A Comparative study of identifying hate and offensive comments on social media platforms using text classification models and NLP techniques

Eranki Vasistha

Abstract

In today's internet-dependent world, social media platforms like Twitter, Instagram, and Facebook have interconnected the globe in a truly fascinating way, enabling instant communication between individuals across geographical boundaries, whether in the United States, Japan, India, or even the most remote locations. However, this increased connectivity has also led to a surge in online cyberbully and the proliferation of inappropriate racial, sexist, and abusive slurs on these platforms. All these hateful comments are posted on social media in text form. Through this project, I aim to develop an automated identifier that detects such hate comments, which would help us in enabling the blocking of offending users or deletion of harmful content, or both

1 Brief overview of tasks performed

This project focuses on detecting hate speech on social media using machine learning models. The aim is to evaluate and compare the performance of various algorithms in classifying text into three categories: "Hate Comment," "Offensive Comment," and "Neither." Key tasks included feature selection, data preprocessing, and training multiple models for text classification. Error analysis and result evaluation were conducted for each model, followed by hyperparameter tuning to enhance performance. Finally, the project discusses potential improvements and future directions for this work.

2 Data Description

The dataset consists of 24,783 labeled text records, with each record containing a tweet and its corresponding label indicating one of three categories: "Hate Comment" (the minority class), "Offensive Comment," or "Neither." The data is split into a training set comprising 17,348 records (70 % of the dataset) and a test set of 7,435 records (30%), providing a balanced setup for training and evaluation.

The dataset distribution, visualized in the pie charts in figure -1, highlights the proportion of each label category in both the training and testing datasets. Overall, the dataset consists of 19,190 offensive comments, 1,430 hate comments, and 4,163 "neither" comments. This imbalance is also reflected in the training dataset, where offensive comments dominate with 13,319 instances (76.8%), followed by 2,926 "neither" comments (16.9%) and 1,103 hate comments (6.4%). Similarly, in the testing dataset, offensive comments constitute 79% of the records, while "neither" and hate comments make up 16.6% and 4.4%, respectively. The charts clearly illustrate the skewed distribution toward offensive comments, which will influence model training and evaluation.

3 Related Work

3.1 Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection

This paper provided critical insights into preprocessing text from social media platforms, focusing on unique aspects such as slangs, hashtags, and other platform-specific text formats. A major takeaway was the decision to adopt the same three-class labeling scheme used in the study, categorizing comments as *Hate Comment*, *Offensive Comment*, and *Neither*. This approach ensured a structured framework for annotating and organizing the dataset.

3.2 Effective Hate-Speech Detection in Twitter Data Using Recurrent Neural Networks

The methodologies in this paper offered guidance on dataset transformation techniques, including the

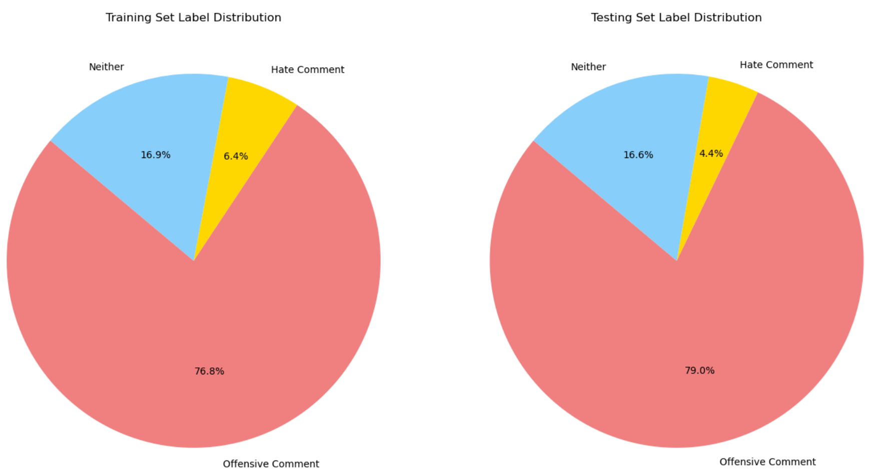

Figure 1: Label wise dataset distribution of the training and testing dataset

classification of labels and removal of usernames and special characters. It also explored the use of recurrent neural networks (RNNs) for hate speech detection. Inspired by this, future plans for the project include a comparative analysis between RNNs and Naïve Bayes models. This comparison aims to identify the strengths and weaknesses of each model and understand their relative performances in hate speech detection.

3.3 Thumbs Up? Sentiment Classification Using Machine Learning Techniques

This study highlighted the effectiveness of unigram features in text classification tasks. Unlike bigram, trigram, or n-gram features, unigrams proved to be simpler yet more impactful. The key lesson from this paper was that simplicity often leads to superior outcomes in machine learning, a principle that has been integrated into the project's approach to feature representation.

3.4 A Comparative Study on Feature Selection in Text Categorization

This paper emphasized the importance of feature selection in text classification, introducing methods such as information gain and term frequency-based approaches. Among these, information gain stood out as an effective measure for determining the relevance of features. By assigning higher values to significant features, this technique plays a vital role in improving classification accuracy and reducing noise in the dataset.

3.5 Comparing the Performance of Different NLP Toolkits in Formal and Social Media Text

The findings from this paper underscored the advantages of using NLP toolkits specifically designed for processing social media text. It encouraged the idea of comparing the results of traditional machine learning models with NLP toolkit-based approaches. This comparative analysis aligns with the project's goal of evaluating diverse methodologies to identify the most effective strategy for detecting hate speech on platforms like Twitter.

4 Data Preprocessing and Transformation

The dataset underwent several preprocessing steps to prepare it for modeling. These included the removal of URLs, special characters (e.g., @, !, &), and hashtags. While hashtags could potentially serve as valuable features if deconstructed into individual words, they were excluded to evaluate the performance of the Naïve Bayes model under minimal preprocessing. This decision allowed for an assessment of the model's effectiveness without relying on advanced preprocessing techniques. Additionally, user mentions and usernames were removed, as they were considered less relevant for the task of text classification. This approach aimed to establish a baseline performance for the Naïve Bayes model while minimizing unnecessary complexity. The resulting preprocessed data was saved in a new column named **cleaned_tweets** within the dataset's CSV file, which was subsequently used for training the model.

Class	Tweet	New Label	Processed Tweets
2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. &amp; as a man you should always take the trash out...	Neither	as a woman you shouldn t complain about cleaning up your house amp as a man you should always take the trash out
1	!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!	Offensive Comment	boy dats cold tyga dwn bad for cuffin dat hoe in the st place
1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit	Offensive Comment	dawg you ever fuck a bitch and she start to cry you be confused as shit

Table 1: A snapshot of the dataset: The first two columns depict dataset before pre-processing and the two show dataset after transformation and pre-processing.

Lemmatization was avoided during preprocessing because it turned out to be too aggressive for this dataset. During initial tests, it was found that lemmatization changed the structure and meaning of the tweets too much, often turning them into text that was hard to understand or interpret. This made the tweets lose their original context and important details that are useful for text classification.

Instead, a simpler and less intrusive preprocessing(like the ones implemented and mentioned before) approach was used to keep the tweets as close to their original form as possible. This helped preserve the natural structure and context of the data, allowing the model to better capture patterns and relationships within the tweets. The goal was to evaluate how well the Naïve Bayes model could perform without relying on heavy preprocessing steps and to keep the process straightforward while still working with the kind of informal text commonly found on social media. Table-1 shows the dataset before and after the pre-processing and transformation techniques.

5 Feature Selection Methods

Feature selection is a critical step in machine learning, particularly for text classification tasks where the feature space is often large. Below are the main feature selection methods:

1. Univariate Selection (e.g., k-Best Features): Selects the best features based on statistical tests like chi-square, mutual information, or analysis of variance (ANOVA).This is Suitable for ranking features individually without considering feature interactions.

2. Information Gain (Mutual Information): Measures the reduction in entropy or uncertainty about the target variable when the feature is known. Effective for identifying features that maximize the predictive capability of the model.

3. Low Variance: Low variance feature selection is a method used to eliminate features with little to no variability across samples in the dataset. Features with low variance contribute minimally to distinguishing between classes because their values remain nearly constant and do not provide meaningful information for classification.

4. L1 Regularization (LASSO): A sparsity-inducing method that assigns zero weights to irrelevant features during training. This Works well for reducing high-dimensional datasets and improving interpretability.

5.1 Reasons for choosing k-best features using information gain

In the context of the current task, univariate selection(k-best features) using information gain (IG) emerges as the most suitable feature selection method due to the following reasons:

1. Text Classification Suitability: In text classification tasks, IG efficiently evaluates the relevance of unigram features by calculating the reduction in uncertainty about the target variable. This aligns well with the Naïve Bayes assumption of feature independence, allowing for a straightforward feature evaluation process.

2. Reduction of High-Dimensional Feature

Model	Feature Space	Metric		
		Precision	**Recall**	**F1**
Baseline (Naïve Bayes)	15344	0.73	0.54	0.57
Low variance (threshold = 0.001)	1180	0.71	0.66	0.60
Low variance (threshold = 0.005)	324	0.67	0.60	0.63
k-best using mutual information with $k = 1000$	1,000	0.72	0.62	0.62
k-best using mutual information with $k = 2000$	**2,000**	**0.73**	**0.63**	**0.65**

Table 2: Performance of Naïve Bayes model with different feature selection methods.

Space: The unigram bag-of-words representation resulted in 15,344 features, creating computational challenges. Information gain effectively reduces this high-dimensional space to the top 2,000 features with the highest discriminative power, improving model efficiency and performance.

3. Proven Effectiveness: IG has been demonstrated in literature (e.g., [5] and [6]) as a robust feature selection method for Naïve Bayes classifiers, outperforming alternative methods like bigram/trigram representations or other feature selection techniques.

4. Improved Model Interpretability: By selecting features that maximize IG, the resulting feature space consists of the most informative unigrams. This enhances interpretability, as each selected feature has a clear, measurable contribution to the model's predictions.

5. Performance Gains: Models trained on IG-selected features are typically faster to train and achieve better generalization on the test set. This is due to the removal of noisy or irrelevant features that could degrade model performance.

6. Flexibility for Text Data: IG is highly adaptable to text data, where many features (words) may have little or no correlation with the target labels. IG ensures that only the most informative words are retained.

While IG's computational cost (15–20 minutes for this dataset) is a notable drawback, this is an acceptable trade-off given the significant performance benefits and the dimensionality reduction achieved. Thus, IG is the most practical and effective method for feature selection in this project.

In addition to the reasons mentioned above, the Table-2 presents the performance of the Naïve Bayes model using various feature selection methods. It highlights the accuracy and F1-score achieved by each method, providing a clear comparison of their effectiveness.

6 Using traditional Machine learning Models

6.1 Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm based on Bayes' Theorem. It assumes conditional independence between features, meaning the occurrence of one feature is independent of another given the class label. This assumption, though simplistic, often works well for text classification tasks like sentiment analysis and spam detection in general.

In context to the work at hand, Naive Bayes is particularly suitable because:

- It performs efficiently on unigram Bag-of-Words (BoW) features, which align well with the assumption of feature independence.

- It is computationally inexpensive, making it ideal for large datasets like yours.

- The method is robust when dealing with imbalanced classes, especially with proper Laplace smoothing applied.

Naive Bayes provides a solid baseline for comparing more advanced classifiers, as its probabilistic nature makes it interpretable and effective for initial experimentation.

6.2 Logistic Regression

Logistic Regression is a linear model used for binary or multi-class classification. It predicts the probability of a class by modeling the data using a sigmoid function, enabling it to handle complex relationships between features.

In context to the work at hand, Logistic Regression offers: - The ability to model relationships between features and class labels, which is useful when the conditional independence assumption of Naive Bayes does not hold.

- Flexibility in handling sparse datasets like text data represented by BoW or TF-IDF vectors.

- Good interpretability, as feature coefficients

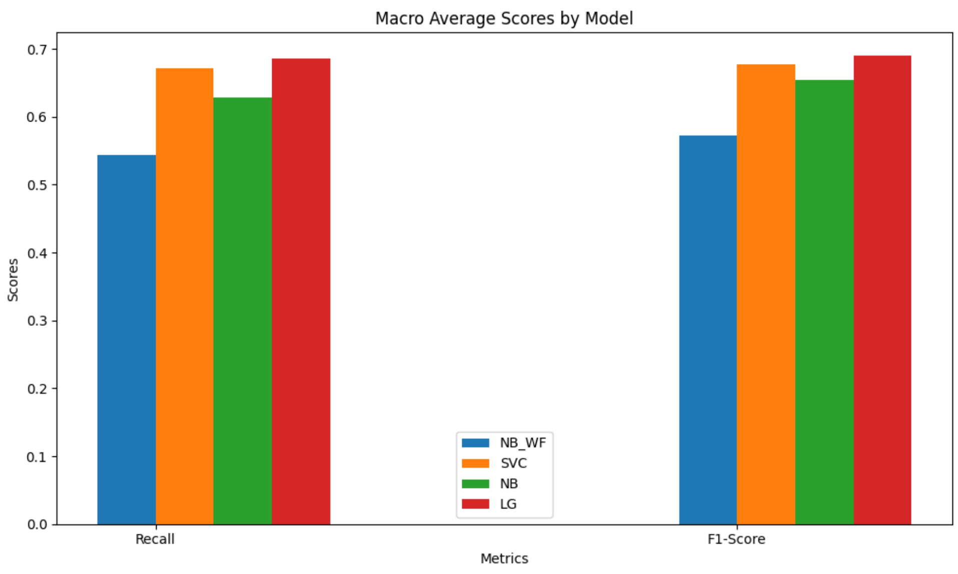

Figure 2: Metric comparison of the various models trained over the dataset

indicate the impact of each feature on the classification.

Logistic Regression is well-suited for tasks involving feature selection, as it can leverage the reduced feature set (e.g., the top 2,000 features with the highest information gain) to improve efficiency without sacrificing accuracy.

6.3 Support Vector Machines (SVM)

SVM is a powerful classifier that constructs a hyperplane or set of hyperplanes to separate data points into classes. It is particularly effective for high-dimensional spaces and datasets where the number of features exceeds the number of samples.

In context to the work at hand, SVM is advantageous because: - It handles text classification effectively by maximizing the margin between classes, making it robust to imbalanced datasets like yours. - It supports different kernel functions (e.g., linear, RBF) to model complex relationships between features. - It is less prone to overfitting in high-dimensional spaces, which is common in text-based datasets.

SVM may require more computational resources than Naive Bayes or Logistic Regression but often yields higher performance, particularly when combined with effective feature selection like mutual information.

7 Using BERT and neural network

7.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a powerful pre-trained language model that has shown superior performance in many natural language processing tasks, including hate speech detection. BERT's strength lies in its ability to capture contextual information and semantic nuances in text, which is crucial for understanding the subtle nature of hate speech. By using a large corpus of text data during pre-training, BERT develops a deep understanding of language structure and meaning, allowing it to be fine-tuned for specific tasks like hate speech detection with remarkable effectiveness.

7.2 Neural Networks

Neural networks, particularly deep learning models such as Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Convolutional Neural Networks (CNN), have demonstrated significant improvements in hate speech detection compared to traditional machine learning methods. These models excel at processing sequential data and capturing complex patterns in text. BiLSTM, for instance, has shown superior performance by analyzing text in both forward and backward directions, allowing it to better understand context and improve sensitivity to semantic nuances in tweets. CNN models, on the other hand, are effective at extracting local features from text, which can be

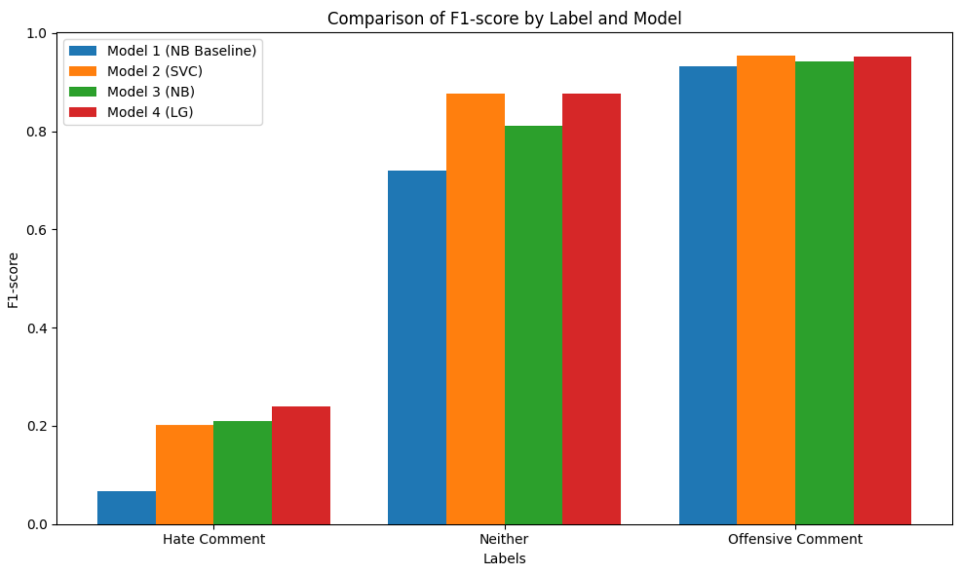

Figure 3: label wise f1 score comparison of the various supervised machine learning models trained over the dataset

particularly useful for identifying hate speech patterns.

Combining BERT with neural networks yields better results in hate speech detection compared to traditional methods. This approach leverages BERT's pre-trained knowledge and the neural network's ability to learn task-specific features. Studies have shown that BERT-based models can achieve F1-scores of up to 96% on hate speech detection tasks, outperforming traditional machine learning approaches. The integration of BERT with neural networks allows for more nuanced understanding of context, sarcasm, and implicit hate speech, leading to more accurate and reliable detection.

8 Error and Result Analysis

To evaluate the performance of different models on the classification task, we conducted experiments using Naive Bayes, Logistic Regression, SVM, and a Neural Network utilizing BERT embeddings. Each model was tested using the top 2,000 features selected via information gain. Below is an analysis of the results, highlighting strengths and areas for improvement.

8.1 Naive Bayes

The Naive Bayes model achieved an overall accuracy of 90.05%, performing particularly well for the "Offensive Comment" class with a recall of 97% and an F1-score of 94%, demonstrating its effectiveness in identifying the majority class. However, it struggled significantly with the minority "Hate Comment" class, achieving a precision of 42% and a recall of only 14%, indicating a high false-negative rate where many hate comments were misclassified as either "Neither" or "Offensive Comment." The "Neither" class showed decent performance with an F1-score of 81%. These limitations can be attributed to the conditional independence assumption of Naive Bayes, which restricts its ability to capture nuanced interactions between features, particularly in minority classes.

8.2 Logistic Regression

Logistic Regression achieved an overall accuracy of 91.59%, showing improved performance over Naive Bayes for all classes. For the majority class ("Offensive Comment"), it maintained a high recall of 95% while slightly improving the precision of the "Neither" class to 93%. However, the "Hate Comment" class saw only a marginal improvement in recall (17%) and continued to suffer from a low F1-score of 24%, reflecting persistent challenges in distinguishing this minority class. While Logistic Regression better captures feature interactions, its performance is still limited by the imbalance in the dataset, as the minority class remains underrepresented.

8.3 Support Vector Machine (SVM)

SVM achieved the highest accuracy among traditional models, with an overall accuracy of 91.81% and robust performance across the majority and

Table 3: Label-wise Precision and F1-Score for Machine Learning Models

Model	**Hate Comment**		**Neither**		**Offensive Comment**	
	Precision	F1-Score	Precision	F1-Score	Precision	F1-Score
Naive Bayes (2000 Best Features)	0.42	0.21	0.85	0.81	0.92	0.94
Logistic Regression (2000 Best Features)	0.39	0.24	0.83	0.88	0.95	0.95
SVM (2000 Best Features)	0.40	0.20	0.84	0.88	0.95	0.95
Neural Network (BERT, Epoch 5)	0.41	0.37	0.88	0.88	0.95	0.95
Naive Bayes (Bigram)	0.67	0.02	0.93	0.51	0.84	0.91
Naive Bayes (Trigram)	0.40	0.01	0.93	0.42	0.83	0.90
SVM (Trigram)	0.47	0.33	0.83	0.87	0.95	0.95

"Neither" classes. For the "Hate Comment" class, the results were similar to Logistic Regression, with a recall of 13% and an F1-score of 20%. SVM's margin-based approach enabled it to effectively separate the "Offensive Comment" class, which dominated the dataset, achieving a precision of 95% and recall of 96%. However, despite its high overall accuracy, SVM struggles with imbalanced data, overemphasizing the majority class and sacrificing precision and recall for the minority class, "Hate Comment."

8.4 Neural Network with BERT Embeddings

The neural network with BERT embeddings achieved an overall accuracy of 91.69%, performing on par with the traditional models. It delivered strong results for the majority class ("Offensive Comment"), with a precision of 95% and recall of 96%. Notably, it showed noticeable improvement in identifying the minority "Hate Comment" class, achieving a precision of 41% and a recall of 33%. This highlights the strength of BERT's contextual embeddings in capturing semantic nuances that traditional BoW-based models miss. The "Neither" class also showed competitive results, with an F1-score of 88%. However, while BERT embeddings significantly improve the ability to identify minority classes, they still fall short of the ideal, particularly for "Hate Comment," likely due to the inherent difficulty of classifying rare instances and the subtle distinctions between "Hate Comment" and "Offensive Comment."

This analysis highlights the trade-offs between model complexity, feature representation, and dataset characteristics in achieving optimal performance for the task at hand.

9 Conclusion: Overall Insights and Challenges

Imbalance in Dataset: The small representation of the "Hate Comment" class created significant challenges for all models. Despite achieving high performance for majority classes, all models suffered from poor recall for "Hate Comment," indicating a high rate of false negatives. This imbalance suggests the need for targeted solutions such as data augmentation, oversampling, or class weighting to mitigate the skewed distribution and improve the model's ability to recognize minority classes.

Majority Class Dominance: The dominance of the majority "Offensive Comment" class led to inflated overall accuracy and F1-scores, as models favored the majority class at the expense of the minority ones. This behavior was most pronounced in simpler models like Naive Bayes, which misclassified many minority class instances into the majority class. Even advanced models like Neural Networks and SVM showed signs of this issue, although they performed better at separating minority classes than simpler models.

Feature Representation: The use of the top 2,000 features selected through information gain proved to be an effective method for improving classification performance. By focusing on the most discriminative features, models were able to reduce noise and better distinguish between classes. However, feature representation methods such as bigrams and trigrams, though computationally expensive, were less effective than expected for capturing contextual nuances. Contextual embeddings, such as those provided by BERT, significantly enhanced the Neural Network's ability to capture semantic relationships and improved minority class predictions.

Model Comparison: SVM and Neural Networks were the best-performing models overall, with SVM achieving the highest accuracy due to its effective margin-based separation. Neural Networks, especially when combined with BERT embeddings, demonstrated superior handling of the "Hate Comment" class by leveraging contextual information. However, Neural Networks required more computational resources and hyperparameter tuning, which could pose a challenge in resource-constrained scenarios. Logistic Regression provided a balance of simplicity and performance, while Naive Bayes, although less accurate, demonstrated rapid training and inference capabilities suitable for baseline evaluations.

10 Future Work and Improvement strategies

1. Class Balancing: Addressing the class imbalance is critical to improving the performance on underrepresented classes, such as "Hate Comment." Techniques such as oversampling the minority class (e.g., SMOTE or ADASYN), undersampling the majority class, or creating synthetic data can help balance the training dataset. Additionally, experimenting with class weighting in loss functions can penalize misclassifications of minority classes more heavily, forcing the model to focus on these instances. These methods should be evaluated to determine which yields the best balance between precision and recall for minority classes.

2. Advanced Embeddings: The use of pretrained language models like RoBERTa, GPT, or XLNet can further enhance the contextual understanding of text data. These models, which are trained on vast amounts of text data, can capture nuanced relationships between words, phrases, and context. Fine-tuning these models on the specific dataset could help differentiate between similar classes, such as "Hate Comment" and "Offensive Comment." Moreover, exploring embeddings tailored for specific domains (e.g., hate speech detection) could further boost performance on the task.

3. Hybrid Models: Combining the strengths of traditional machine learning models with neural approaches can offer complementary advantages. For example, integrating SVM with embeddings from BERT or RoBERTa can leverage the robust feature separation of SVM alongside the semantic richness of contextual embeddings. Hybrid models could also include ensemble methods that combine predictions from multiple architectures, such as a blend of logistic regression, SVM, and neural networks, to reduce individual model biases and improve overall classification robustness.

References

[1] Sidra Gul, Muhammad Salman Khan, Asima Bibi, Amith Khandakar, Mohamed Arselene Ayari, Muhammad E.H. Chowdhury,Deep learning techniques for liver and liver tumor segmentation: A review,Computers in Biology and Medicine,Volume 147,2022,105620,ISSN 0010-4825

[2] Pitsilis, G.K., Ramampiaro, H. & Langseth, H. Effective hate-speech detection in Twitter data using recurrent neural networks. Appl Intell 48, 4730–4742 (2018). W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation, 19(90):297–301.

[3] G. Koushik, K. Rajeswari and S. K. Muthusamy, "Automated Hate Speech Detection on Twitter," 2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA), Pune, India, 2019, pp. 1-4, doi: 10.1109/ICCUBEA47591.2019.9128428. keywords: Bag of Words;TFIDF;Hate Speech

[4] Alexandre Pinto, Hugo Gonçalo Oliveira, and Ana Oliveira Alves. Comparing the Performance of Different NLP Toolkits in Formal and Social Media Text. In 5th Symposium on Languages, Applications and Technologies (SLATE'16). Open Access Series in Informatics (OASIcs), Volume 51, pp. 3:1-3:16, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2016)

[5] Panneerselvam, Kalaiselvan. "Thumbs up? Sentiment Classification using Machine Learning Techniques."

[6] Yang, Yiming, and Jan O. Pedersen. "A comparative study on feature selection in text categorization." icml. Vol. 97. No. 412-420. 1997. Mary Harper. 2014. Learning from 26 languages: Pro- gram management and science in the babel program. In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. Dublin City University and Association for Computational Linguistics, page 1. http://aclweb.org/anthology/C14-1001.