URL: https://www.kaggle.com/datasets/uetchy/sensai/data
As the dataset was divided into a number of parquet files,we combined them into a single CSV file.
The dataset employed in this research study comprised a substantial dataset, consisting of a total of 7,700,072 rows. These rows were structured around two primary columns, namely 'body' and 'label.'Label were of three types-

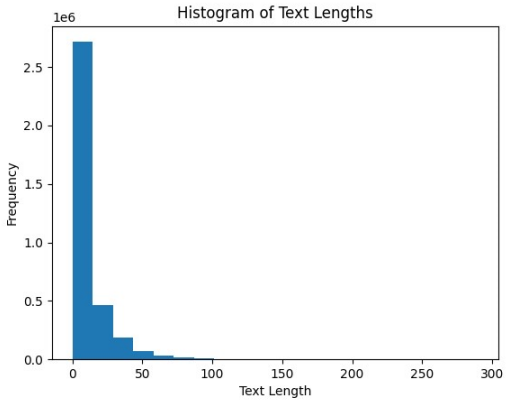

Fig.1.Histogram for Frequency of Length of Chats

-Hidden/Deleted: For flagged(toxic) chats
-Nonflagged: For non-flagged chats

Chats contained both ASCII and non-ASCII characters along with special characters. The following techniques were used in Data Preprocessing-
Removing Duplicates and NAN/empty strings-
Duplicates were removed from the dataset and the NAN/empty strings were deleted thus reducing the size from 7,700,072 to 3,508,663 rows.
Then we used the following techniques-
a)Lowercasing

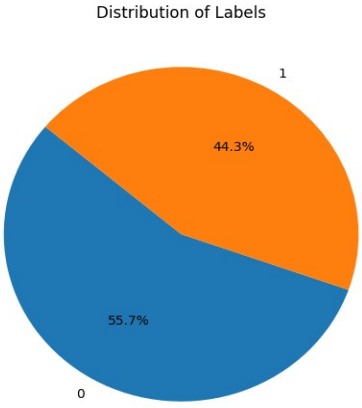

Fig.2. Pie Chart depicting Distribution of Labels

b)Removing Non-ascii and special characters
c)Tokenization

d)Normalizing Contractions
We also label encoded the labels-
-Hidden/Deleted: 1
-Nonflagged: 0
The dataset is well balanced as it contains 44.3% label 1 and 55.7% label 2 indicating the dataset is not biased.
We also analysed the frequency of the length of chats to help understand the feasibility of pre-processing and training on a large dataset.

4. Methodology and Model details

Initially, to perform preprocessing and data analysis, we used several libraries in Python including - numpy, pandas,sklearn, sea born, matplot.lib, contractions.
Preprocessing is vital in text sentiment analysis to cleanse and structure text data, removing noise and irrelevant information, ensuring accurate sentiment classification and meaningful insights.
Data preprocessing-
a) Removing Duplicates-Removed duplicates rows for better accuracy and preventing biasing.
b)Lowercasing: Converted all characters into lowercase.For eg- otherwise man and Man would be treated differently by the model.
c)Tokenization: Splitted the large sentences into smaller chunks which will help in preprocessing of the data more efficiently and reasonably using word_tokenize from NLTK library
d)Normalizing Contractions-Contractions were expanded during data processing; for instance, "don't" was transformed into "do not" to facilitate comprehensive text analysis
e) Stemming - Converting the word to it's root form.
f)Number of Samples split: Samples from the column of the label has been divided in half -
-Hidden/Deleted: 1
-Nonflagged: 0
Feature Extraction-Feature extraction is the process of transforming raw data, often high-dimensional or unstructured, into a reduced set of meaningful and informative features that can be used for analysis, modelling, or machine learning tasks.We implemented this using -
TFIDF Vectorizer-The TFIDF Vectorizer was employed to convert textual sentences into numerical vectors, serving as a standardization and model representation technique for building various models, such as logistic regression and SVM.
Fasttext-FastText provides a simple way to obtain vector representations (embeddings) for words using its pre-trained models or models trained on custom data. The vectors capture semantic information about the words and can be used as features for various natural language processing (NLP) tasks.:
Word2Vec - Word2Vec is a word embedding technique in natural language processing (NLP) that represents words as dense vectors in a continuous vector space. It learns these representations by analyzing the contextual relationships between words in a given corpus. Word2Vec aims to capture semantic similarities, allowing words with similar meanings to have similar vector representations. The resulting word vectors are useful in various NLP applications for tasks such as text classification, sentiment analysis, and similarity analysis.

Postprocessing, the dataset was reduced in size and better suited to ML models.
The following ML models were used-
We used an 80:20 (train: test) data split for our models.

Logistic Regression - Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance belongs to a given class. It is used for classification algorithms its name is logistic regression. it's referred to as regression because it takes the output of the linear regression function as input and uses a sigmoid function to estimate the probability for the given class.
Max_Iterations:10,000 Random State:42
Performance with -
TF-IDF- Accuracy-0.651
Word2Vec- Accuracy-0.644

Naive Bayes-Multinomial Naive Bayes is a supervised ML algorithm for classification, widely applied in text-based tasks like spam detection. Leveraging conditional independence assumptions, it calculates class probabilities using Bayes' theorem to make predictions, offering a probabilistic approach to classification tasks. It is used as a benchmark for

classification problems. Along with vectorizer we use standardization for better performance.
Performance with different vectorizers:-
TF-IDF: Accuracy = 0.5836

Decision Trees-Decision Trees is a supervised ML algorithm, that excels in classification and regression tasks. They recursively split data based on feature conditions, creating a tree structure for decision-making. We have used a decision tree with minimum sample split = 6000, maximum features = 12, entropy criterion and maximum depth = 8. Along with vectorizer we use standardization for better performance.
Performance with different vectorizers:-
TF-IDF: Accuracy = 0.5054
Word2Vec: Accuracy = 0.65
FastText: Accuracy = 0.6501

Random Forest - Random Forests, an ensemble learning method, leverages multiple Decision Trees for robust predictions in classification and regression tasks. By combining diverse trees, they mitigate overfitting and enhance accuracy, making them effective in diverse domains. We have combined 10 decision trees, using gini entropy as a criterion, maximum depth = 8, minimum samples split = 5300, maximum features = 32 and random state = 42. Along with vectorizer we use standardization for better performance.
Performance with different vectorizers:-
TF-IDF: Accuracy = 0.5354
Word2Vec: Accuracy = 0.6506
FastText: Accuracy = 0.6507

SVM - Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for classification or regression tasks. It is a type of discriminative classifier that works by finding the hyperplane that best separates different classes in the feature space. The "support vectors" are the data points that lie closest to the decision boundary, and the margin is the distance between the decision boundary and the support vectors.SVM was applied on the subsampled dataset consisting of 2,00,000 samples instead of the original dataset due to the high computational cost and time.
Performance with different vectorizers:-
TF-IDF: Accuracy = 0.613
Word2Vec: Accuracy = 0.598

XG Boost - XGBoost is an optimized distributed gradient boosting algorithm which is designed to be highly efficient, flexible and able to handle higher dimensional data like sparse datasets and it has lower computational expenses. Since XGBoost uses Decision trees to identify the most important variables and build models. By default, XGBoost uses a greedy algorithm for split finding which makes the split finding process quick enough and because of this XGBoost is generally faster than Regression algorithms like linear and logistic regression. Here we have used XGBoost as XGBClassifier with hyperparameters as objective binary logistic and random state=42 by using Fasttext and Word2Vec vectorizers for conversion of words

Recurrent Neural Network (RNN) - Recurrent Neural Networks (RNNs) are specialized neural networks for sequential data, excelling in NLP. Featuring recurrent connections, they capture dependencies in sequential patterns. We have applied RNN on ⅓ of the original samples for faster convergence. We have used a different preprocessing step here using Tokenizer, we tokenize the text input. Then perform training with an embedding layer that gives an output of 32 features. Then we added 3 layers of Simple RNN with a dropout of 0.1. Then we added a dense layer with with sigmoid activation function. We use binary cross entropy, adam optimizer. We run the model for 7 epochs.

5. Result and Analysis

We obtained the following results after training and testing various machine learning models-
Logistic regression-

Accuracy	0.6511450936467289
Precision	0.7032274331820474
Recall	0.3678058508499654
F1 Score	0.48299370011890147

Table 1.Logistic Regression Results

Label	Precision	Recall	F1 Score	Support
0	0.64	0.88	0.74	390838
1	0.70	0.37	0.48	310895

Table 2.Logistic Regression Classification Report

Naive Bayes:-

Accuracy	0.5836
Precision	0.64
Recall	0.58
F1 Score	0.58

Table 3.Naive Bayes Result

Label	Precision	Recall	F1 Score	Support
0	0.55	0.90	0.68	20007
1	0.72	0.27	0.39	19993

Table 4.Naive Bayes Classification Report

Decision Trees:-

Accuracy	0.6501
Precision	0.6772
Recall	0.5734
F1 Score	0.6210

Table 5.Decision Trees Results

Label	Precision	Recall	F1 Score	Support
0	0.63	0.73	0.68	20007
1	0.68	0.57	0.62	19993

Table 6.Decision Tree Classification Report

Random Forest:-

Accuracy	0.6507
Precision	0.6764
Recall	0.5772
F1 Score	0.6229

Table 7.Random Forest Results

Label	Precision	Recall	F1 Score	Support
0	0.63	0.73	0.68	20007
1	0.68	0.57	0.62	19993

Table 8.Random Forest Classification Report

SVM-

Accuracy	0.613
Precision	0.723
Recall	0.448
F1 Score	0.553

Table 7.Random Forest Results

Label	Precision	Recall	F1 Score	Support
0	0.56	0.80	0.66	12574
1	0.72	0.45	0.55	14461

Table 8.Random Forest Classification Report

RNN:-
Accuracy = 0.8745
Total Parameters = 663969
Validation Accuracy = Accuracy on a Validation Data set.

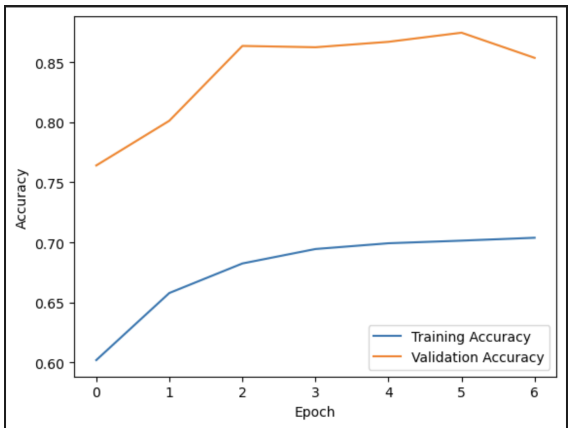

Fig.3. Accuracy vs Epochs (RNN)

XGBoost:-

Accuracy	0.65385
Precision	0.6470792285278202
Recall	0.515702753511003
F1 Score	0.5739692307692308

Table 9.XG Boost Results

Label	Precision	Recall	F1 Score	Support
0	0.66	0.77	0.71	21914
1	0.65	0.52	0.57	18086

Table 10.XG Boost Classification Report

MLP:-

Accuracy	0.658175
Precision	0.629557865069579
Recall	0.5928342364259649
F1 Score	0.6106444172338183

Table 11.MLP Results

Label	Precision	Recall	F1 Score	Support
0	0.68	0.71	0.70	21914
1	0.63	0.59	0.61	18086

Table 12.MLP Classification Report

5. Conclusion

Based on the results,we can conclude that the RNN model produced the best performance in detecting toxic chats. Word2Vec, being a contextually better vectorizer than TF-IDF, performed better in most cases except Logistic Regression and SVM. Most models averaged around 0.65 in accuracy indicating that the dataset might not be optimally created for machine learning models. The lower accuracy of the Multilayer Perceptron (MLP) compared to the RNN (65% vs. 87.5%) suggests that for this specific task, the sequential nature of the conversation, captured effectively by the RNN, plays a critical role in achieving high accuracy, which the other models failed to achieve.