Web Mining Project
Hate Speech Detection

Lin Yueyi[1961779], Lei Minghao[2035819], Momin Abdullah[1869371], Nguyen Anh-Nhat[2034311], and Wei Yiyi[1873027]

Team 1

1 Introduction

1.1 Background and Motivation

Hate speech and hate crimes have been pervasive societal issues throughout history. However, the advent of social media and other online communication platforms has exacerbated the problem, providing new avenues for the propagation of hateful ideologies and the potential radicalization of individuals. Addressing the issue of online hate speech has therefore become a pressing concern, as it has the potential to incite real-world harm, fuel societal divisions, and contribute to the normalization of discriminatory attitudes. Developing effective techniques for detecting and mitigating hate speech on social media and other online platforms is crucial for promoting a safer and more inclusive digital environment, as well as combating the insidious spread of hate-fueled ideologies.

1.2 Definition of Hate Speech and Offensive Language

Distinguishing between hate speech and offensive language can be challenging even for humans, given its subjective nature. We have found some fundamental concepts related to them, to enhance our understanding of the differences [8].

Hate speech Language that attacks or diminishes, that incites violence or hate against groups, based on specific characteristics such as physical appearance, religion, descent, national or ethnic origin, sexual orientation, gender identity or other" For example, the following sentence is a hate speech: All black are leeches on society and should be deported immediately. They contribute nothing and just breed crime and violence.

Offensive Language Language that is insulting, rude, or abusive, without targeting a protected group based on characteristics like race or religion. For example, the following sentence is an offensive language: You're a complete idiot and a waste of space. Your ideas are downright stupid.

1.3 Objectives

The primary objective of this project is to develop an effective hate speech detection method that can accurately identify and classify instances of hate speech, offensive language, and non-offensive content in textual data, such as social media posts or online comments.

2 Methodology

2.1 Data Preprocessing

There are no missing values or duplicate values in the dataset, so we don't need to process these issues. There are several essential text preprocessing steps, including text preprocessing and text tokenizing (See Table 1).

Table 1: Data Preprocessing: Operation And Reason

Operation	**Reason**
Lowercasing	Transforming all text to lowercase ensures uniformity and prevents the model from treating words with different cases as distinct entities.
URL Removal	Eliminating irrelevant information that may not contribute to the analysis.
Number Removal	Eradicating numerical digits from the text, reducing noise and focusing analysis on textual content.
Word Tokenization	Utilizing the NLTK library, the text is tokenized into individual words for futher training.
Handling Emojis	Converting them into text representations to get more information from emojiis.
Stopword Removal	Stop words, which are common words that often do not contribute significant meaning to the text, are removed from the tokenized text.
Lemmatization	Lemmatization reduces words to their base or dictionary form, ensuring that different inflected forms of a word are treated as the same token.

By incorporating these preprocessing steps, the tokenizer effectively prepares the text for analysis or model training by enhancing its consistency, reducing noise, and improving the representation of meaningful content.

2.2 Vectorization

For converting textual data into numerical representations suitable for machine learning models, we employed two popular techniques 'CountVectorizer' and

'TfidfVectorizer'. CountVectorizer transforms a collection of text documents into a matrix of token counts. It represents each document as a vector where each element corresponds to the count of a token in the document. TfidfVectorizer stands for the Term Frequency-Inverse Document Frequency Vectorizer. It transforms a collection of text documents into a matrix of TF-IDF features. TF-IDF reflects the importance of a word in a document relative to the entire corpus. We utilize both CountVectorizer and TfidfVectorizer to facilitate subsequent steps aimed at selecting the optimal performance.

2.3 Models

LogisticRegression Model Logistic regression, a statistical technique, employs the logistic model to predict the likelihood of an event by evaluating the log-odds [3]. This model expresses the log-odds as a linear combination of various independent variables. Logistic regression is a popular method for binary classification, used to predict the probability of an instance belonging to a specific class. Features are combined linearly with weights, and the logistic function transforms the result into the predicted probability. During training, the model adjusts these weights to minimize the difference between predicted probabilities and actual binary labels. Logistic regression offers simplicity, interpretability, and efficiency, particularly with large datasets.

Multinomial Naive Bayes Model Naive Bayes is a probabilistic algorithm family based on Bayes' Theorem [10]. It's "naive" because it presupposes feature independence, which means that the presence of one feature does not affect the presence of another (which may not be true in practice). The Multinomial Naive Bayes (MNB) model is a variant of the Naive Bayes algorithm, which is a probabilistic classification algorithm based on Bayes' Theorem [5] [7]. It is specifically designed for features that describe discrete frequency counts, such as the frequency of words in text data.

Bernoulli Naive Bayes Model Bernoulli Naive Bayes, a subset of the Naive Bayes Algorithm [9]. In contrast to the Multinomial Naive Bayes model, which works with term frequencies, the Bernoulli Naive Bayes model considers only the presence or absence of each feature (binary features) in the dataset. Bernoulli Naive Bayes specializes in classifying binary features like 'Yes' or 'No', '1' or '0', and 'True' or 'False', among others.

Random Forest Model Random Forests, also known as Random Decision Forests, is an ensemble learning technique designed for classification, regression, and various other tasks [2] [1]. It works by creating numerous decision trees during training. In classification tasks, the Random Forest output is determined by the most frequently selected class among the trees. Random Forest model

is known for its high accuracy, robustness, and ease of use, making it a popular choice across various machine learning applications, including classification, regression, and feature selection.

Ensemble Model Ensemble modeling involves creating multiple distinct models to forecast an outcome, achieved by employing various modeling algorithms or utilizing diverse training datasets [4]. These models are then combined to generate a final prediction for unseen data, aggregating the predictions from each base model. The primary aim of ensemble modeling is to mitigate the generalization error of predictions.

LSTM Model Long Short-Term Memory (LSTM) is a specialized type of recurrent neural network (RNN) designed to address the issue of vanishing gradients encountered in traditional RNNs [6]. Its notable advantage over other RNNs and sequence learning methods lies in its ability to maintain sensitivity to long-range dependencies, regardless of the gap length between relevant information. The fundamental structure of an LSTM unit comprises a cell, along with three gates: the input gate, output gate, and forget gate. These gates work in concert to manage the flow of information into and out of the cell. The cell's primary function is to retain information over extended periods, allowing for the storage of relevant values across thousands of time steps. The selective output mechanism enables the LSTM network to effectively maintain and utilize valuable long-term dependencies, thereby facilitating accurate predictions across current and future time steps.

3 Experimental setting

3.1 Data Structure

The dataset utilized for this project is the "Hate Speech and Offensive Language Dataset" obtained from Kaggle (https://kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset/data). This dataset contains a collection of tweets or social media messages, along with their corresponding labels indicating whether the text contains hate speech, offensive language, or neither. It comprises 24,783 entries, organized into 6 attributes as delineated in Table 2. The 'Class' attribute serves as a categorical indicator of speech type. A value of '0' denotes hate speech, '1' signifies offensive language, and '2' designates neither.

3.2 Data Visualization

We employ histograms as visual aids to portray several key aspects of our dataset in Figure 1. Firstly, they reveal the distribution of classes, showcasing that hate speech constitutes 5.77%, offensive language dominates at 77.43%, and the remaining 16.80% falls into neither category. Secondly, histograms shed light on

Table 2: Dataset structure

Attribute	Attribute Explanation
Count	Number of CrowdFlower users who coded each tweet (min is 3), sometimes more users coded a tweet when
Hate_Speech	Number of CF users who judged the tweet to be hate speech
Offensive_Language	Number of CF users who judged the tweet to be offensive
Neither	Number of CF users who judged the tweet to be neither offensive nor non-offensive
Class	Class label for majority of CF users
Tweet	The text content of the tweet or message collected

the distribution of word counts, with a notable concentration around 10, indicating a prevalent pattern in the length of the texts. Additionally, we highlight the top 10 most frequently occurring words, providing insights into the linguistic landscape of the dataset. Through these visualizations, we gain a comprehensive understanding of the dataset's composition and characteristics.

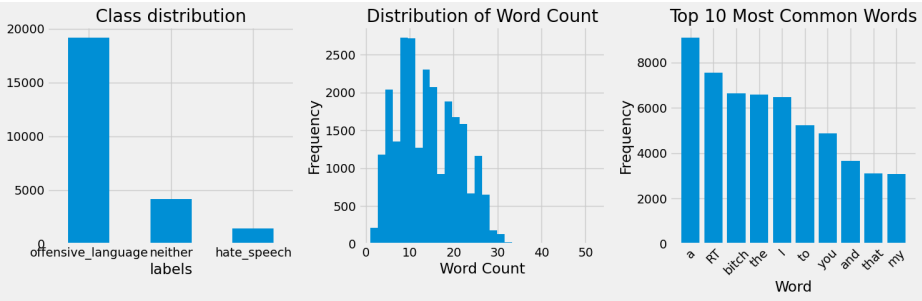

Fig. 1: Data Visualization

3.3 Model Training

Subsequently, we proceeded to fit the data into a diverse array of eight distinct models, each imbued with unique architectures and learning strategies, including Logistic Regression, MultinomialNB using CountVectorizer, Multinominal TF-IDF, BernouliNB with CountVectorizer, BernouliNB with TF-IDF, Random Forest, ENsemble Model, and LSTM Model with pre-trained Glove embedding. Through rigorous experimentation, we obtained a set of valid results encapsulating the performance of each model.

Dataset Splitting Before training our models, we divide the dataset into two subsets: the 'training set', and 'test set', maintaining an 8:2 proportion. To ensure the integrity of our models' performance evaluation.

Hyperparameter Tuning We utilize grid search to find the best hyperparameters in each of our training models. Grid search is a technique used for hyperparameter tuning, which involves selecting the optimal parameters for a model to maximize its performance. In grid search, a predefined set of hyperparameters is specified, and a grid of all possible combinations of these hyperparameters is created. The model is then trained and evaluated using each combination of hyperparameters, typically through cross-validation, and the combination that yields the best performance metric is selected as the optimal set of hyperparameters for the model.

Cross Validation We employ cross-validation to rigorously evaluate a model's performance and its ability to generalize. It involves partitioning the dataset into multiple subsets or folds. The model is then trained on the training set and evaluated on the validation set. This process is repeated multiple times, with each fold serving as the validation set exactly once. The performance metrics obtained from each fold are then averaged to provide an overall assessment of the model's performance.

3.4 Evaluation Measures

To assess the performance of different models, we employ three distinct methods of assessment.

Accuracy Accuracy measures the proportion of correctly classified instances out of the total instances. The formula for accuracy is:
$Accuracy = (TP + TN)/(TP + TN + FP + FN)$

Recall Recall measures the proportion of actual positive instances that are correctly predicted by the model. The formula for recall is:
$Recall = TP/(TP + FN)$

F1-score The F1-score is the harmonic mean of precision and recall. The F1-score is calculated as:
$F1 - score = 2 * (Precision * Recall)/(Precision + Recall)$
Where $Precision = TP/(TP + FP)$

4 Evaluation and Validation

In preparation for model fitting, we meticulously crafted an evaluation framework tailored to the intricacies of hate detection. This framework incorporated

essential performance metrics including accuracy, precision, recall, and F1-score. These metrics were diligently implemented to provide a comprehensive assessment of each model's efficacy in identifying hate speech.

The culmination of our evaluation efforts is presented in Table 3, wherein the performance metrics for each model for the training dataset and test dataset are meticulously tabulated. This tabular representation serves as a foundational reference point for elucidating the relative strengths and weaknesses of the models under scrutiny.

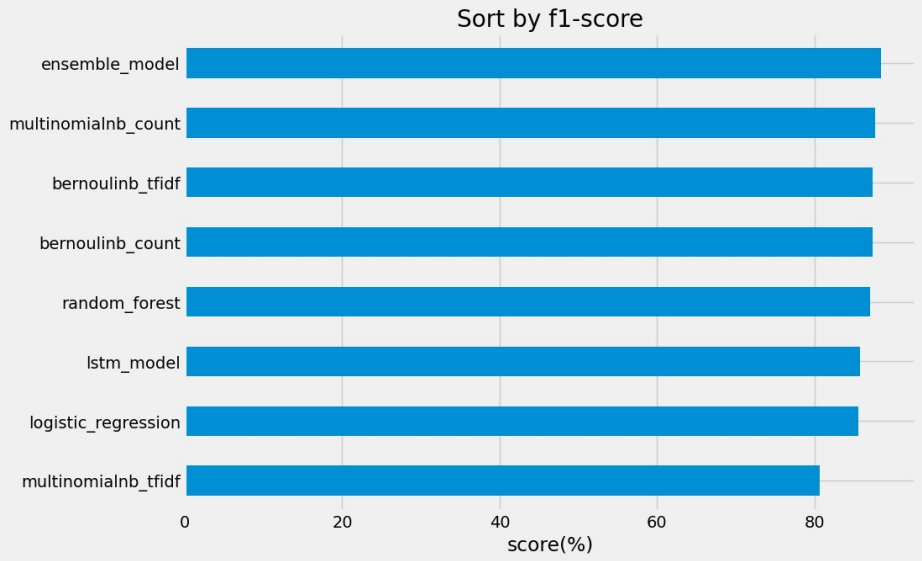

Fig. 2: Model Performance in F1 Score

For enhanced visual clarity, Figure 2 illustrates the percentage scores of each model across these four evaluation dimensions. Despite minor variations, it is evident that the performance metrics across the models are closely clustered. To discern finer distinctions, we relied on the F1-score as a pivotal metric for ranking model performance. Notably, the Ensemble Model emerged as the optimal choice from our experimentation, exhibiting the highest F1-score of 89.44% on the test dataset. Another significant finding is that when we train our ensemble model without employing stop-word elimination and lemmatization, we achieve an F1-score 2% higher than when utilizing these preprocessing techniques. This disparity may stem from the fact that these steps disrupt certain combinations of offensive words and hate speech, potentially diminishing the models' ability to discern specific patterns for tweet classification.

Table 3: Model Performance(score) on Training

Model	Dataset	Accuracy	Precision	Recall	F1-score
Logistic Regression	Train	92.79	92.28	92.79	91.8
	Test	90.22	88.86	90.22	88.94
MultinomialNB (CountVectorizer)	Train	91.37	91.03	91.37	91.16
	Test	89.17	88.35	89.17	88.68
MultinomialNB (TF-IDF)	Train	86.86	87.29	86.86	83.88
	Test	84.95	85.28	84.95	81.43
BernouliNB (CountVectorizer)	Train	91.08	90.48	91.08	90.62
	Test	89.49	88.18	89.49	88.61
BernouliNB (TF-IDF)	Train	91.08	90.48	91.08	90.62
	Test	89.49	88.18	89.49	88.61
Random Forest	Train	99.65	99.65	99.65	99.65
	Test	90.4	88.98	90.4	89.0
Ensemble Model	Train	95.05	94.86	95.05	94.73
	Test	90.46	89.26	90.46	89.44
LSTM Model	Train	-	-	-	-
	Test	86.75	85.28	86.75	85.82

4.1 Error Analysis

A crucial tool in assessing the performance of classification models is the confusion matrix, which succinctly summarizes the model's predictions against the ground truth across various classes. Figure 3 presents the confusion matrix of the ensemble model, elucidating its performance across three classes within our dataset.

To delve deeper into model weaknesses, we scrutinized the ensemble model's performance on the test dataset, specifically focusing on the top 20 instances of erroneous predictions with high probability. These instances represent cases where the model misclassified data despite exhibiting strong confidence, offering valuable insights into potential areas for improvement.

4.2 Comparsion with LLMs

Distil-BERT Model Refining the Distil-BERT model for hate speech and offensive language detection involves preprocessing the dataset for quality and uniformity. The pre-trained Distil-BERT model is initialized with extensive text data, fine-tuned using annotated hate speech datasets to capture linguistic nuances and contextual cues. Evaluation metrics like precision (89.91%), recall (90.32%), accuracy (90.32%), and F1-score (90.09%) comprehensively assess performance. Comparative analysis with the base BERT model offers insights into architectural differences. A customized evaluation function, incorporating

domain-specific factors, yields a holistic score. This approach ensures a rigorous methodology for hate speech and offensive language detection, leveraging LLM experimentation.

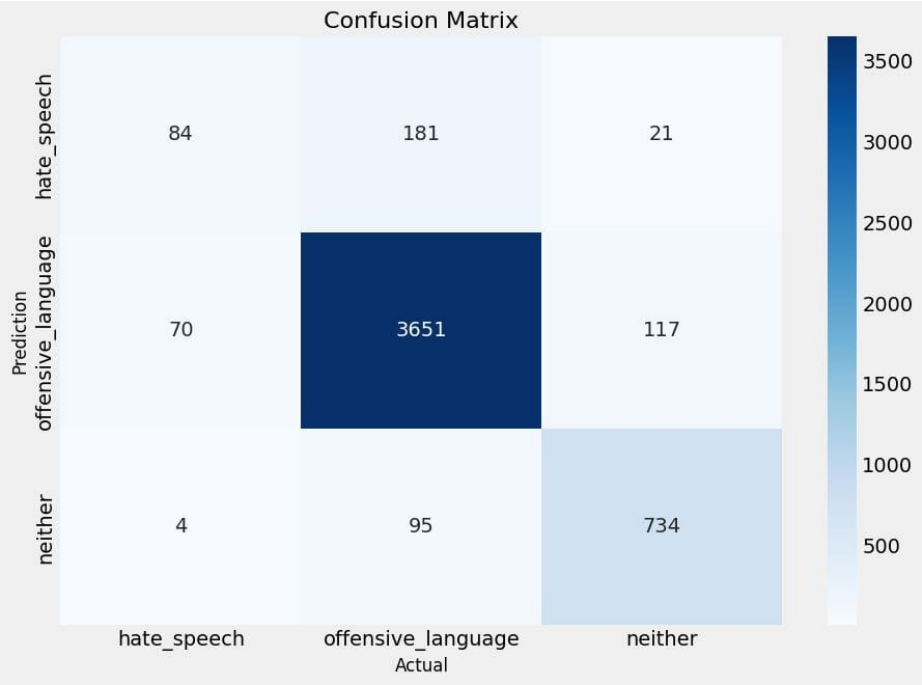

Fig. 3: Confusion Matrix of the ensemble model

Comparison between Benchmark In the current landscape of natural language processing, the dominant force is undeniably the rise of Large Language Models (LLMs), which serve as the epitome of the ongoing trend. Within this dynamic context, we embarked on a comparative analysis between two prominent LLMs: GPT-3.5-turbo-0125 (GPT) and llama-3-70b (llama). Our objective was to delve into their predictive capabilities by conducting an experiment centered on a meticulously curated dataset comprising 25 instances initially mispredicted by all our models. To guide the experiment, we utilized below carefully crafted prompts, aiming to unravel insights into the performance discrepancies between the two models.

Prompt: You are an expert linguist specializing in social media discourse analysis. Your expertise is in identifying and classifying comments on Twitter based on their content and intent. You have developed a nuanced understanding of the differences between hate speech, offensive language, and neutral expres-

sions. Your task is to help classify Twitter comments into one of the following categories:

- Hate Speech (label=0): Comments that involve hostility or prejudice against a particular group based on race, ethnicity, nationality, religion, gender, sexual orientation, disability, or similar grounds.
- Offensive Language (label=1): Comments that include profanity, vulgarity, or other language that may be considered disrespectful or rude, but do not necessarily target a specific protected group.
- Neither (label=2): Comments that do not contain hate speech or offensive language and are generally neutral or benign in nature. Return only the label (0, 1, or 2) without any explanation.

Nr.	tweet	true_label	pred_label	pred_prob	true_class	pred_class	gpt3.5-turbo-0125	llama3-70b-instruct	GPT True?	llama-3 True?
1	RT @obeyyreggie: stfu hoe RT “@tedddydgaf: "I'm proud to be African American" - Proud African American "I'm proud to be white" - Racist …	0	1	0.791	hate_speech	offensive_language	1	0	0	1
2	@jacksparenohoe let's hang out nigger	1	0	0.591	offensive_language	hate_speech	0	0	0	0
...	I openly admit to being the level of white trash that will drive across town to the gas station with free hot dogs & half price drinks.	2	1	0.402	neither	offensive_language	1	2	0	1
24	Hitler didn't finish it. Can u. If a nigger ur Jew confronts u in the street what then.	0	1	0.616	hate_speech	offensive_language	0	0	1	1
25	RT @StonerBoii2cold: “@TreVaughnLG: Moma said no pussy cats inside my dog house” that's what got bro nem locked inside the dog pound !!	2	1	0.925	neither	offensive_language	1	1	0	0
								Result	8	10

Fig. 4: Results from Benchmarks

Within this sample set in Figure 4, llama exhibited superior predictive accuracy by furnishing correct prognostications for 10 instances, whereas GPT evinced accuracy in only 8 cases. Hence, llama's discernment proved marginally more adept, securing accurate predictions for 2 additional cases compared to its GPT counterpart. Nonetheless, a comprehensive elucidation of the comparative strengths and limitations of these models necessitates a more exhaustive analysis.

5 Conclusion

Our extensive analysis of hate speech and offensive language detection, utilizing eight meticulously tuned models, marks a significant milestone in combating online toxicity. Rigorous evaluation revealed our ensemble model as the top

performer, boasting the highest F1-score among all tested methods. This underscores the effectiveness of ensemble techniques in combining model strengths, enhancing predictive accuracy and robustness.

5.1 Challenge

In our investigation, we encountered a notable challenge pertaining to overfitting, notably observed in the LSTM Model. This phenomenon was particularly conspicuous as evidenced by the discernible rise in test loss subsequent to the sixth epoch.(Shown in Figure 5)

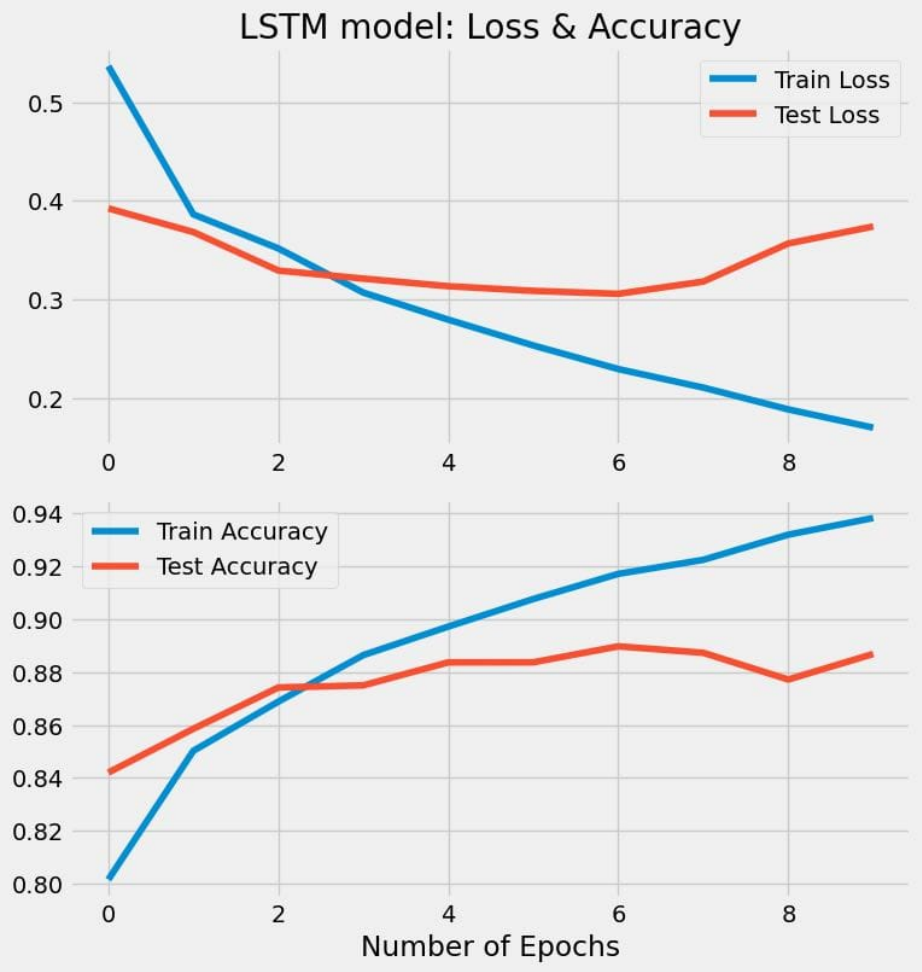

Fig. 5: Overfitting in LSTM Model

In our exploration of hate speech and offensive language detection, we encountered a formidable obstacle: the striking resemblance between offensive and hateful content (Figure 6). This similarity presented a significant challenge, rendering it difficult to discern between the two categories accurately. The subtle differentiations between offensive language and outright hate speech blurred boundaries, complicating the task of identification and classification. Despite this hurdle, our meticulous approach, harnessing eight finely-tuned models, propelled us towards a resolution. Through thorough evaluation, we unveiled the power of ensemble learning, leading to the identification of the model boasting the highest F1-score. Furthermore, our comparison with the LLM framework yielded valuable insights, underscoring the intricacies of the task at hand while validating the progress made in advancing hate speech detection methodologies.

Furthermore, in our quest for benchmarking, we juxtaposed our ensemble model against LLM, a widely recognized baseline in natural language processing tasks. Our findings illuminate the advancements made, showcasing not only competitive performance but also highlighting areas where our model excels. This comparative analysis not only solidifies our model's standing but also contributes to the broader discourse surrounding hate speech detection methodologies.

In conclusion, our study underscores the efficacy of ensemble learning in hate speech and offensive language detection, offering a potent tool in combating online toxicity. Moving forward, continued research and refinement of these techniques hold promise in fostering safer and more inclusive digital environments.

	tweet	true_label	pred_label	pred_prob	true_class	pred_class
189	Lmao let these hoes be hoes ain't no Savin nem	0	1	0.991680	hate_speech	offensive_language
2747	RT @_iHATEMOON: All these bitches & niggaz...	0	1	0.989366	hate_speech	offensive_language
1509	Black bitches don't be kickin up in our school...	0	1	0.988873	hate_speech	offensive_language
2634	@bonnoxxx haha bitch ima draw a webb in bullet...	0	1	0.988282	hate_speech	offensive_language
4880	He ain't shit girl, 💯he a bitch made n...	0	1	0.987566	hate_speech	offensive_language
327	RT "@_ThatGAPeach: & alla my niggas hot bo...	0	1	0.986541	hate_speech	offensive_language
4177	RT @dirtyimage: @Tronkitty not just cause of h...	0	1	0.985885	hate_speech	offensive_language
3407	RT @JHazeThaGod: You other niggas a call up a ...	0	1	0.985340	hate_speech	offensive_language
4590	It's so shady when you bitches talk to guys w/...	0	1	0.985332	hate_speech	offensive_language

Fig. 6: Sample for Similarity between offensive and hate

References

1. Gérard Biau. Analysis of a random forests model. *The Journal of Machine Learning Research*, 13(1):1063–1095, 2012.
2. Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.
3. Scott A Czepiel. Maximum likelihood estimation of logistic regression models: theory and implementation. *Available at czep. net/stat/mlelr. pdf*, 83, 2002.
4. Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.
5. Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29:103–130, 1997.
6. Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.
7. Ashraf M Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. In *AI 2004: Advances in Artificial Intelligence: 17th Australian Joint Conference on Artificial Intelligence, Cairns, Australia, December 4-6, 2004. Proceedings 17*, pages 488–499. Springer, 2005.
8. Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, and Ophir Frieder. Hate speech detection: Challenges and solutions. *PloS one*, 14(8):e0221152, 2019.
9. Gurinder Singh, Bhawna Kumar, Loveleen Gaur, and Akriti Tyagi. Comparison between multinomial and bernoulli naïve bayes for text classification. In *2019 International conference on automation, computational and technology management (ICACTM)*, pages 593–596. IEEE, 2019.
10. Harry Zhang. The optimality of naive bayes. *Aa*, 1(2):3, 2004.