Deerwalk Journal of Computer Science and Information Technology

Automated Detection of Hate Speech in Twitter Using Natural Language Processing

Aayam Ojha*

ojha.aayam@gmail.com

ABSTRACT

This research project aimed to develop a highly efficient and effective Chrome extension that could classify tweets containing hate speech, along with conducting sentiment and topic analysis. Hate speech is a persistent and concerning issue on Twitter, yet the platform has made little effort to address it. To address this challenge, this research performed a series of experiments, including the use of Support Vector Machine (SVM), Random Forest, and Long Short-Term Memory-based (LSTM) neural network classifiers. The results of the experiments showed that the SVM classifier, combined with word2vec [1] feature engineering, outperformed all other methods.

Then developed the Chrome extension using a monolithic repository architecture, utilizing React and Django. By implementing this extension, users are able to automatically analyze live tweets and identify hate speech content, along with obtaining sentiment and topic analysis. The outcome of this research project could provide a significant contribution towards a more positive online environment and towards curbing the prevalence of hate speech on Twitter.

Keywords: *NLP, Hate Speech Classification, SVM, NLP Classifier, LSTM, Chrome Extension*

1 INTRODUCTION

1.1 Overview

Hateful speech is a pervasive problem in modern times due to the rise of the internet, making it easier to spread negative content to a large audience. Twitter, a popular social media platform, claims to promote free speech, but it has been widely used to spread hate and promote hateful ideologies. As such, there is a need to develop a tool to censor such content on Twitter. This paper proposes the development of a Google Chrome plugin extension that utilizes natural language processing to detect hate speech in a user's live Twitter feed.

1.2 Background and Motivation

Despite the negative effects of hate speech on people's emotional well-being, Twitter has refused to take any action against its platform being used for racist, anti-Semitic, homophobic, and transphobic attacks. Twitter’s negligence towards hate speech on their platform is well demonstrated by insurrection at U.S. capital building on 6th January, 2020; it took Twitter 4 years to final take action against an alleged racist and Nazi sympathizer Donald J. Trump. The growth of hate speech on other platforms such as Reddit has shown negative effects on the mental health of college students [2]. To address this problem, this paper proposes a solution that will automatically detect and censor hate speech on Twitter using a Google Chrome plugin. To ensure that the system is effective, it will also incorporate topic and sentiment analysis to provide a more comprehensive hate speech detection system. The removal of hate group Subreddit on Reddit has shown that censoring certain content is necessary to reduce hate speech on online platforms [3], highlighting the importance of developing effective tools to detect and censor hate speech.

Paper purposes a hate speech classification model developed using Natural Language Processing (NLP) techniques such as embedding matrix using word2vec [1], and mean pooling for feature reduction. Paper drew conclusion on using noble SVM model after running 21 different experiments with different NLP feature engineering technique and neural network architecture.

For the sentiment and topic analysis, this paper purposes the use two transformers from the hugging face library: cardiffnlp/twitter-xlm-roberta-base-sentiment [4], and classla/xlm-roberta-base-multilingual-text-genre-classifier [5]. This allowed for a robust hate speech detection system to be built around hate speech classification, sentiment analysis, and topic analysis.

1.3 Problem Statement

In recent years, hate speech has become an inescapable part of daily life for Twitter users, and despite the negative psychological effects linked to hate speech [2], Twitter has not taken sufficient action to address this issue. Any measures taken by Twitter have only been in response to public outcry and not of their own accord. In 2022, Twitter finally banned Kanye West after decade long controversies and hate speech, it was same with Donald J. Trump. Twitter's hate speech policy aims to create a safe and healthy environment on their platform [6], but it seems like twitter has done very little to support this policy, especially when it comes to celebrities and politician. This issue has future magnified after Elon Musk took over Twitter on October of 2022; as state in a piece done by Sheera Frenkel and Kate Conger in the New York Times, the derogatory inuendo against black Americans have gone up from 1,282 times a day to 3,876 times a day, and slurs against gay men used to appear 2,506 times a day now going up to 3,964 times a day, as well as antisemitism has increased by more than 61 percent in the two weeks after Elon Musk acquire the site. [7]

1.4 Objectives

The Objectives of this project are listed below:

a) To censor hateful tweet written in English language
b) To perform Topic and Sentiment analysis on Nepali and English tweets
c) To develop a natural language processing model to identify hate speech tweets
d) To develop Google Chrome plugin that will scrape live tweet feed of user to filter hateful tweets.

1.5 Scope and Limitation

This project is categorized under web-based machine learning applications, which is a specialized field in computer science that uses data to train specific parameters in an equation to make predictions. The project uses a trained machine learning model and two pre-trained transformers to determine if a tweet is hateful, as well as to perform topic and sentiment analysis on the tweets. Hate tweets are concealed, and the topic and sentiment of the tweet are presented above it in the form of a badge [see Appendix I].

The scope of this project can be extended to multiple social networking websites including but not limiting to Facebook, Reddit, etc. As all of these websites require a way to control hate speech, but they cannot censor individuals' negative and hateful messages in order to safeguard freedom of speech. This extension circumvents this issue by allowing users to choose what they wish to view or avoid.

However, a significant limitation of this project is that the developed extension is not cross-browser compatible, nor is the trained model and deployed backend are cross-platform compatible with other social networks such as Facebook. This necessitates the development of entirely new extensions for web browsers like Firefox and new machine learning models and backends for other social media platforms.

2 BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

Machine learning is an established technology with a wide range of applications in various fields of computer science, including computer vision, signal analysis, and natural language understanding. This paper focuses on a specialized branch of machine learning known as NLP.

Prior to the emergence of NLP, understanding and analyzing linguistic idiosyncrasies was nearly impossible. In traditional systems, there was no way to automatically comprehend natural language, so people had to be hired to review the sentiment and topic of any corpus. However, NLP bridged this gap by providing a means to vectorize text.

Social media platforms are excellent for sharing ideas and ideologies, but they are frequently contaminated with hateful speech that has a detrimental impact on viewers' mental health [2].

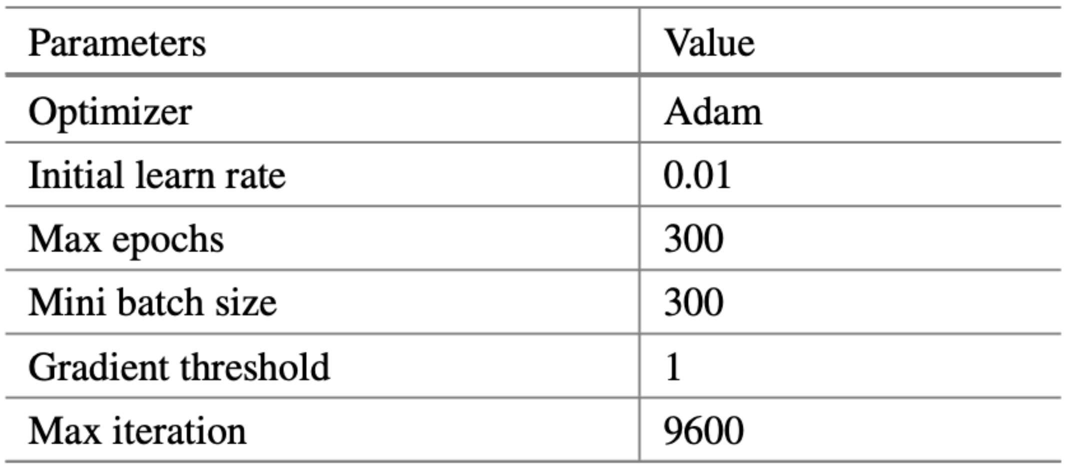
A comprehensive literature review of the implications of hate speech on social networking platforms, as well as various machine learning and deep learning approaches for developing a classification model, is required to be deployed and used by a Chrome-based extension to address the problem of hate speech.

2.2 Literature Review

The prevalence of hateful speech in online communities has garnered attention in recent years, as its psychological effects on individuals have become a cause for concern. Koustuv Saha et al. [2] conducted a survey to investigate the prevalence of hateful speech in online college communities and its psychological effects on students. The study found that hate speech was present in 87.3% of the observed online college communities, and that 49.7% of participants reported experiencing psychological distress as a result of hate speech. These findings suggest that hate speech has a direct negative effect on mental health and requires immediate attention and action. Clay Calvert [3] examined the potential harms of hate speech and how communication theory can help understand and address them. In the ritual communication model, hate messages require a channel to be transferred from sender to receiver, with Twitter serving as such a channel in recent years.

Machine learning algorithms have shown promise in detecting hate speech, but the choice of model remains an important factor. Sindhu Abra et al. [8] aimed to provide insights into the strengths and weaknesses of various machine learning algorithms for hate speech detection. The study employed data cleaning techniques, such as vectorization, lowercasing, removal of special characters, and stemming, followed by text vectorization using Term Frequency-Inverse Document Frequency (Tf-idf) with bi-gram. The study found that Multi-layer Perceptron (MLP) outperformed other models, achieving 96.7% accuracy, while SVM had an accuracy of 95.3%. These results indicate that simple preprocessing and algorithmic models can accurately classify hate speech.

Akanksha Bisht et al. [9] proposed a deep learning approach based on LSTM models for detecting hate speech and offensive language in Twitter data. The study employed various NLP techniques, such as tokenization, stop-word removal, and stemming, followed by text vectorization using Sandford's GloVe2vec [10]. The system architecture consisted of one LSTM/Bi-directional Long Short-Term Memory (Bi-LSTM) layer with 100 units, followed by a fully connected SoftMax layer and a classification layer with follow stopping criteria:



Parameters	Value
Optimizer	Adam
Initial learn rate	0.01
Max epochs	300
Mini batch size	300
Gradient threshold	1
Max iteration	9600

Fig. 1: Stopping Criteria for Neural Network

The study found that a single LSTM-based neural network had an accuracy of 86%, and a single bi-LSTM layer achieved 84.62% accuracy, demonstrating the effectiveness of this approach for detecting hate speech in Twitter data. Gretel Liz De la et al. [11] employed a similar approach, training LSTM models using k-fold stratified cross-validation. The study found an F1 score of 87.7% on a Twitter dataset and 89.9% on a Facebook dataset when using five folds, indicating the efficacy of this approach in detecting hate speech.

Sujatha Arun et al. [12] proposed a stacked weighted ensemble (SWE) model to improve the accuracy and efficiency of hate speech detection on Twitter. The study employed data cleaning techniques using the word_token tool from the Natural Language Toolkit (nltk) library, followed by the removal of hashtags and URLs, conversion to lowercase, removal of punctuation, and Porter stemming. The text was vectorized using Google's word2vec [1] technique. The SWE model had the highest F1 score of 78%, along with precision of 89% and accuracy of 95.54%, outperforming state-of-the-art machine learning algorithms such as SVM. The study concluded that the SWE model can achieve high performance with appropriate feature engineering.

In summary, the studies discussed above provide valuable insights into the prevalence of hate speech in online communities and the psychological effects it has on individuals, as well as effective approaches to detecting and classifying hate speech using machine learning algorithms. These studies demonstrate the need for continued research in this area to develop effective strategies for combating hate speech in online spaces.

2.3 Current System

Larry filter for Twitter is a Chrome extension that allows users to filter out tweets based on specific keywords, hashtags, and usernames. Here is a breakdown of its features:

Features:

- You can filter out tweets based on specific words, hashtags, and usernames.
- The tweets are hidden, but not deleted, so they can be retrieved at any time by disabling the filter or changing the criteria.
- It is compatible with the latest version of Twitter.
- The extension is easy to install and use, with a user-friendly interface.

In summary, both Open Tweet Filter and Larry filter for Twitter offer filtering capabilities for Twitter, but Open Tweet Filter allows filtering based on content or author using regular expressions, while Larry filter for Twitter filters tweets based on specific keywords, hashtags, and usernames.

2.4 Problem with Current System

The current system allows users to filter out unwanted tweets on Twitter using regular expressions or a list of specified words, hashtags, and usernames. However, this system does not address the specific problem of hate speech classification in tweet feeds, or provide functionality for topic and sentiment analysis in English and Nepali.

Some of the specific problems with the current system include:

Lack of hate speech classification: The current system does not have the capability to classify tweets as containing hate speech or not. This is a critical issue for many social media platforms, including Twitter, which has faced significant criticism for not doing enough to combat hate speech and harassment.

Limited language support: The current system only provides support for filtering tweets in English. This is a problem for users who may be interested in analyzing tweets in other languages, such as Nepali.

Limited analysis capabilities: The current system only provides basic filtering functionality and does not provide any analytical capabilities for topic and sentiment analysis. This is a significant limitation for users who may be interested in understanding the topics and sentiments of the tweets in their feed.

To address these problems, a more comprehensive system is needed that includes capabilities for hate speech classification, support for multiple languages, and advanced analytical capabilities for topic and sentiment analysis.

3 SYSTEM ANALYSIS

3.1 Analysis

3.1.1 Sequence Diagram

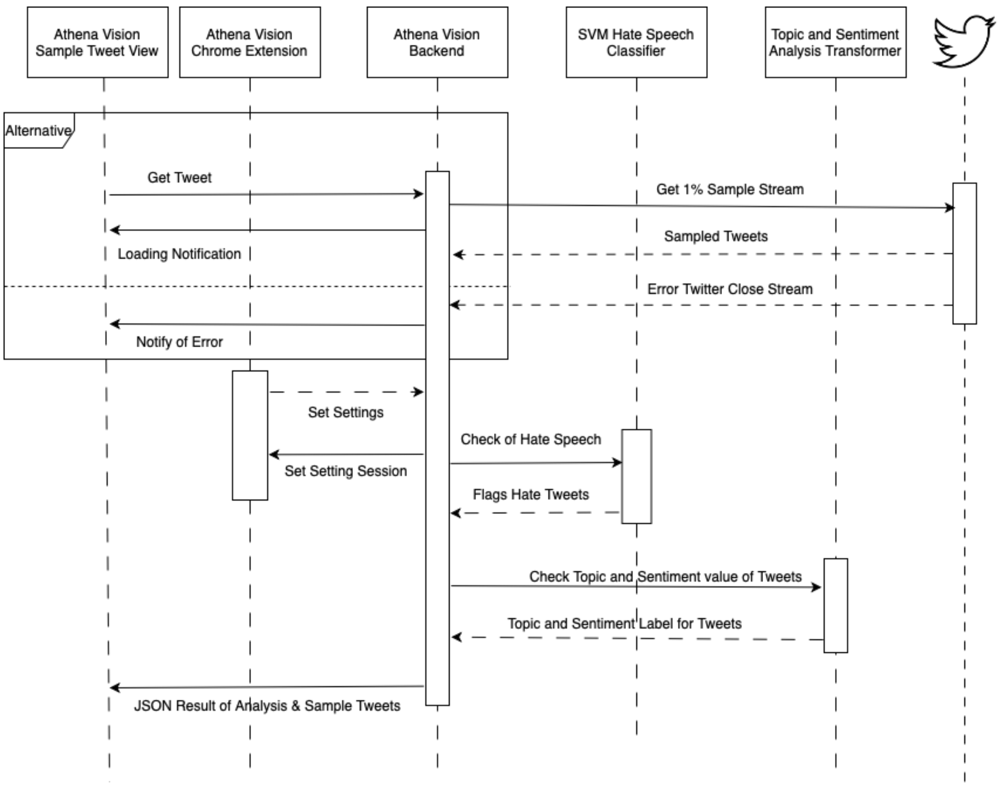

Athena Vision Sample Tweet View
Athena Vision Chrome Extension
Athena Vision Backend
SVM Hate Speech Classifier
Topic and Sentiment Analysis Transformer
Alternative
Get Tweet
Get 1% Sample Stream
Sampled Tweets
Loading Notification
Error Twitter Close Stream
Notify of Error
Set Settings
Check of Hate Speech
Set Setting Session
Flags Hate Tweets
Check Topic and Sentiment value of Tweets
Topic and Sentiment Label for Tweets
JSON Result of Analysis & Sample Tweets

Fig. 2: Sequence Diagram of Entire System

3.1.2 Activity Diagram

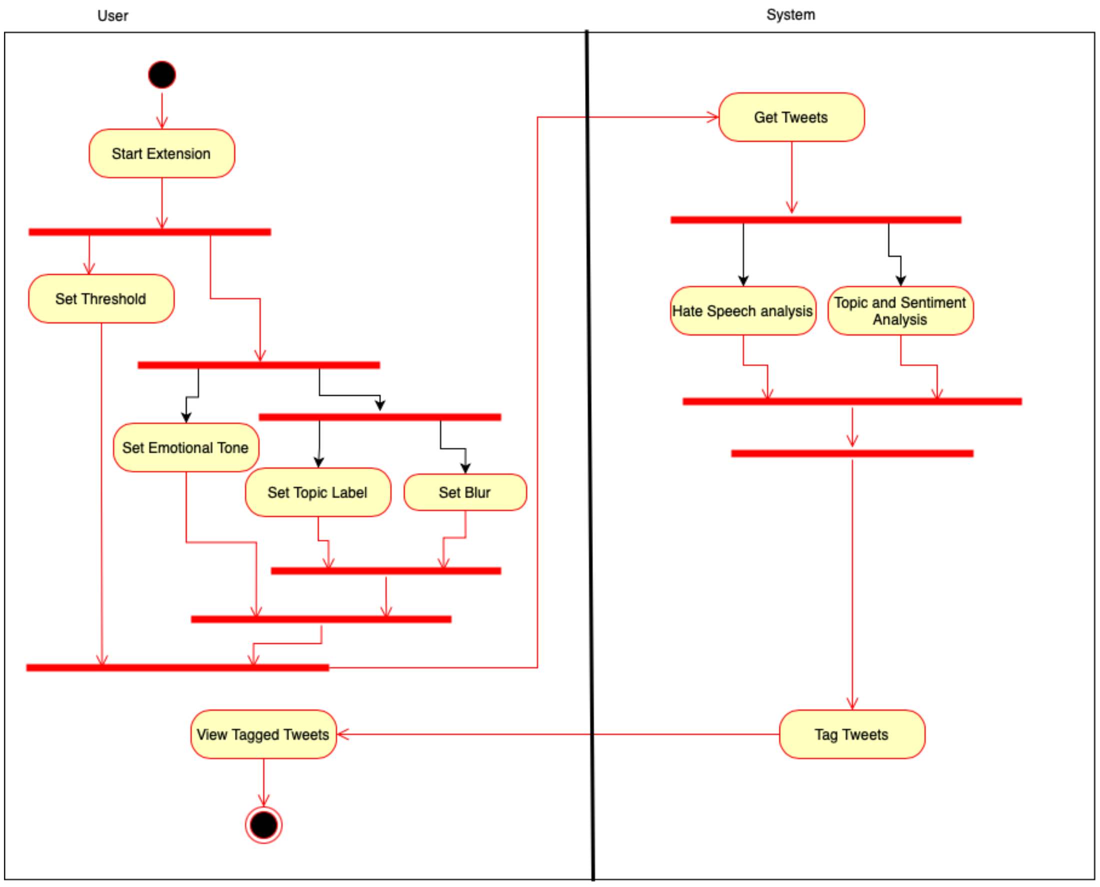


Fig. 3: Activity Diagram of Entire System

Sequence starts by connecting to "https://api.twitter.com/2/tweets/sample/stream" to fetch 1% of all live tweets on English and Nepali Language. Tweet is then passed through custom made hate speech classifier, and topic and sentiment analysis using hugging face transformer API. Tweets are only analyzed if allowed by chrome extension. Analyzed tweets are passed to the frontend to be viewed.

4 SYSTEM DESIGN

4.1 Design

4.1.1 Flow Diagram of System

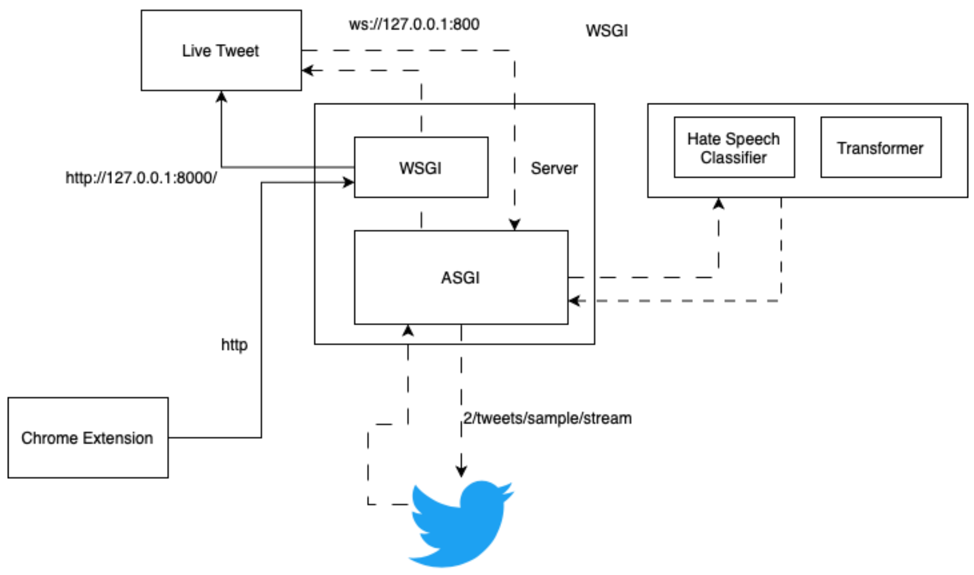


Fig. 4: Flow Diagram of Entire System

Localhost is connected on port 8000 using browser, which initiate the process of fetching and analyzing tweets. WSGI renders the loading skeleton on the browser for the user, at the same time ASGI will connect to Twitter at 2/tweets/sample/stream to fetch 1% of all live tweets, and establish a web socket connection with the frontend rendered by WSGI.

Chrome extension has control over which analysis should be performed (hate speech, topic, or sentiment). Based on the settings, session is set in WSGI which is accessed by ASGI through cache.

If hate speech flag is set, then sampled tweets will be analyzed for hate speech, and if topic or sentiment flag are set, hugging face will be used to perform hate and topic analysis on sampled tweets. After analysis, tweets are sent back to the frontend, along with their context annotation tags, to be displayed for the user.

4.1.2 Deployment Diagram

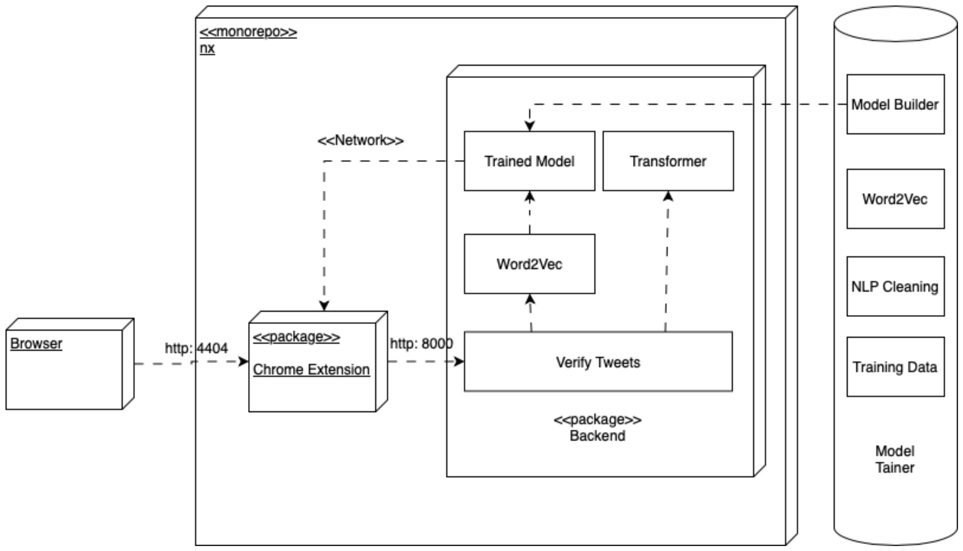


Fig. 5: Development Diagram of System

Entire project was developed using Monolithic architecture, making deployment very easy and fast. Browser connects to chrome extension which can set different flags for the entire system. Port 8000, will render a loading screen and fetch 1% of sample stream tweets for analysis on the trained model and hugging face API transformers.

Model training was done in Google Collaboratory (Google Colabs), using NLP techniques, and word2vec [1] for vectorization.

5 IMPLEMENTATION AND TESTING

5.1 Implementation

The core of the system comprises the hate speech classification model and, the topic and sentiment analysis models. To begin with, a Twitter hate speech dataset from Kaggle was used to develop the hate speech classification model. The entire process of data cleaning, pre-processing, and feature engineering, as well as model training and validation, was carried out on Google Colab. The feature engineering and model training were executed in three phases, with multiple experiments in each phase, as detailed in Section 5.1.1 - Implementation Details of Modules. Additionally, a pre-trained transformer from the Hugging Face library was selected for topic and sentiment analysis on Nepali and English tweets.

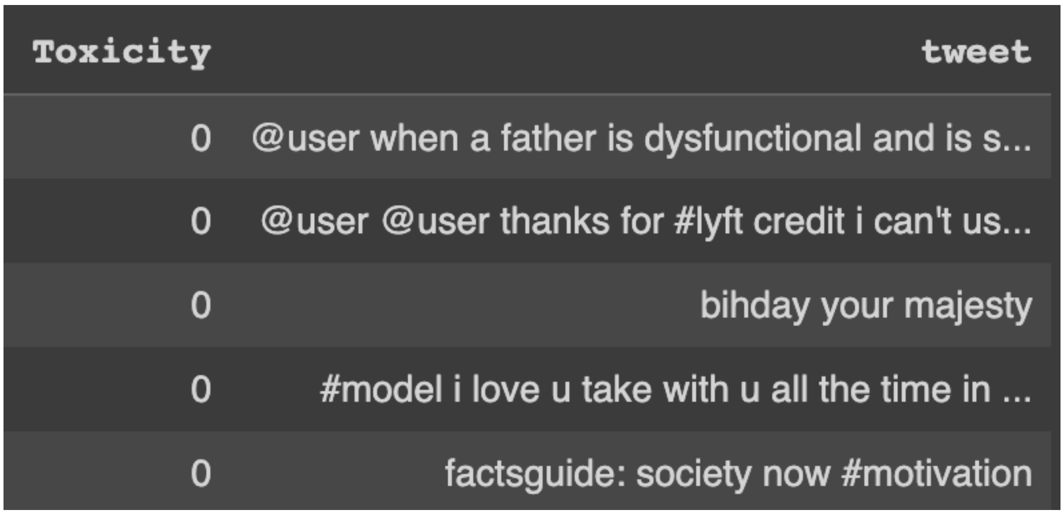
A monolithic architecture was set up using nx for seamless development of both the frontend and backend on the local machine. The frontend of the project, a Chrome extension, was developed in ReactJS with the setup using vite.js for fast Hot Module Reload (HMR) using esbuild. For improved functionality and ergonomic visibility during development, the SCSS styling was used instead of pure CSS, with the Neumorphic design pattern applied in the Chrome extension design.

The backend of the project was developed using Python and Django with Asynchronous Server Gateway Interface (ASGI) to establish a web-socket connection.

5.1.1 Implementation Details of Modules

Methodology for Model Training and Development

To initiate the development of the Twitter hate speech classifier, the initial step involved collecting a dataset named "Toxic Tweets Dataset" [13] from Kaggle, comprising of 56,745 tuples and 2 features.



Toxicity	tweet
0	@user when a father is dysfunctional and is s...
0	@user @user thanks for #lyft credit i can't us...
0	bihday your majesty
0	#model i love u take with u all the time in ...
0	factsguide: society now #motivation

Fig. 6: View of Dataset

In the subsequent step, a countplot was employed to visualize the balance of the label within the dataset.

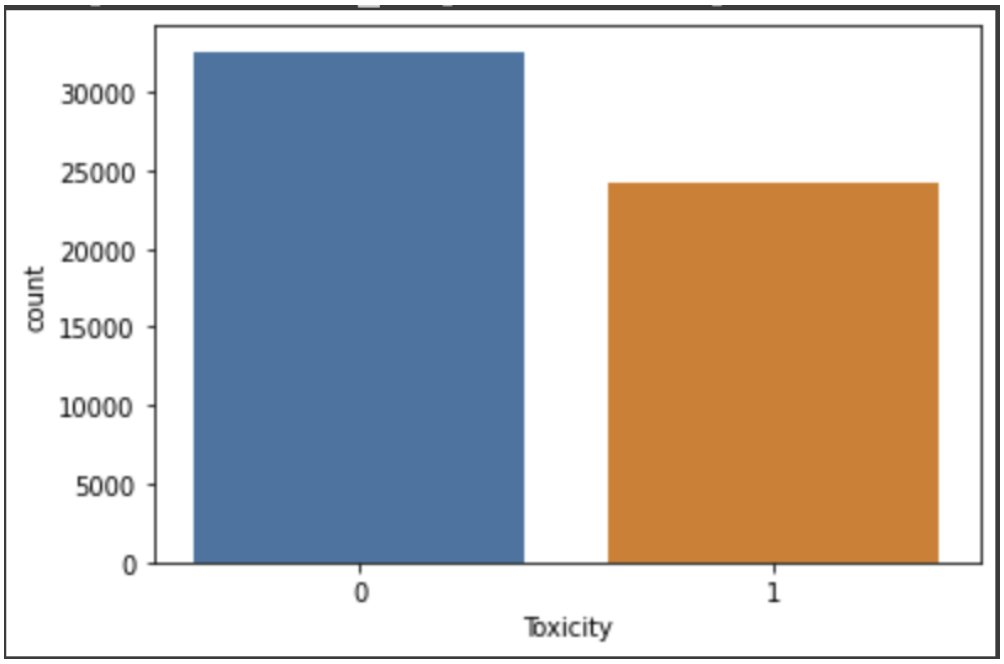


Fig. 7: Count plot of Labels

The development of a Twitter hate speech classifier involved three phases:

The first phase focused on data cleaning, feature engineering, and experimentation using several classification models, including SVM, AdaBoost Classifier, and Random Forest. The dataset was pre-processed using various techniques such as lowercase, contraction fix, removal of stop-words, URLs, HTML decoding, extra spaces, punctuation, emojis, diacritical marking, unicode decoding, and Porter Stemming. Following the data cleaning process, a word cloud visualization was developed.

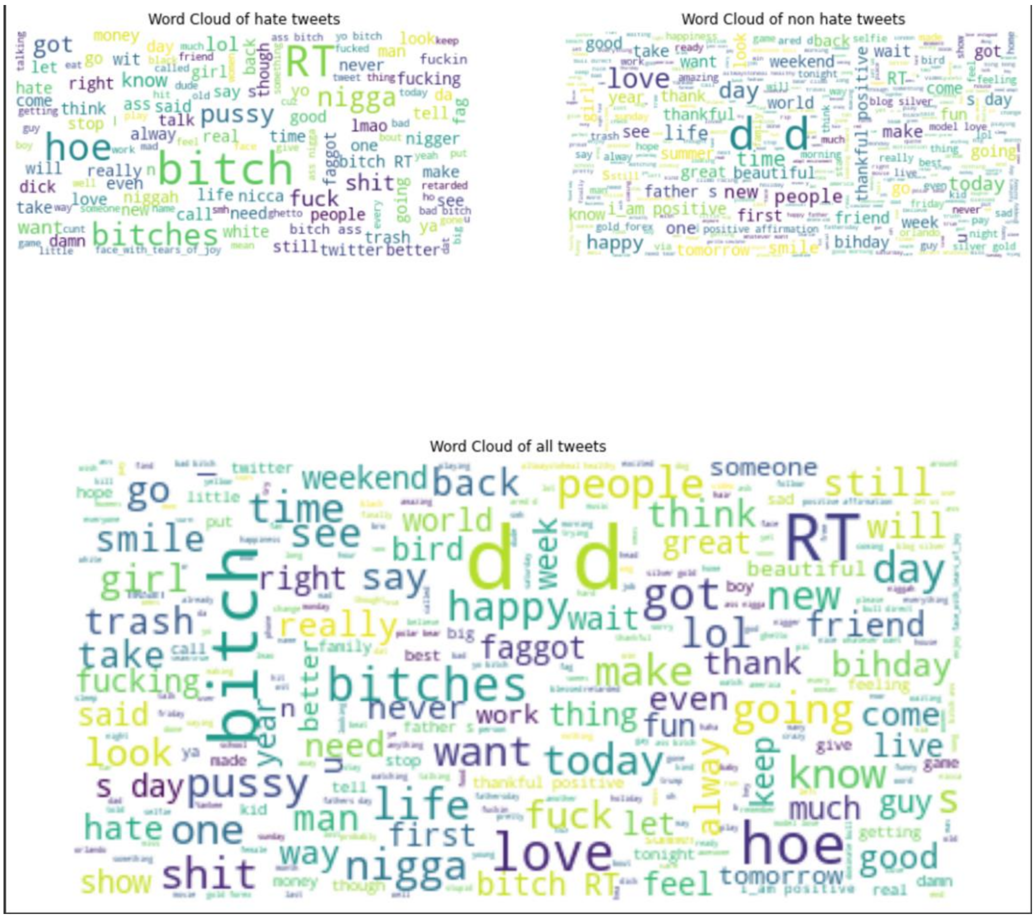
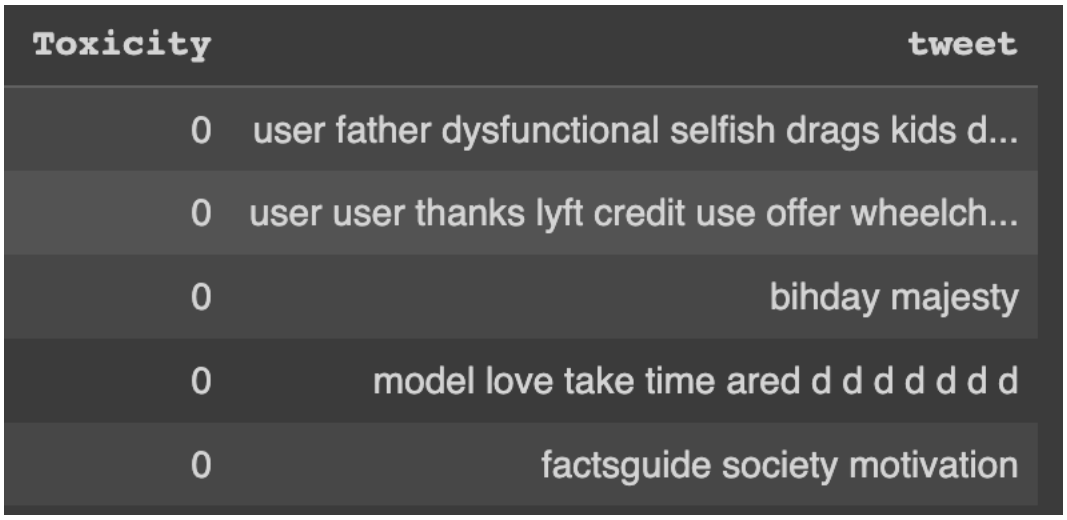


Fig. 8: Word Cloud of tweets



Toxicity	tweet
0	user father dysfunctional selfish drags kids d...
0	user user thanks lyft credit use offer wheelch...
0	bihday majesty
0	model love take time ared d d d d d d d
0	factsguide society motivation

Fig. 9: After preprocessing using TF-IDF

Following that vectorization was done using TF-IDF. In this experimentation phase, the performance of the classifiers was evaluated using F1-score.

In the second phase, the same pre-processing techniques were used as in the first phase, and vectorization was done using Stanford's Glove [10] with glove.twitter.27B.100d.txt to create an embedding matrix of 100 x 25. Instead of using machine learning models, neural networks were developed; LSTM and Bi-LSTM models were used for classification, and hyperparameter tuning was performed for Bi-LSTM. Stratified K-fold cross-validation was also done to ensure that the models generalize well.

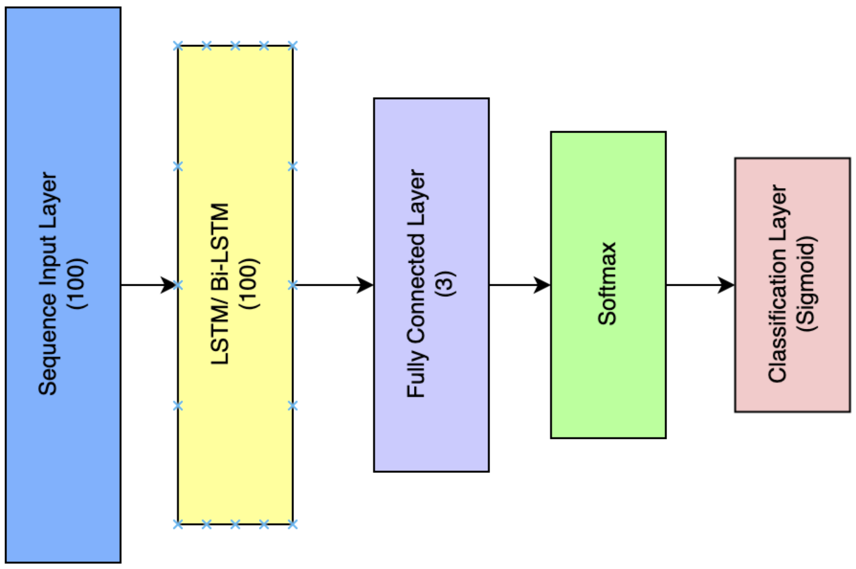
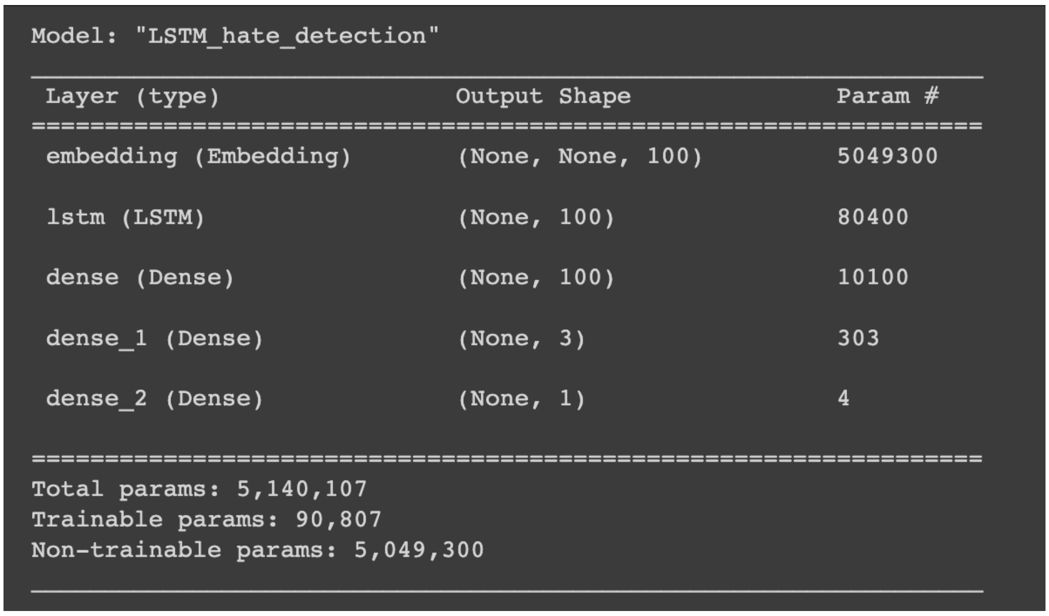
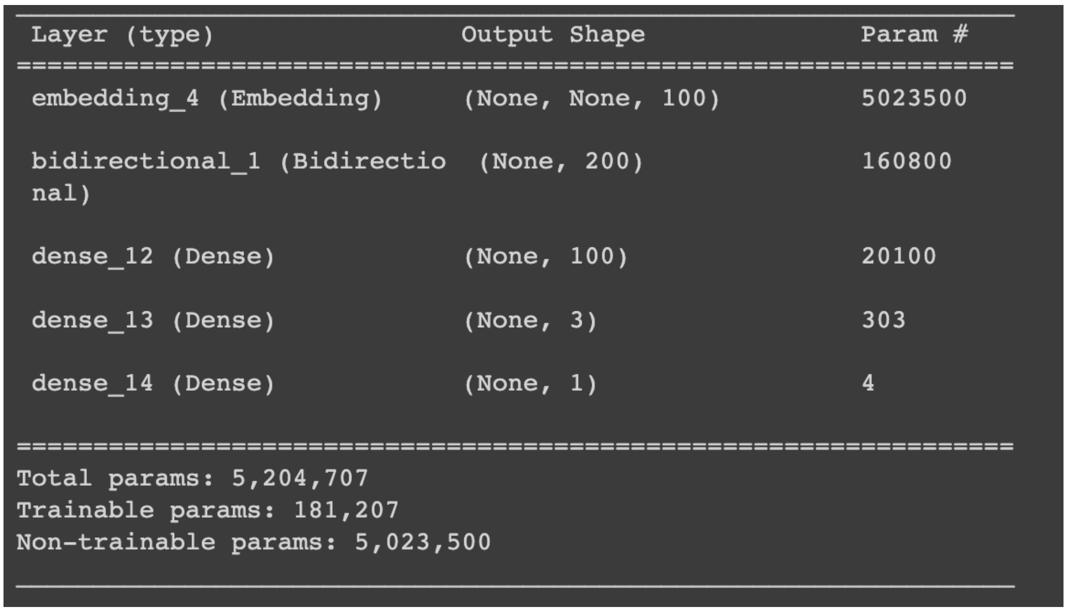


Fig. 10: Architecture of LSTM Neural Network



```
Model: "LSTM_hate_detection"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 100)         5049300

 lstm (LSTM)                 (None, 100)               80400

 dense (Dense)               (None, 100)               10100

 dense_1 (Dense)             (None, 3)                 303

 dense_2 (Dense)             (None, 1)                 4

=================================================================
Total params: 5,140,107
Trainable params: 90,807
Non-trainable params: 5,049,300
_________________________________________________________________
```

Fig. 11: Structure of LSTM Neural Network

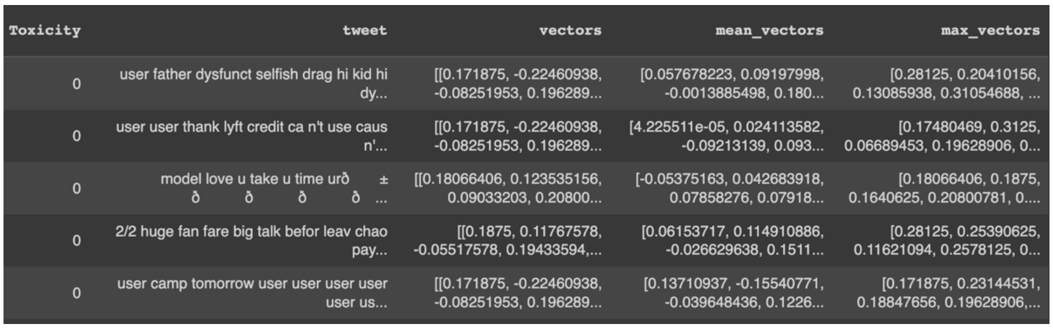


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, None, 100)         5023500

 bidirectional_1 (Bidirectio  (None, 200)              160800
 nal)

 dense_12 (Dense)            (None, 100)               20100

 dense_13 (Dense)            (None, 3)                 303

 dense_14 (Dense)            (None, 1)                 4

=================================================================
Total params: 5,204,707
Trainable params: 181,207
Non-trainable params: 5,023,500
_________________________________________________________________
```

Fig. 12: Structure of Bi-LSTM Neural Network

To evaluation of the metric of LSTM/ Bi-LSTM based neural network was done using train loss vs validation loss, train accuracy vs validation accuracy, and Receiver Operating Characterstics (ROC) curve.

In the third phase, a new set of data cleaning techniques were applied, including tokenization, removal of URLs, mentions, hashtags, lowercase conversion, removal of punctuation, Porter stemming, and stop-word removal. The vectorization technique used was Google's Word2vec. The mean and max pooling of vectors was experimented with the following classifiers: Logistic Regression, Gaussian Naïve Bayes, SVM, and Random Forest Classifier.



Toxicity	tweet	vectors	mean_vectors	max_vectors
0	user father dysfunct selfish drag hi kid hi dy...	[[0.171875, -0.22460938, -0.08251953, 0.196289...	[0.057678223, 0.09197998, -0.0013885498, 0.180...	[0.28125, 0.20410156, 0.13085938, 0.31054688, ...
0	user user thank lyft credit ca n't use caus n'...	[[0.171875, -0.22460938, -0.08251953, 0.196289...	[4.225511e-05, 0.024113582, -0.09213139, 0.093...	[0.17480469, 0.3125, 0.06689453, 0.19628906, 0...
0	model love u take u time urð ± ð ð ð ð ...	[[0.18066406, 0.123535156, 0.09033203, 0.20800...	[-0.05375163, 0.042683918, 0.07858276, 0.07918...	[0.18066406, 0.1875, 0.1640625, 0.20800781, 0....
0	2/2 huge fan fare big talk befor leav chao pay...	[[0.1875, 0.11767578, -0.05517578, 0.19433594,...	[0.06153717, 0.114910886, -0.026629638, 0.1511...	[0.28125, 0.25390625, 0.11621094, 0.2578125, 0...
0	user camp tomorrow user user user user user us...	[[0.171875, -0.22460938, -0.08251953, 0.196289...	[0.13710937, -0.15540771, -0.039648436, 0.1226...	[0.171875, 0.23144531, 0.18847656, 0.19628906,...

Fig. 13: After word2vec and Mean/Max Pooling

Further experiments used stratified K-fold cross-validation was done with the mean pooling technique. In this phase, evaluation was done using f1-score, accuracy, recall, precision, ROC curve and confusion matrix.

Finally, stacking generalization was used with SVM, Random Forest Classifier, and Gaussian Naïve Bayes as base models and Logistic Regression as the meta model.

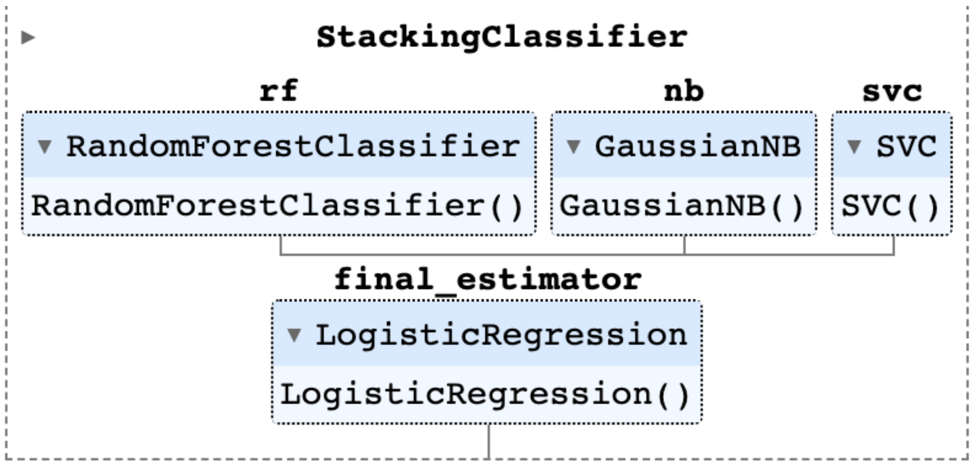


Fig. 14: Stacking Generalization Model Architecture

Throughout the development process, exploratory analysis was performed, which included an analysis of the balance of the dataset and a word cloud. The goal was to build an accurate and efficient hate speech classifier for Twitter data. The different phases of the development process allowed for a thorough evaluation of various pre-processing techniques, vectorization methods, and classification models to achieve the optimal performance of the hate speech classifier. Result of experiments done in different phases are explained in detail in Section 5.2 - Result Analysis.

Summary of data cleaning technique and methodology used during training of mode is provided below:

Table 1: Data Cleaning Technique

Data cleaning	**Reference Number**
Lowercase	1
Contraction fix	2
Removing stop-word	3
URL removing	4
Decoding HTML	5
Removing Extra Space	6
Replace Emoji	7
Remove Diacritical mark	8
Unicode decode	9
Nltk twitter tokenizer	10
Nltk word_tokenizer	11
Porter stemming	12
Removing mentions and hashtags	13
Removing Punctuation	14

Table 2: Implementation Details of Different Experiments

Experiments	**Models**	**Data cleaning (in sequence) r/** Table 1: Data Cleaning Technique	**Vectorization method**	**Metrics**
Phase 1				
P1E1	SVM	1, 2, 3, 4, 5, 6, 14, 7, 8, 9, 12	Tf-idf with bi-gram	F1-score
P12E2	AdaBoost			
P1E3	Random Forest (RF)			
Phase 2				
P2E1	LSTM	10, 1, 2, 3, 4, 5, 6, 14, 7, 8, 9, 12	GloVe	Train loss vs validation loss;
P2E2	Bi-LSTM			

				Train accuracy vs Validation Accuracy; ROC curve
P2E3	Hyperparameter tuning of Bi-LSTM			Train loss vs validation loss; Train accuracy vs Validation Accuracy;
P2E4	Stratified K fold cross validation of LSTM w/ 5 folds			Mean train accuracy vs Mean validation accuracy of every fold
P2E5	Stratified K fold cross validation of LSTM w/ 10 folds			
Phase 3				
P3E1	LR	11, 4, 13, 1, 14, 12, 3	Word2Vec, Max Pooling	F1-score; Accuracy; Recall; Precision; Confusion Matrix; ROC curve
P3E2	Gaussian Naïve Bayes (GNB)			
P3E3	RF			
P3E4	SVM			
P3E5	LR		Word2Vec, Mean Pooling	
P3E6	GNB			
P3E7	SVM			
P3E8	RF			
P3E9	Stratified k fold LR w/ 5 folds			
P3E10	Stratified k fold GNB w/ 5 folds			
P3E11	Stratified k fold RF w/ 5 folds			

P3E12	Stratified k fold SVM w/ 5 folds			
P3E13	Stacking generalization- Base mode: GNB, SVM, RF; meta model: LR			

Configuring Monolithic repo

The project development was structured in a monorepo architecture, facilitating modular development of the React-based frontend, namely the Chrome extension. The implementation of the nx-react plugin and the nx-workspace plugin enabled the creation of a lightweight frontend, while additional components can be added as required.

Subsequently, a contained virtual environment for the backend was created using Pipenv, which generates a dependency configuration in TOML file format that can be utilized to set up the Django project, including all Python-based dependencies, alongside Node modules using Yarn.

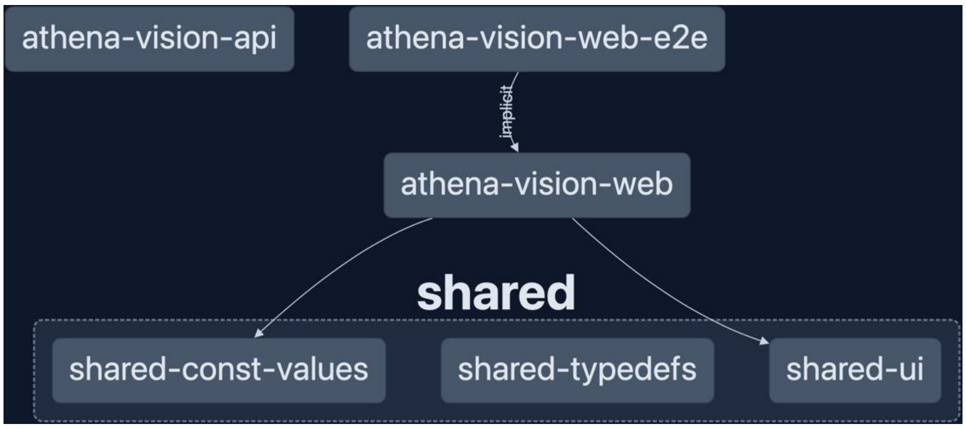


Fig. 15: Monorepo Project Structure

5.2 Result Analysis

In Section 5.1.1—Implementation Details of Modules, we conducted several experiments in three phases to evaluate the effectiveness of different text vectorization techniques and models. These phases included tf-idf with bi-gram trained on machine learning models, 100 x 25 GloVec [10] embedding matrix trained on neural network, and Word2Vec [1] vectors with mean and max pooling trained on machine learning models. The impact of these techniques was thoroughly analyzed and assessed.

During Phase 1, we performed experiments using SVM, AdaBoost Classifier, and Random Forest models after text vectorization was carried out using TF-IDF with bi-gram. Among these models, the Random Forest model exhibited the best F-score of 0.759. However, due to its relatively low score, this approach was not deemed suitable for our needs.

Table 3: F-score of Phase 1 Experiments

Experiment	**Model**	**F-score**
P1E1	SVM	0.677
P1E2	AdaBoost	0.422
P1E3	Random Forest	**0.759**

In Phase 2, the text vectorization technique employed was GloVec [10], and the performance of the LSTM, Bi-LSTM, and Stratified K-fold cross-validation models were evaluated. However, the LSTM and Bi-LSTM models were observed to be overfitting, displaying good performance on the training data but poor generalization to the test data.

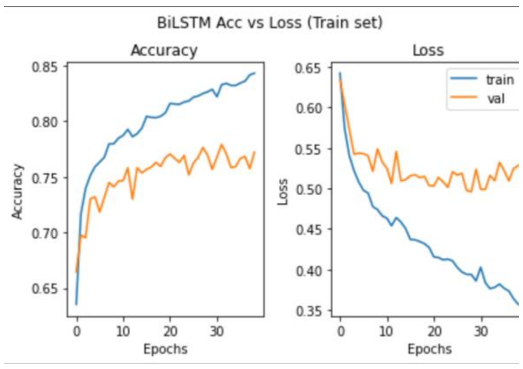


Fig. 16: Bi-LSTM Train vs. Validation Accuracy and Loss

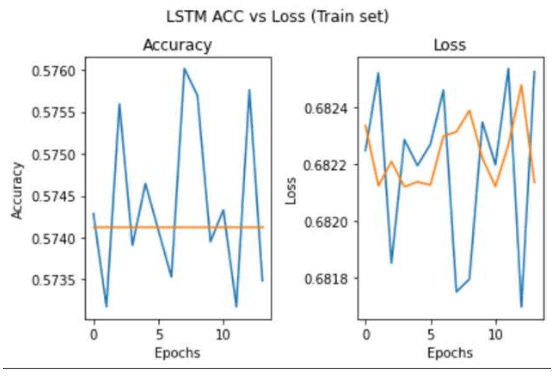


Fig. 17: LSTM Train vs. Validation Accuracy and Loss

Table 4: Phase 2 Experiments Results (Loss and Accuracy) on Test set

Experiments	**Hidden Layer**	**Loss**	**Accuracy**
P2E1	LSTM	0.682	0.574
P2E2	Bi-LSTM	**0.51**	**0.76**

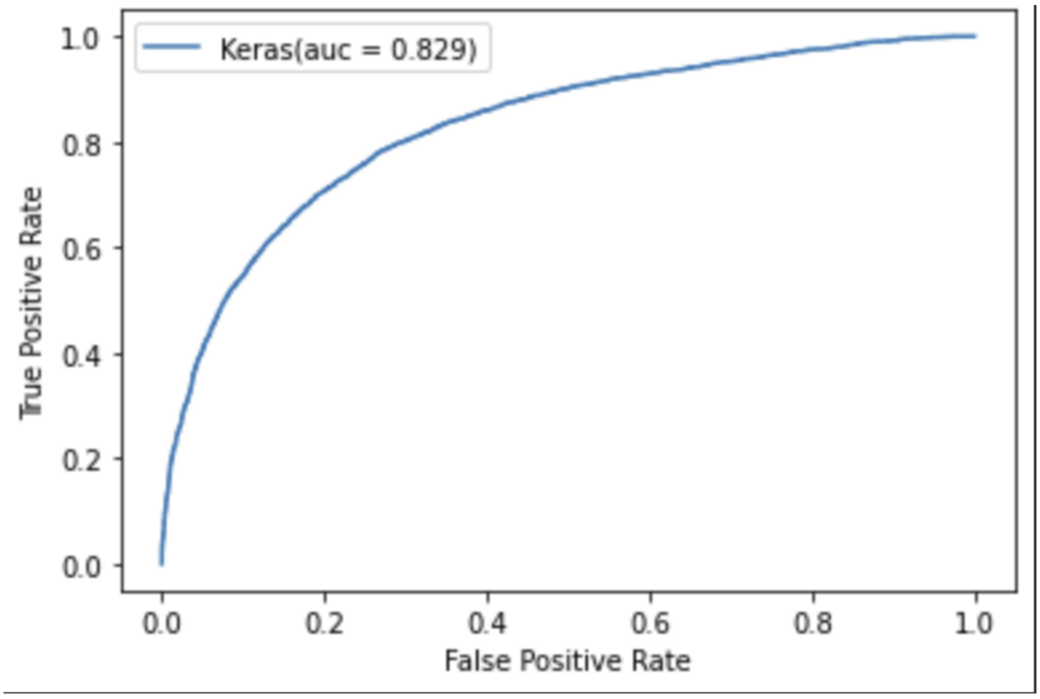
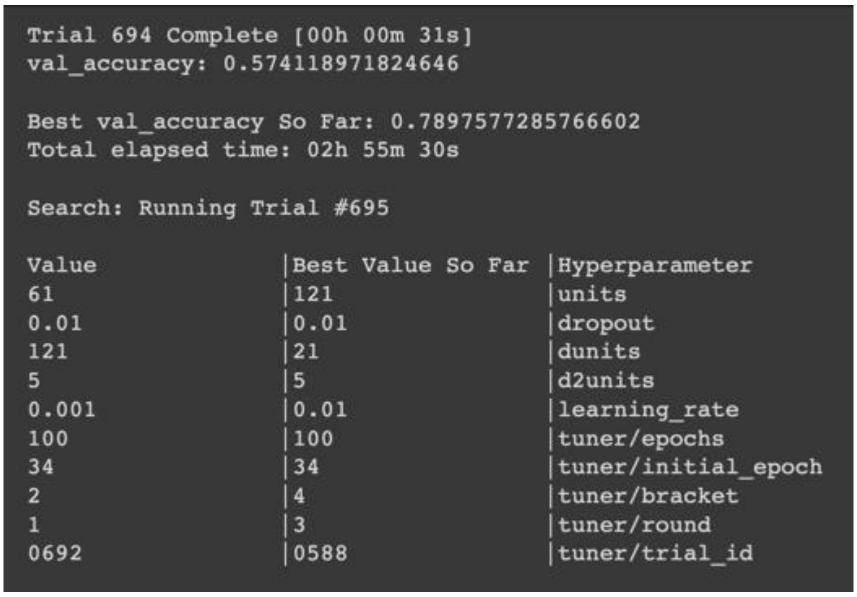


Fig. 18: ROC curve Bi-LSTM

Despite hyperparameter tuning, the highest validation accuracy achieved was less than 80% even after 694 trials.



```
Trial 694 Complete [00h 00m 31s]
val_accuracy: 0.574118971824646

Best val_accuracy So Far: 0.7897577285766602
Total elapsed time: 02h 55m 30s

Search: Running Trial #695

Value             |Best Value So Far |Hyperparameter
61                |121               |units
0.01              |0.01              |dropout
121               |21                |dunits
5                 |5                 |d2units
0.001             |0.01              |learning_rate
100               |100               |tuner/epochs
34                |34                |tuner/initial_epoch
2                 |4                 |tuner/bracket
1                 |3                 |tuner/round
0692              |0588              |tuner/trial_id
```

Fig. 19: Best Parameters and Validation Accuracy of Hyperband Bi-LSTM

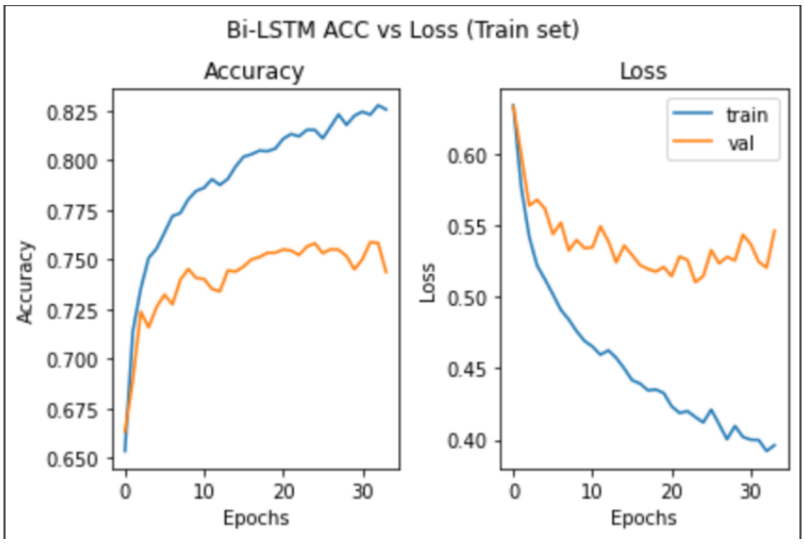


Fig. 20: Hyperband Tuned Bi-LSTM Train vs. Validation

Additionally, K-fold cross-validation experiments were conducted, with 5 folds for LSTM and 10 folds for Bi-LSTM, which confirmed the overfitting issue of the LSTM and Bi-LSTM models. Therefore, LSTM/ Bi-LSTM based neural network techniques were deemed unsuitable for the dataset.

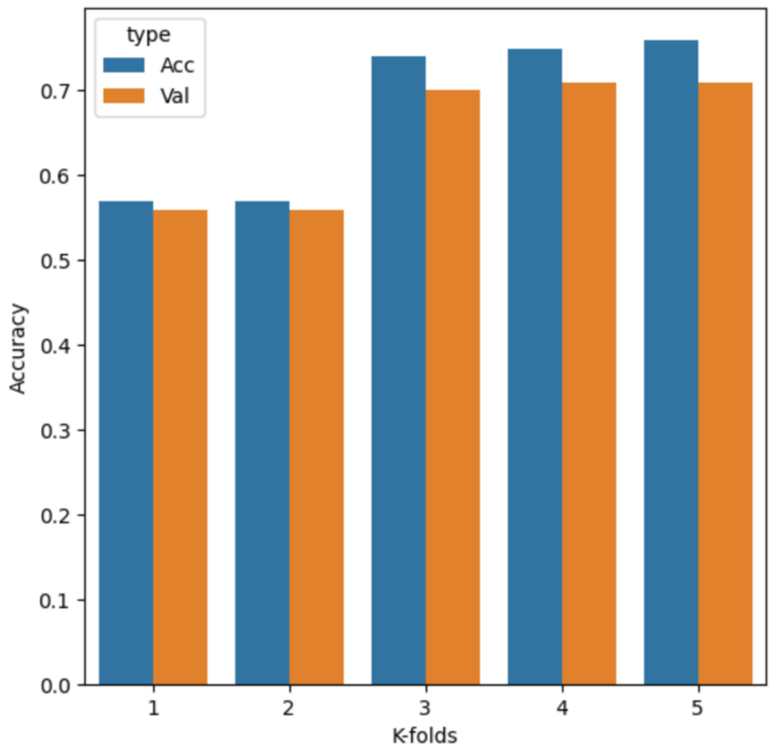


Fig. 21: Train vs Validation Accuracy LSTM

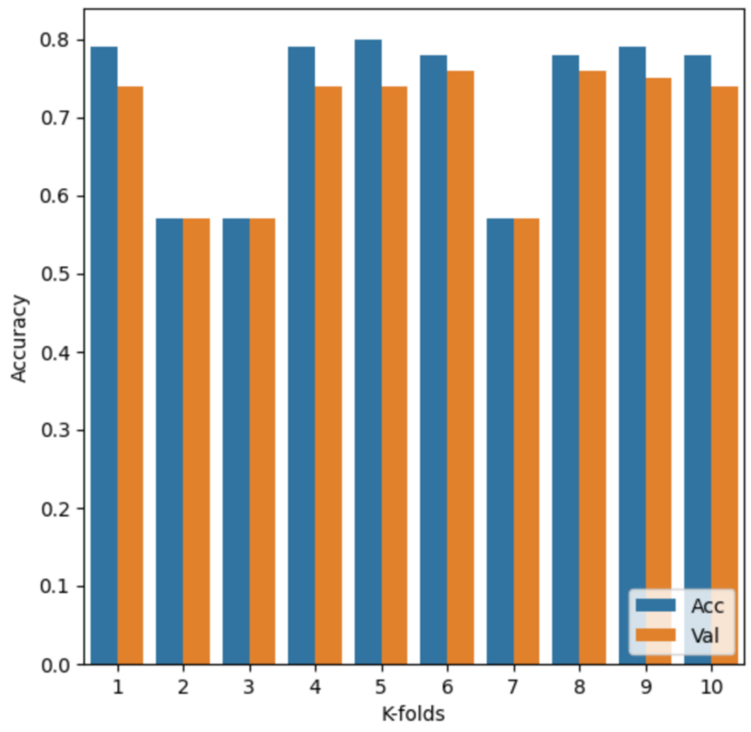


Fig. 22: Train vs. Validation Accuracy Bi-LSTM

In Phase 3, a novel feature engineering technique was employed to vectorize the text through the use of word2vec, and mean and max pooling. Subsequently, experiments were conducted using a range of machine learning models, such as Logistic Regression, Gaussian Naive Bayes, SVM, and Random Forest, in conjunction with mean and max pooling of vectors. The outcomes demonstrated that all models performed effectively with the mean of vectors. Of the models tested, the SVM model yielded the best performance with an F-score of 0.89, accuracy of 0.91, recall of 0.86, precision of 0.93, and an AUC of ROC curve of 0.966.

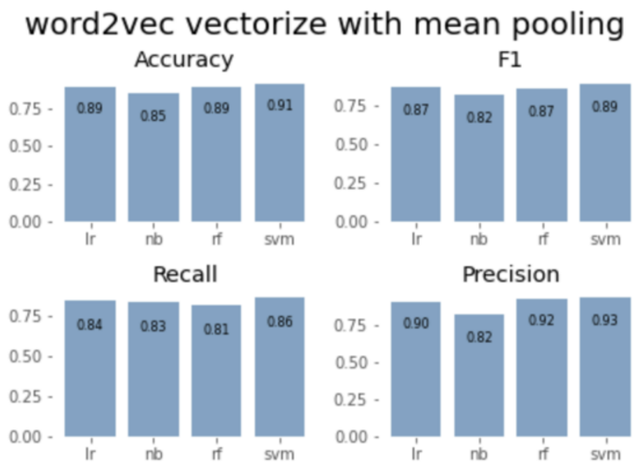


Fig. 23: Bar plot of Different Model Evaluation Metric (Mean Pooling)

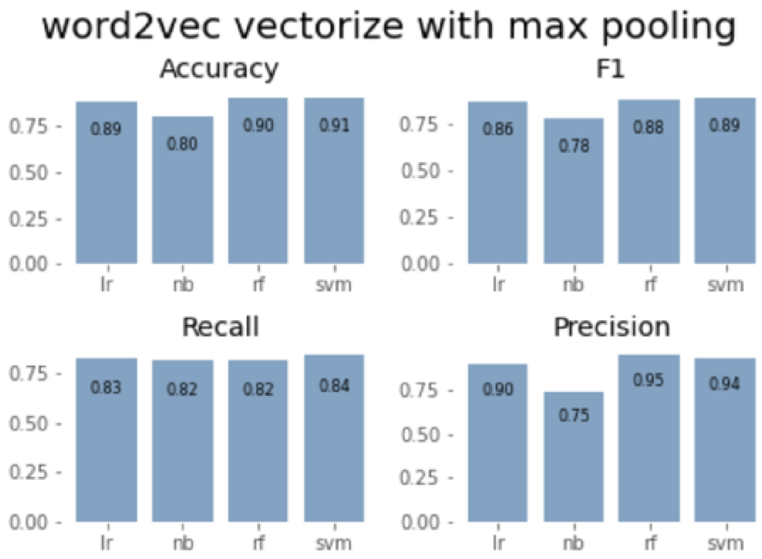


Fig. 24: Bar plot of Different Model Evaluation Metric (Max Pooling)

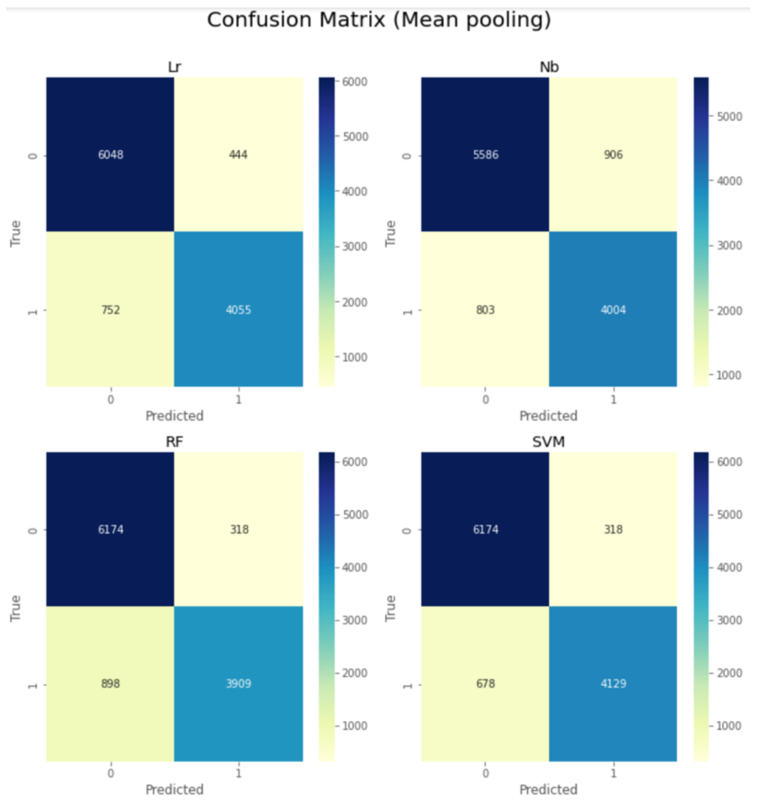


Fig. 25: Confusion Matrix of Different Model (Mean Pooling)

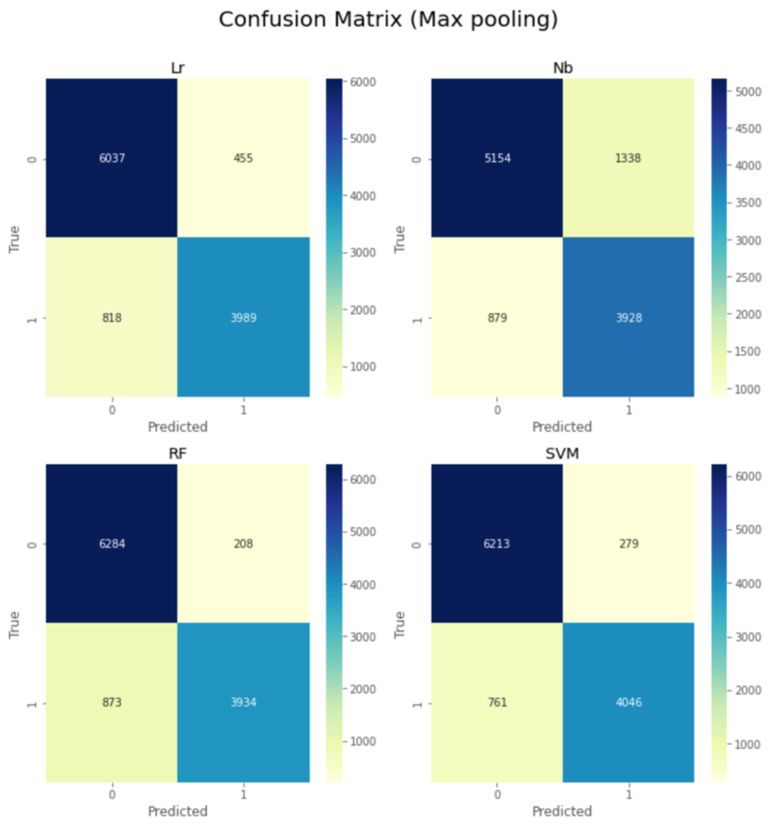


Fig. 26: Confusion Matrix of Different Model (Max Pooling)

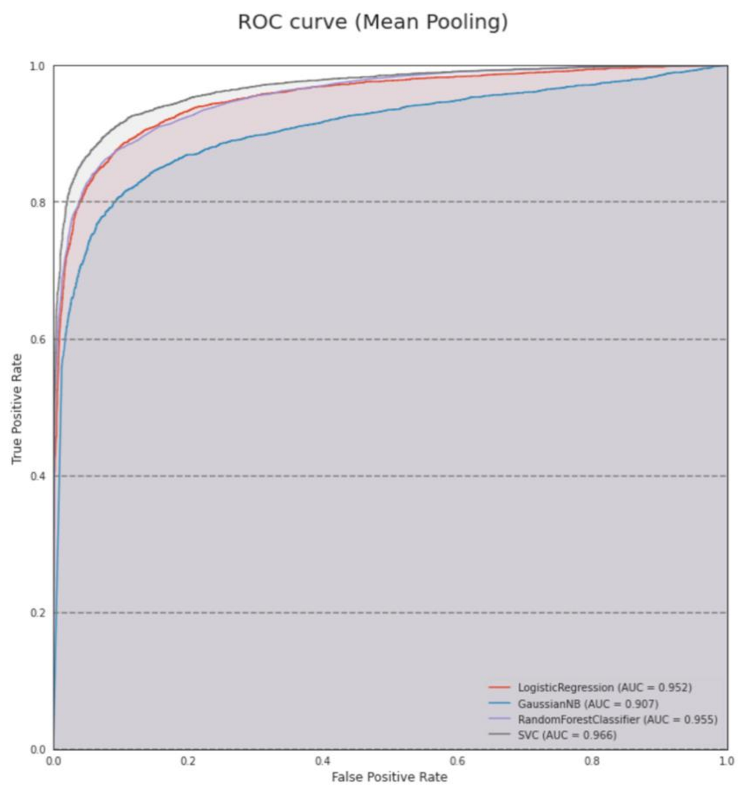


Fig. 27: ROC curve of Different Models (Mean Pooling)

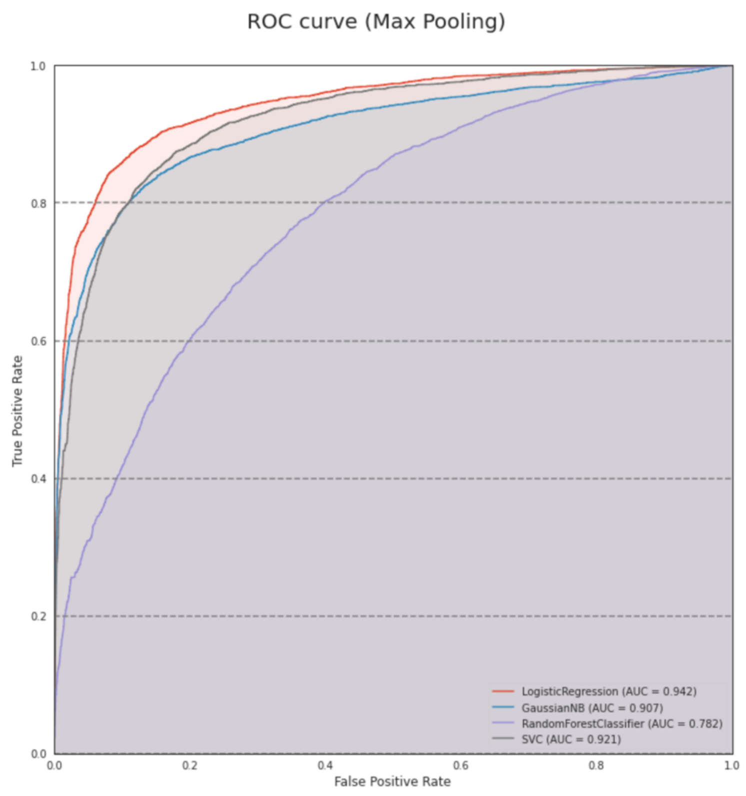


Fig. 28: ROC curve of Different Models (Max Pooling)

Furthermore, a stratified K-fold cross-validation with four folds was conducted on the train-test split, which demonstrated outstanding results. The SVM model was shown to have the highest mean validation accuracy of 0.91.

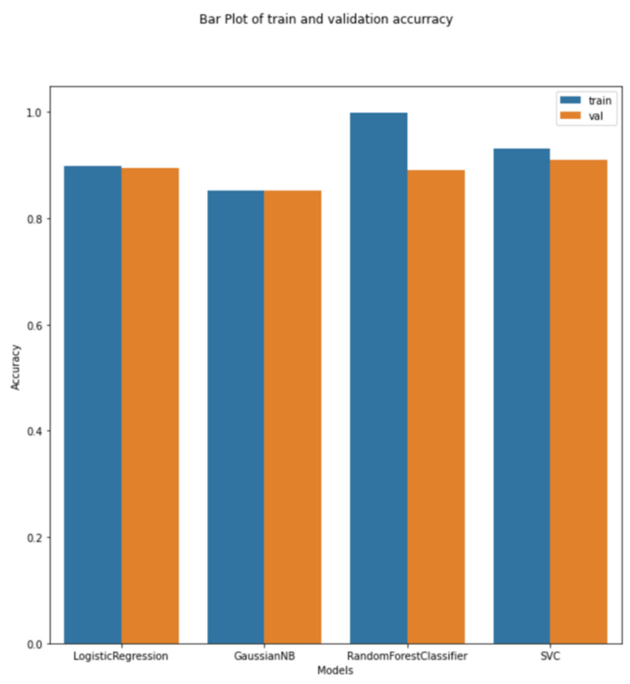


Fig. 29: Train vs. Validation Accuracy of Different Models

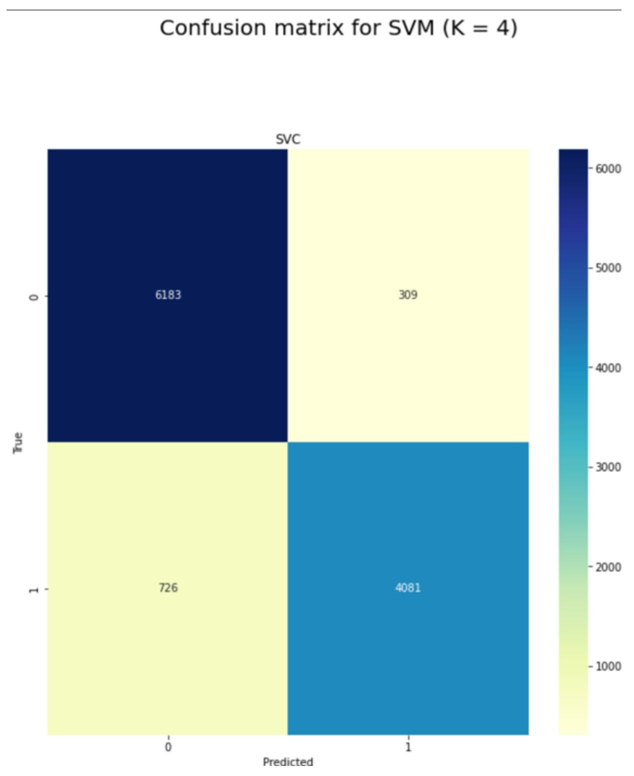


Fig. 30: Confusion Matrix of Stratified K-fold cross validation SVM (K=4)

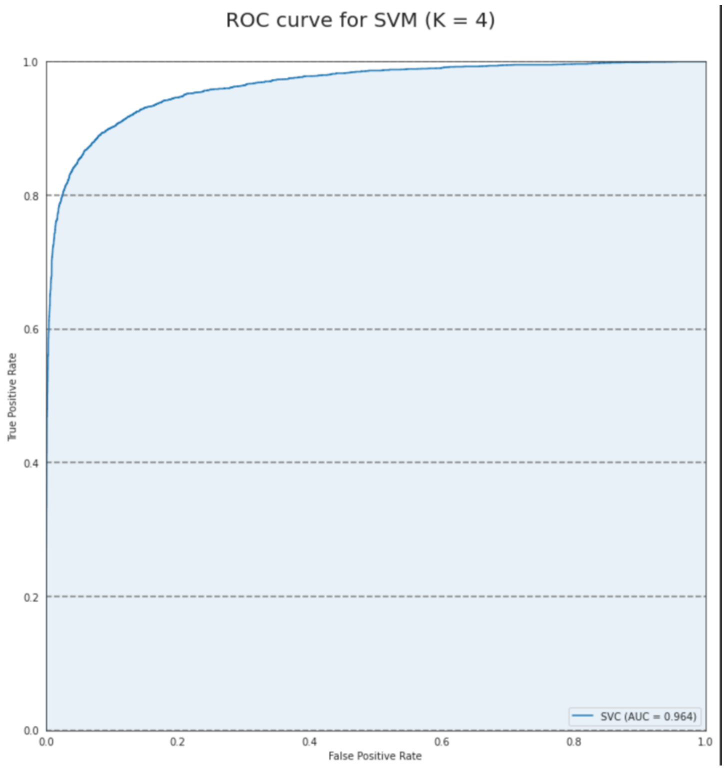


Fig. 31: ROC curve of Stratified K-fold SVM (K=4)

Finally, the last experiment in Phase 3 involved stacking generalization, with SVM, Random Forest, and Gaussian Naïve Bayes utilized as the base models, and Logistic Regression as the meta-model.

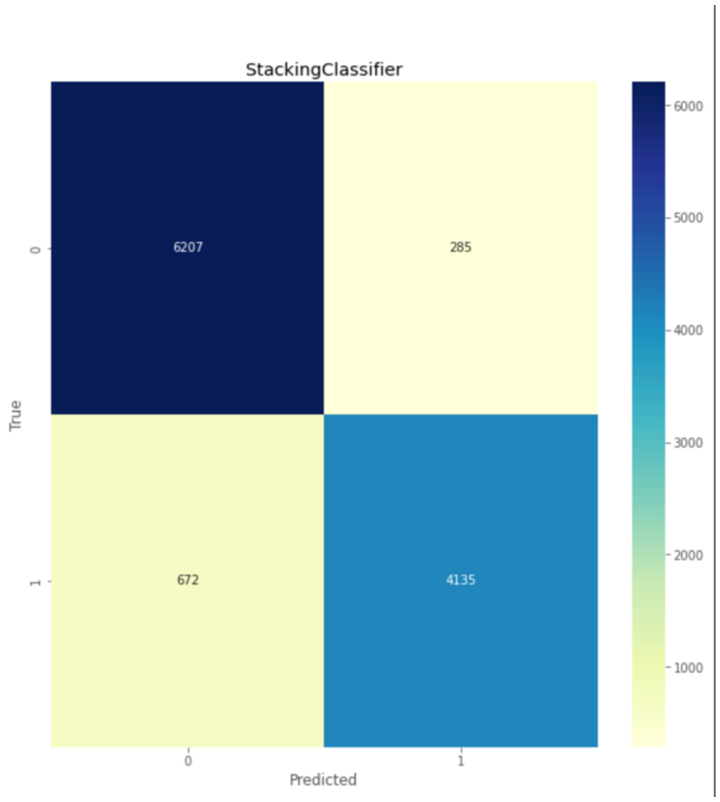


Fig. 32: Confusion Matrix of Stacking Generalization

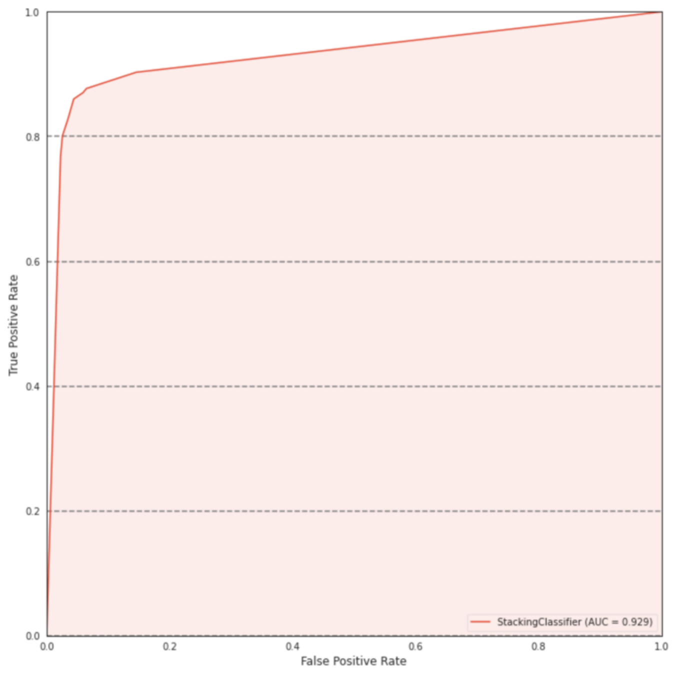


Fig. 33: ROC curve of Stacking Generalization

However, it is worth noting that the novel SVM model still outperformed the generalized model, with an AUC value of 0.966 compared to the AUC of stacking generalization of 0.929.

Table 5: Evaluation Result for Different Experiments

Experiment	**Model**	**Evaluation**
	Phase 1	
P1E1	SVM	F1-score = 0.67
P1E2	AdaBoost	F1-score =0.42
P1E3	Random Forest	F1-score = 0.75
	Phase 2	
P2E1	LSTM	Test Accuracy = 0.57 Test Loss = 0.68
P2E2	Bi-LSTM	Test Accuracy = 0.76 AUC value = 0.51
P2E3	Hyperparameter tuning of Bi-LSTM	Validation Accuracy = 0.78
P2E4	Stratified K fold cross validation of LSTM w/ 5 folds	Mean Train Acc = 0.33 Mean Validation Acc= 0.32
P2E5	Stratified K fold cross validation of LSTM w/ 10 folds	Mean Train Acc = 0.36 Mean Validation Acc = 0.34
	Phase 3	

P3E1	LR (Max pooling)	F-score = 0.86
		Accuracy = 0.89
		Recall = 0.83
		Precision = 0.9
		AUC = 0.942

P3E2	GNB (Max pooling)	F-score = 0.78
		Accuracy = 0.8
		Recall = 0.82
		Precision = 0.75
		AUC = 0.9

P3E3	SVM (Max pooling)	F-score = 0.89
		Accuracy = 0.91

		Recall = 0.84
		Precision = 0.94
		AUC = 0.921

P3E4	RF (Max pooling)	F-score = 0.88
		Accuracy = 0.9
		Recall = 0.82
		Precision = 0.95
		AUC = 0.782

P3E5	LR (Mean pooling)	F-score = 0.87
		Accuracy = 0.89
		Recall = 0.84
		Precision = 0.9
		AUC = 0.952

P3E6	GNB (Mean pooling)	F-score = 0.82
		Accuracy = 0.85
		Recall = 0.83
		Precision = 0.82
		AUC = 0.9

P3E7	SVM (Mean pooling)	F-score = **0.89**
		Accuracy = **0.91**
		Recall = **0.86**
		Precision = **0.93**
		AUC = **0.966**

P3E8	RF (Mean pooling)	F-score = 0.87
		Accuracy = 0.89
		Recall = 0.81
		Precision = 0.92
		AUC = 0.955

P3E9	Stratified K-cross fold validation LR (Mean pooling) w/ 5 folds	Mean Train Acc = 0.89 Mean Validation Acc = 0.89

P3E10	Stratified K-cross fold validation GNB (Mean pooling) w/ 5 folds	Mean Train Acc = 0.85 Mean Validation Acc = 0.85
P3E11	Stratified K-cross fold validation RF (Mean pooling) w/ 5 folds	Mean Train Acc = 0.99 Mean Validation Acc = 0.89
P3E12	Stratified K-cross fold validation SVM (Mean pooling) w/ 5 folds	Mean Train Acc = 0.93 Mean Validation Acc = 0.91

P3E13	Stacking generalization-Base mode: GNB, SVM, RF; meta model: LR	F-score = 0.89
		Accuracy = 0.915
		Recall = 0.86
		Precision = 0.93
		AUC = 0.929

The result presented in this study provides compelling evidence that a novel Support Vector Machine (SVM) utilizing the "word2vec-google-new-300.txt" [1] text vectorizer, in conjunction with the appropriate data cleaning techniques outlined in the paper, is an effective tool for predicting hate speech on Twitter.

6 CONCLUSON AND FUTURE RECOMMENDATION

6.1 Conclusion

In conclusion, the project aims to address the issue of hate speech in social media, particularly Twitter. The proposed solution is to stream 1% of all Twitter data using Django and pass it through a hate speech classification algorithm, along with two transformer APIs for genre classification and sentiment analysis of English and Nepali tweets. The hate speech classifier was developed through 21 experiments divided into three phases, and it was found that the Word2Vec [1] algorithm along with Nobel SVM outperformed all the other approaches. The results show that by leveraging machine learning algorithms, a robust system can be created for hate speech analysis, which could potentially contribute to mitigating the negative impacts of hate speech on society.

6.2 Future Recommendation

Based on the project and the results obtained, there are a number of possible avenues for future work. One potential direction would be to explore the use of premium Twitter APIs to access a larger portion of the Twitter data stream and increase the accuracy and effectiveness of the hate speech classifier. Additionally, it may be worth exploring the development of a user-specific hate speech censoring system that operates on individual user timelines rather than on the entire Twitter data stream.

Beyond Twitter, there is also significant potential for applying similar hate speech detection and censorship techniques to other social media platforms. By expanding the scope of the analysis to include other platforms such as Facebook, Instagram, and YouTube, it may be possible to develop a more comprehensive system for detecting and preventing hate speech across multiple channels. Ultimately, the continued development and refinement of hate speech classification and censorship systems will be an important step towards promoting a more respectful and tolerant online environment.

REFERENCES

[1] T. M. Ilya, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *CoRR,* vol. 1301.3781, 16 Oct 2013.

[2] K. Saha, E. Chandrasekharan and M. D. Choudhury, "Prevalence and Psychological Effects of Hateful Speech in Online College Communities," Association for Computing Machinery, New York, 2019.

[3] C. Calvert, "Hate Speech and Its Harms: A Communication Theory Perspective," *Journal of Communication,,* vol. 47, no. 1, pp. 4-19, 1997.

[4] F. Barbieri, L. E. Anke and J. Camacho-Callados, "XLM-T: Multilingual Language Models in Twitter for Sentiment Analysis and Beyond," in *13th Conference on Language Resources and Evaluation*, Marseille, 2022.

[5] T. Kuzman, "Comparison of genre datasets: CORE, GINCO and FTD," 2022. [Online]. Available: https://github.com/TajaKuzman/Genre-Datasets-Comparison.

[6] Twitter, "Hateful conduct policy," Twitter Help Center, 08 2021. [Online]. Available: https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy. [Accessed 05 03 2023].

[7] S. Frenkel and K. Conger, "Hate Speech's Rise on Twitter Is Unprecedented, Researchers Find," The New York Times, 02 12 2022. [Online]. Available: https://www.nytimes.com/2022/12/02/technology/twitter-hate-speech.html. [Accessed 05 03 2023].

[8] S. Abra, S. Shaikh and Z. Hussain, "Automatic Hate Speech Detection using Machine Learning: A Comparative Study," *ResearchGate,* vol. 11, no. 8, 2020.

[9] A. Bisht, A. Singh, H. S. Bhadauria and V. Jitendra, "Detection of Hate Speech and Offensive Language in Twitter Data Using LSTM Model," *ResearchGate,* pp. 243-264, 2020.

[10] J. Pennington, R. Socher and C. D. Manning, "GloVe: Global Vectors for Word Representation," 2014.

[11] G. L. D. L. P. Sarracen, R. G. Pons, C. E. M. Cuza and P. Rosso, "Hate Speech Detection using Attention-based LSTM," *EVALITA,* pp. 235-238, 2018.

[12] S. A. Kokatnoor and B. Krishnan, "978-1-7281-8818-8/20/$31.00 ©2020 IEEE Twitter Hate Speech Detection using Stacked Weighted Ensemble (SWE) Model," *ResearchGate,* pp. 87-92, 2020.

[13] A. U. Lyer, "Toxic Tweets Dataset," Kaggle.com, 2021. [Online]. Available: https://www.kaggle.com/datasets/ashwiniyer176/toxic-tweets-dataset.

APPENDIX I

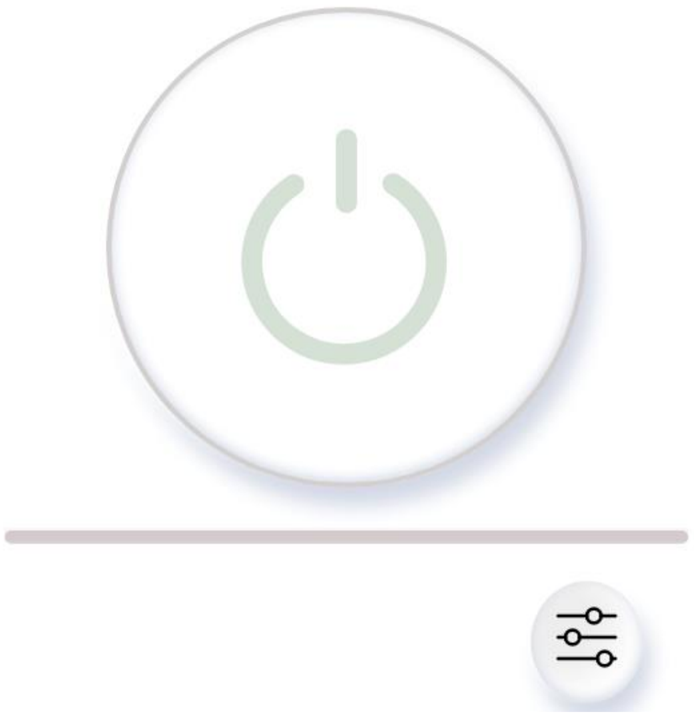
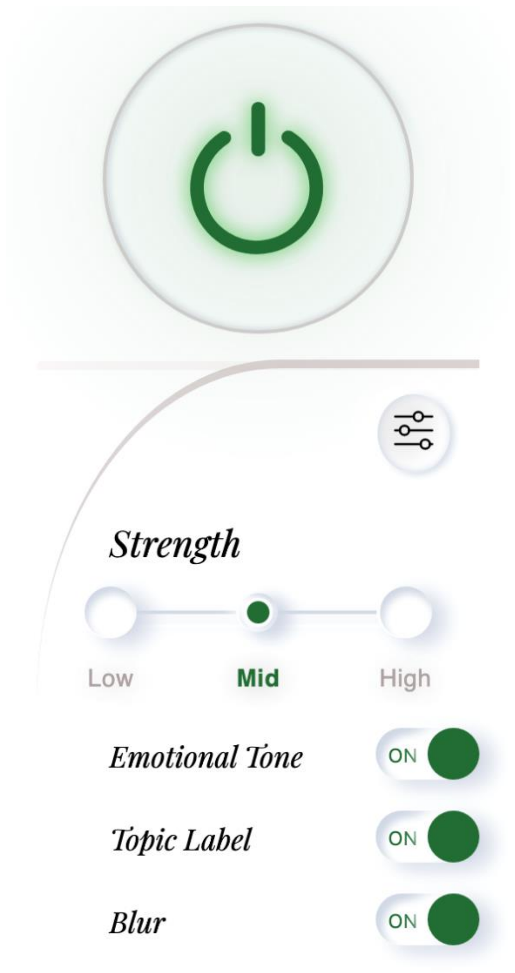


Fig. 34: Chrome Extension When Disabled



Strength

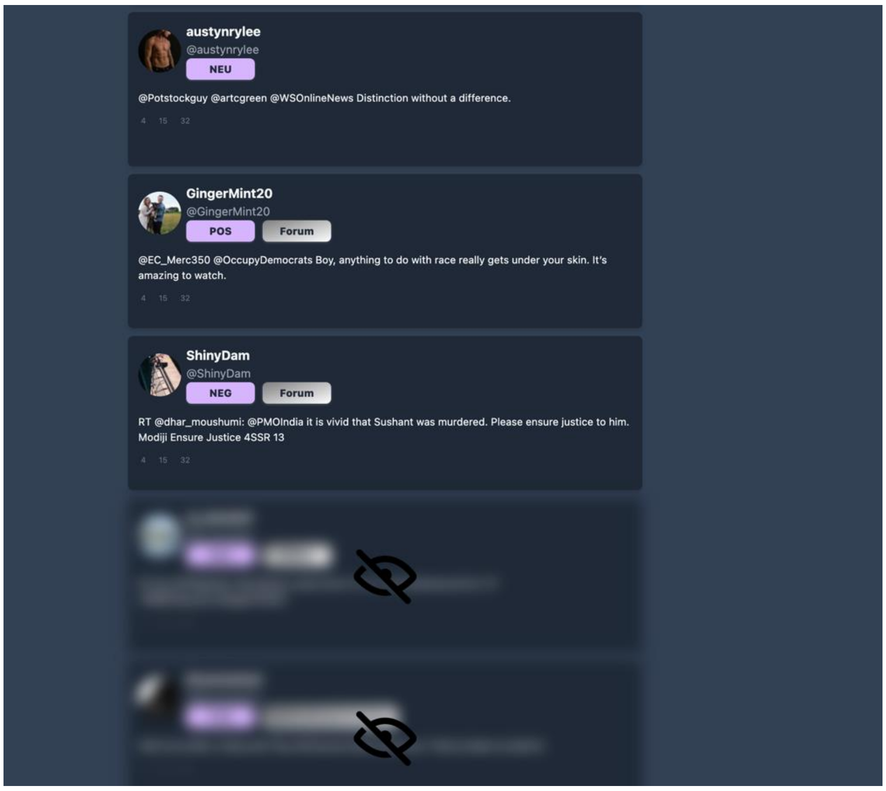
Low · Mid · High

Emotional Tone ON

Topic Label ON

Blur ON

Fig. 35: Chrome Extension When Activated



austynrylee
@austynrylee
NEU

@Potstockguy @artcgreen @WSOnlineNews Distinction without a difference.

4 15 32

GingerMint20
@GingerMint20
POS Forum

@EC_Merc350 @OccupyDemocrats Boy, anything to do with race really gets under your skin. It's amazing to watch.

4 15 32

ShinyDam
@ShinyDam
NEG Forum

RT @dhar_moushumi: @PMOIndia it is vivid that Sushant was murdered. Please ensure justice to him. Modiji Ensure Justice 4SSR 13

4 15 32

Fig. 36: Analysis on Twitter Live Sample Stream