Vol. 6 No. 3, 2025, pp. 207-219 **Jurnal Pepadun**

Detection of Hate Speech in TikTok Comment Sections Using the Naïve Bayes Algorithm with Smoothing Implementation

[*1]Roy Rafles Matorang Pasaribu, [2]Didik Kurniawan, [3]Muhaqiqin, and [4]Akmal Junaidi

[1,2,3,4]Department of Computer Science, Faculty of Mathematics and Natural Sciences, University of Lampung,
Jl. Soemantri Brojonegoro No.1 Gedung Meneng, Bandar Lampung, Indonesia
e-mail: ***[1]royraflmp@gmail.com**, [2]didikunila@gmail.com, [3]muhaqiqin@fmipa.unila.ac.id,
[4]akmal.junaidi@fmipa.unila.ac.id

Abstract *–Hate speech is a biased, antagonistic, and discriminatory expression that commonly appears on social media platforms, including TikTok. The high volume of comments and varied language styles make manual detection challenging. This research proposes a hate speech detection model using the Multinomial Naïve Bayes algorithm with smoothing to address zero-probability issues and enhance prediction performance. The dataset is split into 80% training and 20% testing portions. The model achieves an accuracy of 88.41%, with precision, recall, and F1-score showing balanced performance. A user evaluation involving 35 participants and 7,415 TikTok comments records a detection accuracy of 68.6%. The model is further implemented into a Google Chrome extension capable of real-time hate speech detection, displaying prediction probabilities and allowing user validation. This study aims to support healthier digital interactions by improving automated hate speech detection on social media.*

Keywords**:** *Hate Speech Detection; Naïve Bayes; Smoothing; Machine Learning; Plugin.*

1. INTRODUCTION

Hate speech is defined as a biased, hostile, and malicious expression directed at an individual or group based on inherent characteristics [1]. Such expressions convey discriminatory, intimidating, antagonistic, or prejudiced attitudes toward traits such as gender, race, religion, ethnicity, skin color, national origin, disability, or sexual orientation. The primary aim of hate speech is to harm, discredit, harass, intimidate, insult, and victimize its targets while also fostering insensitivity and violence against them [2]. Detecting hate speech is crucial, particularly as traditional rule-based methods fall short in managing the massive volume of user-generated content on social media and lack the flexibility to adapt to evolving language styles. In contrast, machine learning approaches have demonstrated promising results in automating hate speech detection and analyzing sentiments within text data [3].

In the era of globalization, information technology has become a powerful medium for rapid data transmission and communication. One rapidly growing digital platform is TikTok, which offers video-based content with a unique technical structure and extremely high user adoption [4]. Its features of imitation and remixing continue to accelerate diverse user interactions [5]. With more than 100 million downloads and a user rating of 4.4 on the Play Store, TikTok ranks fourth globally in user population, as stated by Julia Chan, Mobile Insights Analyst [6]. The platform's comment section enables open expression however; the increasing volume of interactions has also led to a significant presence of intentional or unintentional hate speech. Hate speech contradicts linguistic politeness as an indicator of communicative intelligence and ethics, and the prevalence of insults, defamation, blasphemy, provocation, and hoaxes on social media including TikTok reflects the misuse of expressive freedom by users who often comment without considering the consequences, reinforced by the natural human tendency toward hatred [7].

Various classification methods have been applied in detecting hate speech, including Support Vector Machine (SVM), Deep Learning (DL), and Naïve Bayes (NB). While SVM performs well with non-linear problems, it is susceptible to overfitting, and DL can recognize complex text patterns but requires very large datasets. Naïve

 | doi: 10.23960/pepadun.v6i3.268

Bayes offers a simpler and computationally efficient approach that performs well on both small and large datasets and is less prone to overfitting. However, Naïve Bayes often encounters zero-frequency issues when certain words do not appear in the training data, resulting in zero probabilities that degrade classification performance. Smoothing techniques are therefore applied to mitigate this problem and improve predictive accuracy [8].

Although previous studies have applied Naïve Bayes with smoothing for text classification, few have specifically integrated this optimized model with a real-time application capable of detecting hate speech directly within the TikTok comment section. The novelty of this research lies in combining the Multinomial Naïve Bayes algorithm enhanced with smoothing for improved accuracy and its implementation into a Google Chrome extension that performs real-time detection, displays prediction probabilities, and incorporates user validation feedback. This enhancement is particularly important because Laplacian smoothing commonly known as add-one smoothing, in which each variable in every parameter is increased by one [9] helps prevent zero probability issues and thereby increases model robustness. This integration addresses both technical and practical gaps by offering a lightweight, accurate, and user-interactive system for moderating hate speech on one of the world's most active social media platforms. Therefore, this study aims to optimize Naïve Bayes using smoothing techniques and apply it in a real-time environment to support healthier and more responsible digital communication.

2. RESEARCH METHODOLOGY

Although modern classification methods such as Support Vector Machine (SVM) and Deep Learning (DL) have been widely applied in hate speech detection research, the selection of Naïve Bayes (NB) in this study is based on methodological considerations and the characteristics of the dataset. SVM is known for producing high accuracy and operating efficiently on complex non-linear classification problems. However, similar to other machine learning algorithms, SVM is prone to overfitting, particularly when parameter tuning is not performed properly [10].

In the context of this research where comment data are highly dynamic, unstructured, and not always balanced this susceptibility to overfitting becomes a significant limitation. Deep Learning offers substantial advantages due to its ability to automatically learn features from raw data and extract complex patterns through its multiple hidden layers. Numerous studies demonstrate that DL achieves superior accuracy in hate speech detection and sentiment analysis tasks [6]. Nevertheless, DL models require a very large amount of data to perform effectively and to avoid overfitting [11]. Given the limited dataset used in this study, DL is not an ideal choice and may result in an unstable or unreliable model. In contrast, Naïve Bayes offers strong compatibility with both small and large datasets, and consistently performs well even in complex classification tasks [12]. NB estimates its parameters using the entire training dataset, which helps reduce the overfitting issues seen in SVM [13].

Although NB has a known limitation related to zero probability where words that do not appear in the training data may result in a zero likelihood during prediction this issue can be effectively addressed using smoothing techniques [14]. Smoothing has been shown to significantly improve NB performance by preventing zero-frequency problems and generating more robust probability estimates [15]. A study by research [8] demonstrated that NB combined with smoothing achieved a high accuracy of 95.9% in classifying eligibility for social assistance, highlighting its effectiveness in real-world mixed-data scenarios. Based on these findings, this study adopts NB with smoothing because it aligns well with the nature of TikTok comment data, which is diverse, moderate in size, and requires a model that is stable, fast, and resistant to overfitting.

Therefore, the choice of Naïve Bayes does not disregard the strengths of modern algorithms. Instead, it is based on matching the method to the dataset characteristics, the need for model stability, computational efficiency, and ease of implementation. Naïve Bayes with smoothing is expected to yield high accuracy and provide meaningful contributions to improving hate speech detection systems on social media platforms. The research workflow, which utilizes the Naïve Bayes algorithm with smoothing and its implementation in a plugin, can be seen in Figure 1.

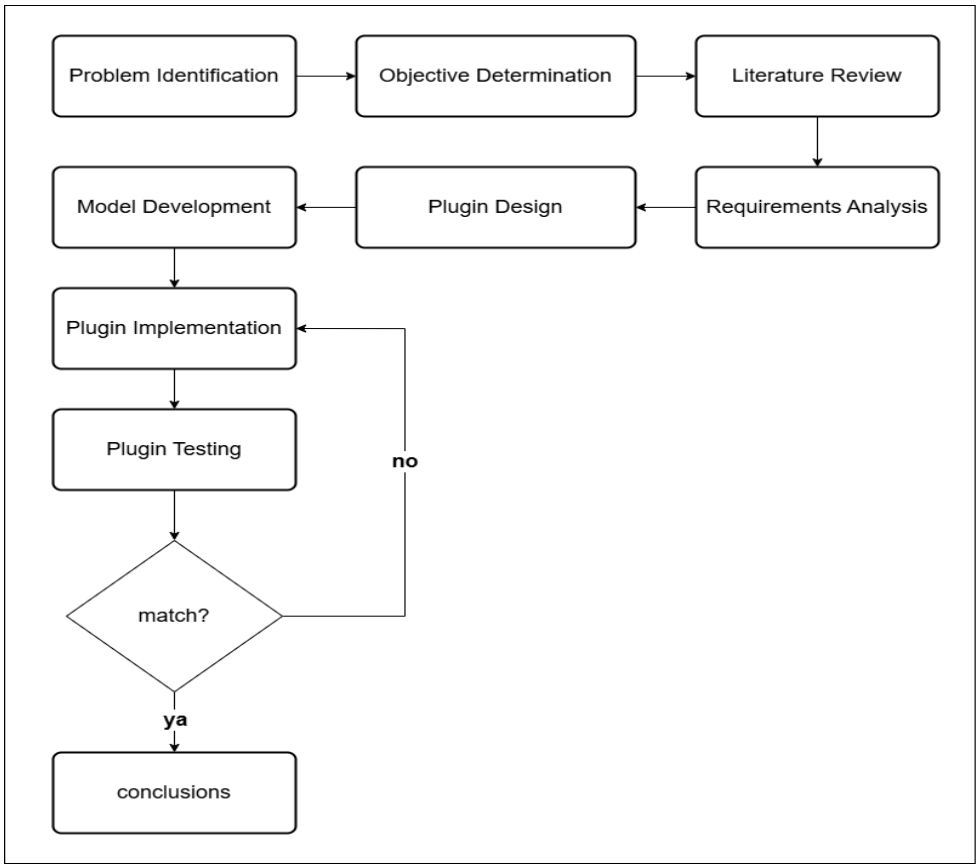

Figure 1. Research workflow.

2.1. Problem Identification

In the digital era, hate speech on social media is increasingly prevalent, and traditional rule-based methods are ineffective in handling the large and diverse volume of data. Therefore, this study aims to develop a machine learning model based on the Naïve Bayes algorithm to automatically detect hate speech in TikTok comment sections.

2.2. Objective Determination

The main objective of this study is to apply the Naïve Bayes algorithm with smoothing to improve the accuracy of hate speech detection in TikTok comments. Naïve Bayes is selected because it offers a strong balance between computational efficiency and classification performance, making it particularly suitable for environments with limited resources such as browser extensions. Unlike modern deep learning models that require heavy computation and large memory usage, Naïve Bayes provides fast inference with minimal overhead, ensuring that predictions can be generated instantly without affecting browser performance. This efficiency is crucial for real-time processing, as the plugin must analyze incoming comments continuously while maintaining a smooth user experience. Additionally, the study aims to develop a browser-based plugin capable of identifying hate speech in real time and providing clear visualizations to users. The lightweight nature of Naïve Bayes enables the plugin to run directly in the browser while still delivering accurate classifications, supporting the overall goal of creating a practical, responsive, and user-friendly hate speech detection tool for TikTok's comment section.

2.3. Literature Review

The literature review was conducted by examining various previous studies related to hate speech detection, the machine learning methods used, and the implementation of technology in the form of browser extensions. The reviewed literature includes the use of statistical models, natural language processing (NLP) techniques, as well as lexicon-based and supervised learning approaches.

 | doi: 10.23960/pepadun.v6i3.268

2.4. Requirements Analysis

Requirements analysis is conducted to determine the specifications needed for plugin development. These requirements include functional aspects, such as automatic hate speech detection with visual effects on comments, as well as non-functional aspects such as compatibility with the Google Chrome browser and ease of use for general users.

2.5. Plugin Design

At this stage, system design is carried out, covering the plugin architecture and its working mechanism. The plugin consists of several main components:

a. Content Script: Responsible for retrieving comments from the TikTok page in real time.
b. Background Script: Acts as a bridge between the content script and the hate speech detection model.
c. Popup UI: A user interface that allows sensitivity adjustment for detection and displays the results of comment analysis.
d. Model Machine Learning: Implementation of Naïve Bayes with smoothing to detect hate speech based on the processed data.

The detection mechanism is carried out in three stages: (1) extracting comments from the TikTok page, (2) analyzing them using a machine learning model, and (3) visualizing the detection results by highlighting comments identified as hate speech.

2.6. Model Development

Model development begins with the collection of TikTok comment datasets using web scraping techniques. The collected dataset then undergoes preprocessing stages [16], which include:

2.6.1. Cleaning: Removing punctuation, numbers, and irrelevant special characters.
2.6.2. Case Folding: Converting all text to lowercase for analysis consistency.
2.6.3. Stopwords Removal: Removing common words that do not carry significant meaning.
2.6.4. Normalization: Converting non-standard words into their standard forms.
2.6.5. Tokenizing: Breaking the text into word units for further analysis.

After preprocessing, data labeling is performed using a lexicon-based approach with the Indonesian Sentiment (InSet) Lexicon, which has been modified to detect hate speech. Features are then extracted using the TF-IDF (Term Frequency-Inverse Document Frequency) method to determine the weight of each word in the model analysis. Then, this research implements the Naïve Bayes algorithm with Laplace smoothing to improve prediction accuracy in detecting hate speech, as it is a simple probabilistic classifier that calculates a set of probabilities based on the frequency and combinations of values in the dataset [17].

The model is trained using the preprocessed dataset and validated using test data. Finally, model evaluation is conducted through two approaches: first, the model is evaluated using a confusion matrix to compute accuracy, precision, recall, and F1-score; and second, a user evaluation is carried out by directly involving users to assess the model's performance in detecting hate speech. The final stage in model development is evaluation. This research includes two types of evaluation: model evaluation and user evaluation. For model evaluation, a confusion matrix is used to calculate accuracy, precision, recall, and F1-score. Meanwhile, user evaluation involves directly engaging users to assess the model’s ability to detect hate speech.

2.6.6. Evaluation

Evaluation is carried out in two main aspects, including model evaluation and user evaluation.

a. Machine Learning Model Evaluation

To evaluate the performance of an algorithm, a confusion matrix is used, which includes 4 terms to represent classification results: true positive (TP), false positive (FP), true negative (TN), and false negative (FN).

 | doi: 10.23960/pepadun.v6i3.268

Table 1. Confusion matrix

Predicted / Actual	Positive	Negative
Positive	TP (True Positive)	FN (False Negative)
Negative	FP (False Positive)	TN (True Negative)

The evaluation is conducted by measuring the best performance based on accuracy, precision, recall, and F1-score [18]. These performance metrics can be calculated using the following equations.

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (1)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (2)$$

$$Recall = \frac{TP}{(TP+FN)} \quad (3)$$

$$f1 - score = 2 \times \frac{(recall \times precision)}{(precision+recal)} \quad (4)$$

b. User Evaluation

In addition to these metrics, a model verification stage involving users is also included. This stage enables direct user involvement, where users interact with the plugin by using like and dislike buttons to indicate their agreement or disagreement with the classification results produced by the model. This participatory approach transforms users into active evaluators, allowing real-world validation of the model's predictions. The collected feedback serves as valuable data that can be used to retrain and refine the model, thereby improving its accuracy and adaptability to evolving language patterns, slang, and context-specific expressions commonly found in TikTok comments. Furthermore, this process provides deeper insights into the model's strengths such as identifying clear-cut hate speech and its weaknesses, particularly in dealing with ambiguous, sarcastic, or context-dependent language. By incorporating user validation into the evaluation framework, the system becomes more robust, user-centered, and capable of continuous improvement over time.

2.7. Plugin Implementation

After the model is developed, it is integrated with a Google Chrome-based plugin. The plugin is built using a combination of JavaScript, HTML, and CSS for the interface, and utilizes FastAPI as the backend to process comments with the machine learning model.

2.8 Plugin Testing

Testing is conducted to ensure that the plugin functions correctly in detecting hate speech in TikTok comment sections. This testing uses two main approaches: black box testing and white box testing:

2.8.1. Black Box Testing

Black box testing is a testing method that focuses on functionality, examining how the software responds to user-provided inputs to produce the desired outputs, without considering the internal processes or underlying code [19]. This testing approach emphasizes the external functionality of the plugin without taking into account how the internal processes work. The testing is conducted by enabling and disabling the hate speech detection plugin and observing how it identifies comments containing hate speech.

 | doi: 10.23960/pepadun.v6i3.268

2.8.2. White Box Testing

White box testing is a software testing technique that focuses on the internal structure of the application, including its logic, code structure, and program control flow [20]. This testing evaluates the internal working logic of the plugin by testing each core function that forms the hate speech detection system. The testing is carried out by thoroughly inspecting the source code. In white box testing, there are three main techniques, namely:

i. Statement Coverage

Statement Coverage is a testing technique that measures the percentage of code lines (statements) that have been executed at least once during testing. The main goal is to ensure that every line of code in the program has been tested.

$$Statement\ Coverage = \left(\frac{Number\ of\ statements\ executed}{Total\ statements\ in\ the\ code}\right) \times\ 100\% \quad (5)$$

ii. Branch Coverage

Branch Coverage (also known as Decision Coverage) measures the percentage of logical branches (true/false conditions) that have been executed during testing. It ensures that each decision point such as those in if, else, or switch statements has been tested in both true and false conditions.

$$Branch\ \ Coverage = \left(\frac{Number\ of\ branches\ executed}{Total\ logical\ branches}\right) \times\ 100\% \quad (6)$$

iii. Path Coverage

Path Coverage measures the percentage of all possible execution paths taken through a program during testing. It is a more comprehensive method because it ensures that every combination of logical paths is tested.

$$Path\ Coverage = \left(\frac{Number\ of\ paths\ tested}{Total\ possible\ execution\ paths}\right) \times\ 100\% \quad (7)$$

3. RESULT AND DISCUSSION

The model development workflow in this research can be seen in Figure 2.

3.1. Dataset Collection

The dataset used for model development consists of comments from TikTok content, totaling 17,710 comments. All comments were thoroughly processed for use as training data. Additionally, the dataset was split using an 80:20 ratio, where 80% of the total dataset was used for training, and the remaining 20% was used for testing. This step aimed to evaluate the ability of the hate speech detection plugin on Google Chrome for TikTok in recognizing and handling comments that the model had not previously encountered. Comment data was retrieved through the TikTok API by sending requests to the endpoint https://www.tiktok.com/api/comment/list/, utilizing the video ID along with parameters such as count and cursor for pagination. The received data, in JSON format, was processed to extract both comments and their replies. A recursive function was implemented to ensure that the entire conversation thread was captured. Once the data was collected, it was converted into a DataFrame and saved in CSV format for further analysis.

3.2. Preprocessing Dataset

Preprocessing was conducted to improve the quality and variability of the dataset, aiming to enhance the prediction results and the overall performance of the developed model. The preprocessing steps applied in this study include cleaning, case folding, stopword removal, normalization, and tokenizing.

© 2025 The Authors | doi: 10.23960/pepadun.v6i3.268

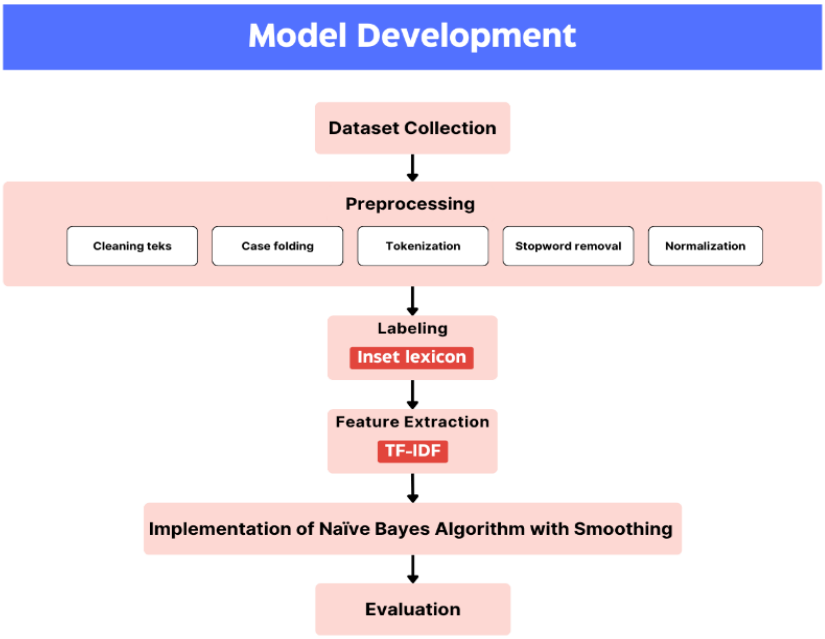

Figure 2. Model development workflow.

3.3. Labeling

Labeling in this study used the Indonesian Sentiment Lexicon (INSET) from INSET GitHub (https://github.com/fajri91/InSet) [21], which was modified for hate speech detection. INSET is divided into two categories: a positive lexicon containing words with positive values, and a negative lexicon containing words with negative values. This modification aims to improve labeling accuracy for model development. Each word in the analyzed text is compared with the lexicon to determine its score. The polarity score is determined based on the values found in the positive lexicon (+1, +2, etc.) and the negative lexicon (−1, −2, etc.).

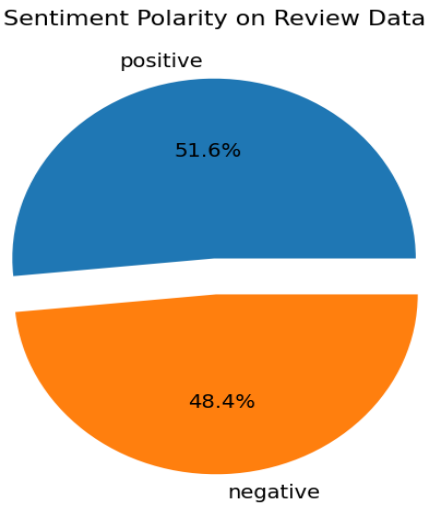

Figure 3. Percentage of positive and negative comments from labeling results.

Out of a total of 17,710 comments analyzed, after the preprocessing stage, 17,300 comments remained. These were then labeled using the InSet Lexicon method. The results showed that 8,921 comments were categorized as positive, while 8,379 comments were classified as negative. This indicates a relatively balanced sentiment distribution between positive and negative comments in the dataset used.

 | doi: 10.23960/pepadun.v6i3.268

3.4. Feature Extraction

The feature extraction process in this research employed the Term Frequency-Inverse Document Frequency (TF-IDF) method, which measures the weight of words in a text. TF calculates the frequency of a word within a document, while IDF measures the importance of a word across the entire corpus. The combination of these two metrics results in a TF-IDF value that reflects the weight of a word. The final TF-IDF score is obtained by multiplying the TF and IDF values, which helps balance the term frequency in a document with its rarity across the corpus [22]. In the implementation, TfidfVectorizer from scikit-learn was used with the parameters max_features=8000, min_df=2, max_df=0.8, and ngram_range=(1, 2). Both unigram and bigram features were extracted to capture more complex word contexts. The resulting TF-IDF matrix was then used as features for training the model.

3.5. Implementation of Naïve Bayes Algorithm with Smoothing

Before training the model, a label distribution analysis was conducted on the training data. The results showed that the training data did not suffer from significant class imbalance, with label 1 (positive) comprising 51.57% and label 0 (negative) comprising 48.42% of the data. Since the difference in class proportions is relatively small (below 10%), this study did not apply any imbalance handling techniques such as undersampling or oversampling. Therefore, the training data was used as-is in the model training process.

This research utilizes the Multinomial Naïve Bayes (MNB) model for hate speech classification, with a TF-IDF Vectorizer configured with max_features=8000, min_df=2, max_df=0.8, and ngram_range=(1, 2) to capture relevant unigrams and bigrams. MNB relies on Bayes' Theorem and implicitly incorporates the Markov assumption, where the probability of a word or bigram depends on the preceding context. To address the issue of unseen words in the training data, Laplace Smoothing is applied with a smoothing parameter of alpha=1. The dataset is split with 80% for training and 20% for testing, ensuring objective model evaluation and good generalization capability.

3.6. Evaluation

3.6.1. Model Evaluation

This research focuses on detecting hate speech in TikTok comment sections using the Naïve Bayes algorithm with smoothing. To evaluate the model's performance, tests were conducted on two variants of the Naïve Bayes algorithm: one with smoothing and one without. In addition to using evaluation metrics such as accuracy, precision, recall, and F1-score, the test results were also analyzed using a confusion matrix to illustrate the distribution of the model's predictions.

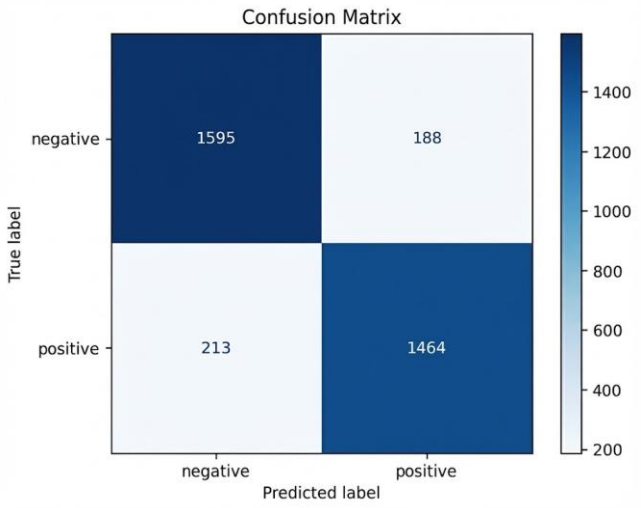

Figure 4. Confusion matrix results.

 | doi: 10.23960/pepadun.v6i3.268

Based on the test results, the confusion matrix shows that the model correctly identified 1,464 hate speech comments as hate speech (True Positive/TP) and correctly identified 1,595 non-hate speech comments as non-hate speech (True Negative/TN). However, there were 188 non-hate speech comments that were incorrectly classified as hate speech (False Positive/FP), and 213 hate speech comments that were incorrectly classified as non-hate speech (False Negative/FN). The classification model was also evaluated using several key metrics such as accuracy, precision, recall, and F1-score. This study further compares the performance metrics of the Naïve Bayes model with and without smoothing.

Table 2. Model evaluation.

	Naïve Bayes + Smoothing	**Naïve Bayes**
accuracy_test	0.884	0.860
precision_test	0.884	0.861
recall_test	0.884	0.860
f1_test	0.884	0.861

Based on Table 2, it can be observed that the Naïve Bayes model with smoothing outperforms the model without smoothing across all evaluation metrics. This improvement is attributed to the use of smoothing, which addresses the issue of zero probabilities for words that do not appear in certain classes. Consequently, smoothing helps enhance the model's prediction accuracy and contributes to more stable performance.

3.6.2. User Evaluation

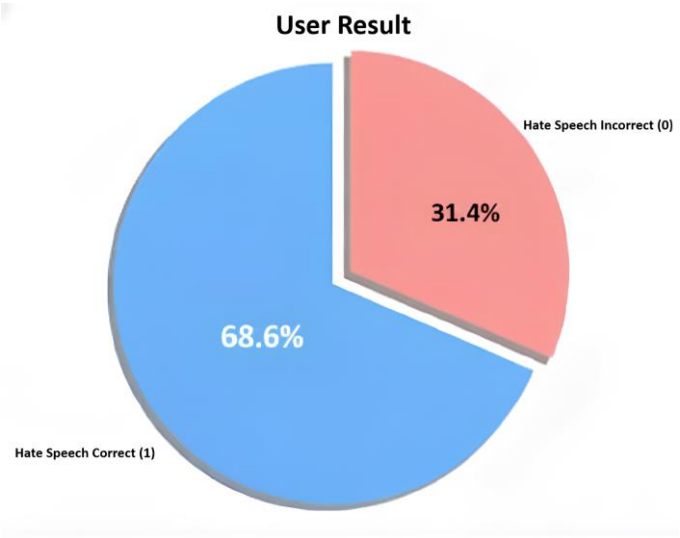

Figure 5. User evaluation percentage.

This research involved user testing with 35 participants who analyzed 7,415 comments from 36 TikTok videos to evaluate the accuracy of the hate speech detection model. The user testing results showed an accuracy of 68.6%, with 5,084 comments correctly identified as hate speech, while 2,331 comments were incorrectly classified. Challenges in detection arose due to the complexity and diversity of TikTok comments, including the use of slang and dynamic context. Additionally, the model relied on the INSET lexicon, which may no longer be fully relevant due to the rapid evolution of language on the platform, highlighting the need for updates to improve detection accuracy.

3.7. Plugin Implementation

Plugins and extensions are software modules designed to add or modify the functionality of a main program without altering its source code [23]. This plugin was developed using JavaScript and the FastAPI framework,

© 2025 The Authors | doi: 10.23960/pepadun.v6i3.268

with two main endpoints: /scrapping and /validasi. The /scrapping endpoint retrieves comments from TikTok, performs preprocessing, and classifies them using the Naïve Bayes model. The /validasi endpoint allows users to provide feedback via like or dislike buttons to validate the prediction results, which are then used for model retraining. The plugin is directly integrated into TikTok through a content script that monitors URL and content_id changes. For visualization, comments are color-coded based on the probability of hate speech. This plugin uses Manifest V3 to ensure efficiency and responsiveness in monitoring changes on TikTok pages.

Figure 6. Plugin implementation.

Figure 6 shows the user interface (UI) of a Chrome extension developed for this research, titled "Hate Speech Detection," which is designed to analyze comments on TikTok. The extension uses a Naive Bayes classification model to determine whether a comment contains hate speech. At the top, there is a toggle button that allows users to activate or deactivate the extension. When the toggle is turned off, the extension is inactive, and a message appears stating "Extension Not Active." The middle section displays a prediction probability legend that indicates the model's confidence level in classifying a comment as hate speech. This is because the Naive Bayes algorithm predicts the class of a comment based on the probability derived from the input features. Gray represents a very low probability (0.50–0.60), yellow indicates low (0.60–0.70), orange indicates medium (0.70–0.80), red represents high (0.80–0.90), and dark red indicates a very high probability (0.90–1.00). These colors are used to visually highlight comments on the TikTok page, allowing users to quickly identify those that are likely to contain hate speech.

Additionally, the interface includes an explanation of the validation buttons. The orange check mark is used when users confirm that a comment is indeed hate speech, while the blue cross mark is used when users indicate that a comment is not hate speech, despite being flagged by the system. This validation process is essential for collecting user feedback to help improve the model's accuracy. At the bottom, the phrase "Powered by Naive Bayes" signifies that the detection algorithm is based on the Naive Bayes model, which works by calculating the likelihood of each class based on the input data.

3.9. Plugin Testing

The testing in this research was conducted using both black box testing and white box testing methods.

3.9.1. Black Box Testing

Black box testing is a software testing method that focuses on evaluating system functionality without any knowledge of the internal code or program structure. In this approach, testers only examine the relationship

 | doi: 10.23960/pepadun.v6i3.268

between the given inputs and the outputs produced based on predefined test scenarios [24]. Test cases are developed to include a description of the scenario, the input data, the expected output, and the actual result in order to determine whether the system status is "pass" or "fail" [25]. In this study, six testing scenarios were conducted on the developed plugin, and all scenarios received a "pass" status, indicating that the plugin operates according to the specified requirements and intended objectives.

3.9.2. White Box Testing

White box testing, also referred to as clear box testing, glass box testing, or structural testing, is a software testing method that examines the internal logic and control flow of a program to ensure that all possible execution paths are exercised at least once [26]. In this research, white box testing was applied to two main functions, namely checkUrlChange and startScraping, using statement coverage, branch coverage, and path coverage techniques, which are commonly used to measure the thoroughness of internal code testing [27]. Testing of the checkUrlChange function demonstrated that all lines of code were executed and all logical branches operated correctly, including detecting URL changes and triggering the scraping process under specified conditions. Meanwhile, white box testing of the startScraping function showed that every execution path was covered, both when the contentId was successfully detected and when it was not, ensuring that the scraping function was called only when valid input was present and that an appropriate error message was generated otherwise. Based on these results, it can be concluded that both functions have been thoroughly tested and operate in accordance with the intended design and system requirements.

3.10. Discussion

Recent studies have addressed hate speech detection on social media. [18] used TF-IDF and SVM for TikTok comments and achieved strong results, with 96.21% accuracy and an F1-score of 95.50%. [28] compared several classifiers and found that MLP with SMOTE obtained the highest accuracy, while Multinomial Naïve Bayes without SMOTE produced the best recall (93.2%), showing that model selection and data balancing affect performance. Naïve Bayes has consistently shown advantages in related work. [29] achieved 93% accuracy on Twitter, and Prasetyo et al. (2024) improved Naïve Bayes performance using smoothing, reaching 95.9% accuracy. These findings demonstrate that Naïve Bayes is effective for text classification, particularly because it is simple, computationally efficient, performs well on high-dimensional data such as TF-IDF, and remains robust even with limited training data.

In this study, a smoothed Naïve Bayes model achieved 88% accuracy on TikTok comments. Although this result is lower than SVM-based research, Naïve Bayes is still preferable in this context due to its speed, low resource requirements, and suitability for real-time implementation. The main contribution of this research is a Google Chrome plugin that performs live hate speech detection directly in TikTok comment sections, providing practical value beyond accuracy alone.

4. CONCLUSIONS

This study successfully developed a hate speech detection model for TikTok comment sections using the Multinomial Naïve Bayes algorithm with smoothing. The model achieved an accuracy of 88.41%, with precision, recall, and F1-score values of 88.41% using TF-IDF features with n-grams (1,2). Furthermore, evaluation through user testing involving 35 participants and 7,415 TikTok comments resulted in a detection accuracy of 68.6%, indicating the model's practical applicability in real usage scenarios. The novelty of this research is demonstrated through the integration of an optimized Naïve Bayes model with smoothing into a real-time Google Chrome extension capable of detecting hate speech directly within the TikTok comment section. This implementation provides probability-based visualization and incorporates user validation, offering a lightweight yet effective solution that bridges the gap between machine learning techniques and real-time online moderation tools an area that has received limited attention in prior studies. Future work may focus on expanding the dataset, employing more adaptive labeling strategies beyond the INSET lexicon, and exploring advanced or hybrid machine learning approaches, while maintaining the computational efficiency required for real-time deployment in browser-based environments.

 | doi: 10.23960/pepadun.v6i3.268

LITERATURE

[1] F. Poletto, V. Basile, M. Sanguinetti, C. Bosco, and V. Patti, "Resources and Benchmark Corpora for Hate Speech Detection: A Systematic Review, *Language Resources and Evaluation*, vol. 55, no. 2, pp. 477-523, 2021.

[2] C. Elliott, W. Chuma, and Y. E. Gendi, "Hate Speech, Key Concept Paper", *Media Conflict and Democratisation (MeCoDEM)*, United Kingdom, 2016.

[3] M. Subramanian, S. V. Easwaramoorthy, G. Deepalakshmi, J. Cho, and G. Manikandan, "A Survey on Hate Speech Detection and Sentiment Analysis Using Machine Learning and Deep Learning Models", *Alexandria Engineering Journal*, vol. 80, pp. 110-121, 2023.

[4] C. M. Murphy and D. McCashin, "Using TikTok for Public and Youthmental Health - a Systematic Review and Content Analysis", *Clinical Child Psychology and Psychiatry*, vol. 28, no. 1, pp. 279-306, 2023.

[5] D. Zulli and D. J. Zulli, "Extending the Internet Meme: Conceptualizing Technological Mimesis and Imitation Publics on the TikTok Platform", *New Media and Society*, vol. 24, no. 8, pp. 1872-1890, 2022.

[6] S. V. Mahardhika, I. Nurjannah, I. I. Ma'una, and Z. Islamiyah, "Faktor-Faktor Penyebab Tingginya Minat Generasi Post-Millenial Di Indonesia Terhadap Penggunaan Aplikasi Tik-Tok", *SOSEARCH: Social Science Educational Research*, vol. 2, no. 1, pp. 40-53, 2021.

[7] R. N. Ria and T. Setiawan, "Forensic Linguistic Analysis of Netizens' Hate Speech Acts in Tik-Tok Comment Section", *Britain International of Linguistics Arts and Education (BIoLAE) Journal*, vol. 5, no. 2, pp. 141-152, 2023.

[8] E. Prasetyo, M. F. Al-adni, and R. F. Tias, "Classification of Cash Direct Recipients Using the Naive Bayes with Smoothing", *Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer*, vol. 23, no. 3, pp. 615-626, 2024.

[9] A. Ali, A. Khairan, F. Tempola, and A. Fuad, "Application of Naïve Bayes to Predict the Potential of Rain in Ternate City", *E3S Web of Conferences*, vol. 328, 2021.

[10] D. A Pisner and D. M. Schnyer, *Support Vector Machine. In Machine Learning: Methods and Applications to Brain Disorders*, Elsevier Inc, 2019.

[11] R. K. Putri, M. Athoillah, A. Haqiqiyah, and F. W. A. Lestari, "Deteksi Penggunaan Masker Wajah Dengan Algoritma Deep Learning", *Prosiding Seminar Nasional Hasil Riset dan Pengabdian*. Surabaya, 2023.

[12] V. Jackins, S. Vimal, M. Kaliappan, and M. Y. Lee, "AI-based Smart Prediction of Clinical Disease Using Random Forest Classifier and Naive Bayes, *Journal of Supercomputing*, vol. 77, no. 5, pp. 5198-5219, 2021.

[13] Y. Tan and P. P. Shenoy, "A Bias-Variance Based Heuristic for Constructing a Hybrid Logistic Regression-Naïve Bayes Model for Classification", *International Journal of Approximate Reasoning*, vol. 117, pp. 15-28, 2020.

[14] A. P. Noto and D. R. S. Saputro, "Classification Data Mining with Laplacian Smoothing on Naïve Bayes Method, *AIP Conference Proceedings*, Solo, 2022.

[15] J. Pan, M. Sun, Y. Wang, and X. Zhang, "An Enhanced Spatial Smoothing Technique with ESPRIT Algorithm for Direction of Arrival Estimation in Coherent Scenarios", *IEEE Transactions on Signal Processing*, vol. 68, pp. 3635-3643, 2020.

[16] A. W. Pradana and M. Hayaty, "The Effect of Stemming and Removal of Stopwords on the Accuracy of Sentiment Analysis on Indonesian-language Texts", *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, vol. 4, no. 3, pp. 375-380, 2019.

 | doi: 10.23960/pepadun.v6i3.268

[17] M. M. Saritas and A. Yasar, "Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification", *IJISAE: International Journal of Intelligent Systems and Applications in Engineering*, vol. 7, no. 2, pp. 88-91, 2019.

[18] A. Ariska and M. Kamayani, "Deteksi Hate Speech pada Kolom Komentar TikTok dengan Menggunakan SVM", *Indonesian Journal of Computer Science*, vol. 13, no. 3, pp. 284-301, 2024.

[19] D. Febiharsa, I. M. Sudana, and N. Hudallah, "Uji Fungsionalitas (Blackbox Testing) Sistem Informasi Lembaga Sertifikasi Profesi (SILSP) Batik dengan AppPerfect Web Test dan Uji Pengguna", *Joined Journal (Journal of Informatics Education)*, vol. 1, no. 2, pp. 117, 2019.

[20] A. Verma, A. Khatana, and S. Chaudhary, "A Comparative Study of Black Box Testing and White Box Testing", *International Journal of Computer Sciences and Engineering*, vol. 5, no. 12, pp. 301-304, 2017.

[21] F. Koto and Y. R. Gemala, "InSet Lexicon: Evaluation of a Word List for Indonesian Sentiment Analysis in Microblogs", *International Conference on Asian Language Processing (IALP)*, pp. 391-394, 2017.

[22] Z. Zhu, J. Liang, D. Li, H. Yu, and G. Liu, "Hot Topic Detection Based on a Refined TF-IDF Algorithm", *IEEE Access*, vol. 7, pp. 26996-27007, 2019.

[23] K. Teguh, K. Kridalukmana, R. Rinta and M. Martono, "Pembuatan Chrome Extension untuk Akses Website Sistem Komputer", *Proceedings Business Intelligence: Extending Your Business*, pp. 81-92, 2012.

[24] M. N. Huda, M. Burhan, A. Satibi, H. A. Pradita, and A. Saifudin, "Implementasi Black Box Testing pada Aplikasi Sistem Kasir dengan Menggunakan Teknik Equivalence Partitions", *Jurnal Teknologi Sistem Informasi dan Aplikasi,* 2023.

[25] S. Robinson and M. Heusser, "Black-box testing", *TechTarget: SearchSoftwareQuality*, 2024.

[26] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing (3rd ed.)*, Wiley, 2011.

[27] P. Ammann, and J. Offutt, *Introduction to Software Testing (2nd ed.)*, Cambridge University Press, 2016.

[28] T. T. A. Putri, S. Sriadhi, R. D. Sari, R. Rahmadani, and H. D. Hutahaean, "A Comparison of Classification Algorithms for Hate Speech Detection", *IOP Conference Series: Materials Science and Engineering*, vol. 830, no. 3, 2020.

[29] N. R. Fatahillah, P. Suryati, and C. Haryawan, "Implementation of Naive Bayes Classifier Algorithm on Social Media (Twitter) to the Teaching of Indonesian Hate Speech", *Proceedings - 2017 International Conference on Sustainable Information Engineering and Technology*, pp. 128-131, 2017.