Sentiment Analysis with Multi-Models: A Focus on Hate Speech Detection

Keyu He   Haofeng Xu   Qiang Zeng

Abstract

In the digital era, the increasing prevalence of online hate speech poses a severe societal challenge. This study evaluates advanced machine learning models, including Naive Bayes and Bidirectional Encoder Representations from Transformers (BERT), for effective hate speech detection in online content. We adopt a novel comparative approach to assess these methodologies, focusing on their capability to handle the complex nuances of hate speech. Our rigorous evaluation, using metrics such as accuracy, precision, recall, and F1-score, indicates a marked variance in performance across models. Notably, while the Naive Bayes show limited effectiveness, the BERT model demonstrates superior performance, substantially outperforming the baseline. These findings underscore the potential of transformer-based models in addressing the challenges of hate speech detection in the digital landscape.

1. Introduction

There is an increasing prevalence of online hate speech, which can lead to real-world harm, including mental health issues and the perpetuation of discrimination and violence. This endeavor is driven by the imperative need to mitigate the proliferation of hate speech on social media platforms.

This project is driven by the goal of enhancing the safety and inclusivity of online environments through the automated detection and moderation of hate speech. We focus on the identification of hate speech within a dynamically generated dataset (Sharengaraju, 2021), which presents unique challenges due to its evolving nature and the subtle complexities inherent in human language.

The investigation led us through a detailed analysis of the three models, each evaluated based on its ability to accurately classify text into 'hate' or 'not hate' categories. Our findings painted a nuanced picture of model performance. The Naive Bayes model, traditionally favored for its simplicity and efficiency in text classification, delivered moderate results. It was able to go beyond mere keyword matching but struggled with the subtler aspects of language that are often pivotal in correctly identifying hate speech.

The BERT model stood out for its exceptional performance. Its sophisticated mechanism for processing text in a bidirectional context allowed for a far deeper and more accurate understanding of the intricacies involved in differentiating hate speech from benign expressions. BERT's advanced approach to contextual analysis proved to be a significant asset, leading to substantially higher accuracy and setting a new benchmark in our project.

It becomes evident that while traditional models like Naive Bayes have their merits, the advanced capabilities of models like BERT are indispensable for effectively navigating the complexities of hate speech detection. These outcomes offer a promising direction for future research and practical applications in the realm of online content moderation.

2. Related Work and Proposed Methodologies

The field of hate speech detection has been explored through various innovative approaches, offering valuable insights into model performance and robustness. This project's methodologies are informed by the findings in existing literature.

2.1. Avoiding Overfitting and Bias

Dixon et al. (Dixon et al., 2018) not only highlighted the challenge of overfitting in text classification models but also demonstrated how imbalances in training data can lead to unintended bias in the resulting models. This can result in potentially unfair applications, a significant concern in hate speech detection. Our approach addresses these concerns in two ways:

- **Dataset Balance**: In response to the issues raised by Dixon et al., we utilized a balanced dataset for training our models. This helps in mitigating the bias that can arise from skewed data distributions.

- **Cross-Validation**: We also employed k-fold cross-validation, particularly in training our Naive Bayes model, to ensure that the model learns generalized patterns and remains accurate when confronted with new data.

2.2. Contextualizing Hate Speech Classifiers

Kennedy et al. (Kennedy et al., 2020) emphasized the importance of context in classifying hate speech accurately. Their work involved developing context-aware classification models, which is aligned with our decision to utilize BERT for its superior contextual understanding capabilities. BERT's bidirectional processing of text enables a deeper understanding of language nuances, addressing the complex and subtle aspects of hate speech detection more effectively.

Extending beyond Kennedy et al.'s focus on context, we integrate BERT for its innate deep contextual analysis capabilities, offering an advanced approach to understanding the complexities of hate speech.

Informed by Dixon et al. (Dixon et al., 2018) and Kennedy et al. (Kennedy et al., 2020), our project strategically uses balanced datasets, cross-validation, and context-aware models like BERT to develop a robust hate speech detection system. This system is designed to be fair, unbiased, and effective across diverse online platforms, tackling the unique challenges of hate speech detection in the digital era.

3. Dataset and Evaluation

3.1. Dataset

A critical aspect of any machine learning project, particularly in the realm of natural language processing, is the selection of an appropriate and representative dataset. In the field of hate speech detection, however, many existing hate classification datasets have imbalance issues, such as the Kaggle's "Hate Speech and Offensive Language Dataset", where most samples are skewed heavily towards offensive language. Such imbalances can lead to significant biases in model training, potentially resulting in models that are overfitted to the dominant class and unable to generalize effectively to real-world data. To address this, we chose Kaggle's "Dynamically Generated Hate Speech Dataset," comprising 40,463 samples with a more balanced distribution: 54% hate speech and 46% non-hate. For our project, we retained only the text and label columns.

The dataset itself comprises sentences that are algorithmically generated, encompassing a diverse array of hate speech manifestations. These sentences are then meticulously labeled by human annotators, ensuring that the classification of 'hate' or 'not hate' accurately captures the nuanced and often subjective nature of hate speech. This human element in the labeling process is critical, as it imbues the dataset with a level of complexity and realism that purely algorithmic approaches might miss.

For the purposes of our project, we focused exclusively on the text and label columns of the dataset. The text column contains the sentences identified as potential instances of hate speech, while the label column provides the corresponding classification. This streamlined approach allows us to concentrate our analysis on the linguistic content of the text and its classification, which are the most pertinent aspects for our study on hate speech detection.

3.2. Dataset Split

For systematic model training and evaluation, we shuffled and partitioned our dataset into three subsets: training, development (or validation), and testing. Specifically:

- Training Set (70%): Used primarily for model training, capturing patterns and nuances from the majority of the data.
- Development Set (10%): A crucial subset for model evaluation and tuning. After initial training, the model's performance on the development set guides potential refinements.
- Testing Set (20%): Provides an unbiased performance measure on entirely unseen data, ensuring our model evaluations are genuine.

3.3. Evaluation

3.3.1. Metrics Selection

Model performance was gauged through multiple conventional metrics for a comprehensive understanding:

- **Accuracy:** This metric represents the proportion of correctly predicted instances (both hate and non-hate) out of the total number of predictions made. A higher accuracy indicates that the model is generally reliable in its predictions across both classes. This is evaluated as:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All Predictions}}$$

- **Precision:** Precision focuses on the proportion of true positive predictions (correctly identified hate speech) in relation to all positive predictions (all instances predicted as hate speech, correctly or not). High precision implies that when the model predicts a text as hate speech, it is likely to be correct, thereby reducing the occurrence of false positives. It is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall:** Also known as sensitivity, recall measures the ability of the model to correctly identify actual instances of hate speech. It is the ratio of true positive predictions to the total number of actual hate speech instances. High recall indicates that the model is effective in capturing most of the true instances of hate speech, missing very few. It is given by:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1-Score:** The F1-Score is the harmonic mean of precision and recall. It is a single metric that balances both precision and recall, providing a more holistic view of the model's performance. A high F1-Score suggests that the model has a balanced trade-off between precision and recall, excelling in both aspects. It is calculated as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.3.2. Evaluation Steps

1. **Development Phase:** Models are initially trained on an 70% training subset and evaluated using a 10% development set. This flags any overfitting and guides hyperparameter tuning.

2. **Final Testing:** After the models are tuned, they are subjected to a final evaluation on a distinct 20% test subset of the data. This phase is critical as it simulates real-world conditions, providing an unbiased assessment of the models' performance in a setting that mirrors actual usage scenarios.

3. **Analysis of Results:** The last step involves a thorough analysis of the models' performance metrics – accuracy, precision, recall, and F1-score. This comprehensive evaluation is designed to verify not only the models' ability to detect explicit instances of hate speech but also their effectiveness in identifying more nuanced and subtle expressions. A balance of high accuracy with strong precision and recall scores is indicative of a model proficient in accurately identifying hate speech while minimizing false positives.

4. Methods

In our endeavor to accurately classify sentiments, multiple approaches, ranging from basic to advanced techniques, will be implemented and assessed. This diverse set of methods ensures a comprehensive understanding of model capabilities and provides a clear comparison of their performances.

4.1. Majority-Rule Baseline

The Majority Classifier serves as our baseline model. This simplistic approach always predicts the most frequent label observed in the training dataset, irrespective of the input it encounters during the testing phase. The rationale behind employing this method is to establish a foundational performance metric, against which the efficacy of more sophisticated models can be juxtaposed.

4.1.1. Procedure

1. Load the dataset and segregate it into features (text) and labels (label).
2. Split the dataset into training and testing subsets.
3. Identify the majority class label within the training set.
4. Generate a prediction array for the testing set, populated exclusively with the majority class label.
5. Evaluate the model's performance using accuracy, precision, recall, and F1-score.

4.2. Naive Bayes

The Naive Bayes classifier, based on Bayes' theorem, is a probabilistic model that calculates the probability of a class label based on the features of the instance. It operates under the assumption of conditional independence between features given the class label. The classifier's effectiveness in text classification tasks stems from its ability to handle high-dimensional data and make predictions based on the probabilities derived from feature-class relationships.

The formula for the Naive Bayes classifier is given by:

$$P(C_k|x) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}{P(x)}$$

Where:

- $P(C_k|x)$ is the posterior probability of class C_k given predictor x. ($C_k \in \{0, 1\}$, where 0 represents "not hate" while 1 represents "hate")

- $P(C_k)$ is the prior probability of class C_k.

- $P(x_i|C_k)$ is the likelihood, which is the probability of predictor x_i given class C_k.

- $P(x)$ is the prior probability of predictor x. (We do not directly compute this, instead, we find $\text{argmax}_{C_k} P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$)

The x_i terms we used here refer to the features derived from the TF-IDF vectorization of the text data.

4.2.1. Feature Extraction using TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure used to evaluate the importance of a word to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The TF-IDF value increases proportionally with the number of times a word appears in the document but is offset by the frequency of the word in the corpus, which helps control for the fact that some words are generally more common than others.

The TF-IDF for a word in a document is calculated as:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

Where:

- TF(t, d) is the term frequency of term t in document d.

- IDF(t) is the inverse document frequency of term t, calculated as $\log(\frac{N}{df_t})$, with N being the total number of documents and df_t the number of documents containing term t.

In our implementation, we used the TF-IDF vectorizer to convert the raw text into a numerical format. We set a maximum feature limit of 5000 and removed common English stop words. The resultant TF-IDF vectors replace the traditional count-based features in the Naive Bayes classifier, providing a more nuanced and weighted representation of text for classification.

4.2.2. Hyperparameter Tuning and Cross-Validation

We focused on tuning the Laplace smoothing parameter α of the Naive Bayes classifier, which is critical in handling the problem of zero probability in unseen data. A range of α values was tested: $[0.001, 0.01, 0.1, 0.5, 1, 2, 5, 10]$. To determine the optimal α, we employed a rigorous k-fold cross-validation approach with $k = 5$. This method involved dividing the training dataset into k subsets, and training the model k times, each time using a different subset as the validation set and the remaining as the training set. The cross-validation process helps ensure that our model is not just tuned to a specific subset of the data, thereby enhancing its generalizability and robustness. The optimal α was selected based on the configuration that yielded the highest accuracy during this process.

4.2.3. Model Evaluation

The optimized Naive Bayes model was then evaluated on the test set. Performance metrics including accuracy, precision, recall, and F1-score were employed to comprehensively assess the model's capability in accurately classifying text as either hate speech or non-hate speech.

4.3. BERT

The BERT (Bidirectional Encoder Representations from Transformers) model, developed by Google, represents a significant leap in natural language processing. Utilizing a transformer architecture, BERT captures the contextual relationships between words by considering their full context, looking at the words that come before and after. This method is a notable advancement over traditional models like Naive Bayes, which rely primarily on keyword frequency without contextual awareness. BERT's ability to interpret entire sentence structures leads to more accurate detection of hate speech, effectively addressing the challenge of subtlety and nuances in language. This capability makes BERT exceptionally suited for tasks that require nuanced language processing, such as accurately identifying hate speech, a critical requirement that had been a key limitation in earlier methods like Naive Bayes.

(Devlin et al., 2018).

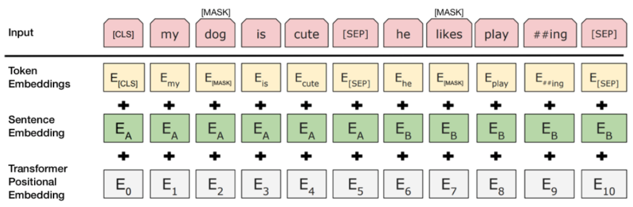
Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 1. The BERT model

The BERT model is pre-trained on a large corpus of text and then fine-tuned for specific tasks like classification, question-answering, or sentiment analysis.

4.3.1. Fine-Tuning BERT for Classification

Fine-tuning the pre-trained BERT model involves training it on a smaller dataset specific to the task at hand. For our classification task, we add a dense layer on top of the BERT architecture, which takes the output representations from BERT and processes them to produce class probabilities.

The architecture of our BERT-based classifier is as follows:

$$\text{Output} = \text{DenseLayer}(\text{BERT}(x))$$

Where:

- BERT(x) represents the BERT model processing the input text x.

- The DenseLayer is a fully connected neural network layer that outputs the probability of each class.

Our implementation utilizes the 'bert-tiny' version of the model, which is a smaller, more computationally efficient variant of the original BERT model designed for environments with constraints on model size and processing power. The dataset is tokenized using BERT's tokenizer, ensuring compatibility between the model architecture and the input data format.

4.3.2. Tokenization and Input Representation

BERT requires a specific input format, which includes tokenized text data, segment IDs, and attention masks. The tokenization process involves converting each word into a token that corresponds to an entry in BERT's vocabulary. Special tokens, such as $[CLS]$ for the start of a sequence and $[SEP]$ for the end of a sequence or separation between sentences, are also added.

The input representation for each text in our dataset is formed as follows:

$$\text{Input} = \text{Tokenize}([CLS], x, [SEP])$$

Where:

- Tokenize is the function applying BERT's tokenization process.

- x is the raw text of the instance.

- $[CLS]$ token stands for "Classifier" and is used at the beginning of every input sequence in BERT. In our project, when classifying a text as hate speech or not, the model looks at the transformed representation of this $[CLS]$ token to make a decision.

- $[SEP]$ token, short for "Separator," is used in BERT to denote the end of a sentence or to separate two different sentences.

The tokenized inputs are then padded or truncated to a fixed length and passed through the BERT model for training and classification tasks.

4.3.3. Hyperparameter Optimization

In deep learning, the fine-tuning of hyperparameters is crucial for enhancing model performance. Hyperparameters in BERT networks, such as the number of learning rates, optimizer types, and batch sizes, significantly influence the learning process and final model accuracy. However, manually searching for the optimal combination of these parameters is time-consuming and computationally expensive.

Role of WandB We employed Weights & Biases (WandB), an advanced tool for machine learning experiment tracking, to streamline and optimize the hyperparameter tuning process of our BERT model. WandB is particularly beneficial for its capability to automate the exploration of hyperparameter spaces efficiently.

Hyperparameter Combinations Explored with WandB In our BERT model optimization, several key hyperparameters were identified for the WandB sweep. Below is a detailed enumeration of each hyperparameter and the range of values that were explored:

1. **Learning Rate:** The learning rate is a critical parameter in training neural networks, influencing the speed and quality of learning. We experimented with a range of learning rates from 1e-5 to 5e-5 to find the optimal setting for our BERT model.

2. **Optimizer Type:** The choice of optimizer can impact the convergence and performance of the model. We evaluated several optimizers: Adam, Rmsprop, and SGD.

3. **Batch Size:** Batch size influences the speed and stability of the learning process. We included several batch sizes (16, 32, 64, 128, and 256) in our sweep to determine the optimal size for our dataset.

5. Experiment

5.1. Baseline

As part of our experiment, the Majority Classifier was chosen as the baseline. As previously stated, this model always predicts the label that is most frequent in the training dataset. The main purpose of this model was to provide a fundamental metric, making it easier to understand the advancements made by our more sophisticated models. We got an accuracy of 54.16% for baseline model.

The Confusion Matrix for the baseline is shown in Figure 2.

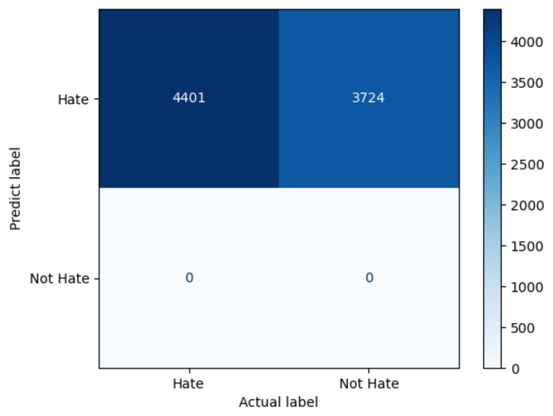

Figure 2. Confusion Matrix for Baseline

The evaluation metrics for the Baseline are shown in table 1. Note that we reach 1 in recall as the baseline does not give any negative labels, causing $FN = 0$.

5.2. Naive Bayes

We ran the Naive Bayes model multiple times to find the best Laplace smoothing parameter alpha:

Metric	Value
Accuracy	0.542
Precision	0.542
Recall	1.000
F1 Score	0.703

Table 1. Evaluation Metrics for Baseline

Alpha	Accuracy
0.001	0.6596
0.01	0.6606
0.1	0.6594
0.5	0.6594
1	0.6606
2	0.6616
5	0.6530
10	0.6426

Table 2. Accuracy of Naive Bayes for Varying Alpha

In table 2, we changed alpha for Naive Bayes model; higher values indicate more smoothing. The table suggests that alpha $= 2$ gives the best dev accuracy for Naive Bayes at 66.16%, which suggests that some amount of smoothing benefits the model, but too much (alpha $= 10$) or too little (alpha $= 0.001$) is detrimental.

By utilizing the best option for alpha (alpha=2), we trained the model and got train accuracy of 72.54%, dev accuracy 66.16%, and test accuracy of 67.27%.

The Confusion Matrix for the Naive Bayes is shown in Figure 3, and the evaluation metrics for the Naive Bayes are shown in table 3.

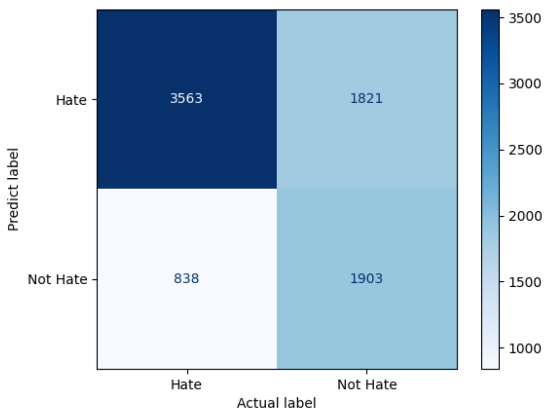

Figure 3. Confusion Matrix for Naive Bayes

The higher recall and lower precision in our Naive Bayes model can be attributed to its reliance on keyword detection.

Metric	Value
Accuracy	0.673
Precision	0.662
Recall	0.810
F1 Score	0.728

Table 3. Evaluation Metrics for Naive Bayes

Naive Bayes tends to classify sentences as hate speech if they contain certain keywords often found in hate speech, regardless of context. This leads to many true positives but also a substantial number of false positives, where non-hate speech is incorrectly labeled as hate. Consequently, while the model is effective at identifying most hate speech instances (high recall), its precision is compromised by mistakenly classifying many non-hate sentences as hate speech.

The Naive Bayes model outperforms the Majority Classifier baseline, having around 0.13 higher in accuracy. This shows that the Naive Bayes model is capturing some underlying patterns in the data, while the Majority Classifier only serves as a rudimentary measure.

In summary, Naive Bayes has some ability to generalize from the training data. However, 67% accuracy is not very satisfying, we need a more powerful model to reach higher accuracy.

5.3. BERT

In our study, we utilized a Weights and Biases (WandB) system for the effective optimization of hyperparameters in the BERT model.

5.3.1. Sweep Results Visualization

A crucial aspect of our methodology was the visualization of sweep results. We present a result from the WandB dashboard (Figure 4), which showcases the outcomes of various runs in the sweep. This visual representation offers an intuitive understanding of how different hyperparameter configurations impacted the model's performance. The metrics tracked in each run include accuracy, loss, and other relevant performance indicators, providing a clear comparison between different experimental setups.

The optimal set of hyperparameters is: **learning rate = 3.334e-5, optimizer = Adam,** and **batch size = 16**.

Remarkably, with the optimal hyperparameter configuration determined through our systematic analysis, the BERT model exhibited exceptional performance, significantly surpassing the other models we evaluated in this study. These outcomes reinforce the capability of BERT in handling com-

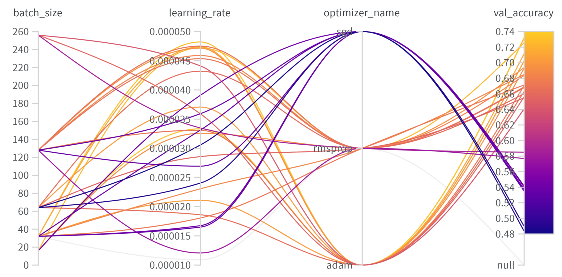

Figure 4. Visualization of Hyperparameter Optimization Results in WandB Dashboard

plex natural language processing tasks. In the ensuing sections, we will provide a more in-depth exploration of these results, aiming to uncover the underlying factors that contribute to the superior performance of the BERT model.

We then utilized the best combination of hyperparameters identified for both training and testing the model. Notably, we set the number of training epochs to 20 and we incorporate an early stopping mechanism to enhance the training efficiency and prevent overfitting. During the training process, the model achieved a training accuracy of 0.8718, a validation accuracy of 0.7871, and a test accuracy of 0.7950.

Figure 5 shows the confusion matrix for BERT, after we choose the optimal hyperparameter and run 8 epochs, and table 4 illustrates the specific metrics achieved with this optimal configuration.

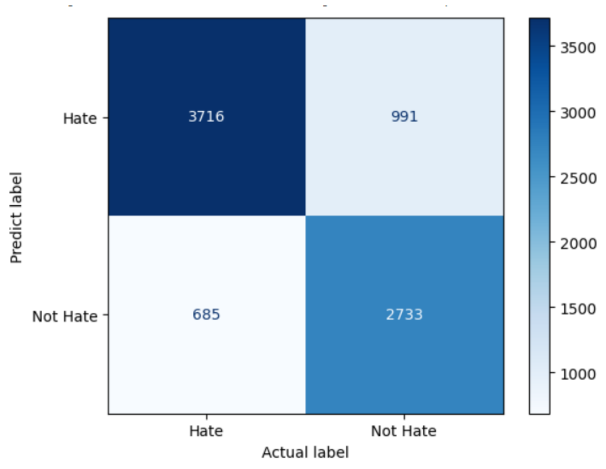

Figure 5. Confusion Matrix for BERT (with optimal hyperparameter chosen)

It is important to highlight that the training was halted at the 8th epoch, a decision governed by the early stopping criteria set in our training configuration. The patience parameter was configured to halt training if there was no improvement in the loss metric for 3 consecutive epochs. This strategic approach of early stopping played a pivotal role in ensuring that the model did not overfit to the training data while still achieving high levels of accuracy. Such a methodology not only optimized the training duration but also ensured the robustness and generalizability of the model, as evidenced by the consistency of performance metrics across training, validation, and test datasets.

Metric	**Value**
Accuracy	0.795
Precision	0.789
Recall	0.848
F1-Score	0.818

Table 4. Evaluation Metrics for BERT

These results underscore the effectiveness of our chosen hyperparameter strategy and the utility of BERT in efficiently processing and classifying complex textual data. The model's ability to cease training at an optimal point further demonstrates the sophistication of the learning process and the appropriateness of the chosen hyperparameters.

5.4. Comparative Analysis and Implications

The experimental results reveal notable differences in the performance of the Majority Classifier, Naive Bayes, and BERT models in the context of hate speech detection. The Majority Classifier, with its simplistic approach, provided a baseline performance, achieving an accuracy of 54.16%. In contrast, the Naive Bayes model showed a marked improvement with a test accuracy of 67.27%. Unsurprisingly, it was the BERT model that stood out, achieving a test accuracy of 79.50%, significantly higher than the other models.

These performance differences can be attributed to the inherent characteristics and capabilities of each model:

- **Majority Classifier**: Its lower performance is expected as it does not consider any features of the input text, merely predicting the most frequent label in the training set. This approach lacks any linguistic analysis, making it a rudimentary measure of classification capability.
- **Naive Bayes**: The improvement in performance over the baseline is due to its ability to analyze text features (using TF-IDF vectorization) and predict based on statistical relationships. However, its limitations in handling context and subtleties in language, as well as its sensitivity to specific keywords, explain why it did not perform as well as BERT.
- **BERT**: The superior performance of BERT is at-

tributed to its deep learning architecture and ability to understand the context of words in a sentence. This model captures nuanced meanings and complex linguistic structures, making it particularly effective for tasks like hate speech detection.

5.4.1. Implications

The results of this study have several implications:

- **Model Selection**: The choice of model plays a crucial role in the effectiveness of hate speech detection. While simpler models may provide quicker and more computationally efficient solutions, their effectiveness is limited compared to more advanced models like BERT.
- **Contextual Analysis**: Models capable of understanding context and nuances in language, like BERT, are essential for accurately identifying hate speech, which often involves complex and subtle expressions.
- **Model Optimization**: The tuning of hyperparameters and the incorporation of mechanisms like early stopping, as demonstrated in the BERT model, are vital for optimizing performance and preventing overfitting.

In conclusion, our experiments underscore the importance of advanced NLP techniques and models for effective hate speech detection. The stark differences in performance among the models highlight the need for sophisticated tools capable of deep linguistic and contextual analysis in addressing the challenges of moderating online content.

6. Discussion

6.1. Baseline

The baseline method Majority Classifier, by design, is a rudimentary model that offers limited insights into the nuances of hate speech detection. Its simplistic approach of always predicting the predominant label from the training set inherently restricts its ability to discern between different classes based on the content of the text.

6.1.1. Error Analysis

Upon examining a random sample from the development set, it became evident that the Majority Classifier's errors were consistent—it invariably misclassified all instances from the 'not hate' class. This behavior is expected given its design. The model's inability to analyze text content means it cannot differentiate between hate speech and non-hate speech, leading to a high false positive rate for the 'hate' class.

6.2. Naive Bayes

6.2.1. Error Analysis

Our analysis of misclassified samples from the Naive Bayes model provides insights into its limitations in accurately detecting hate speech. Key issues identified are:

1. **Sensitivity to Specific Keywords:** The model tends to overemphasize certain keywords associated with hate speech, leading to false positives.

   Example: *"I am seeing less openly gay people in this area compared with a few years ago."*
   Actual Label: *not hate*, Predicted Label: *hate*.
   Despite being a neutral observation, the mention of 'gay people' likely influenced its misclassification, indicating an over-reliance on specific keywords rather than the overall sentiment.

   Example: *"The government is not making things right for small business owners and keeps giving benefits to the rich business ......so what do they do, blame immigrants!"*
   Actual Label: *not hate*, Predicted Label: *hate*.
   This sentence, critical of government policies, is wrongly classified as hate speech due to the presence of the word 'immigrants'.

2. **Difficulty with Subtlety and Nuance:** Naive Bayes struggles with sentences that contain subtle or nuanced expressions.

   Example: *"You should move from your country dude. USA is sieged by j3ws, just like Europe is..."*
   Actual Label: *hate*, Predicted Label: *not hate*.
   Here, the model fails to identify the underlying hate speech, possibly due to the nuanced way it is expressed, underscoring a limitation in detecting subtle forms of hate speech.

   Example: *"black dogs matter more than black lives."*
   Actual Label: *hate*, Predicted Label: *not hate*.
   The statement is incorrectly classified as non-hate, likely due to the lack of direct hateful expressions despite the underlying offensive comparison.

6.2.2. Discussion on Performance

The Naive Bayes classifier, while efficient in many text classification tasks, exhibits notable limitations in the context of hate speech detection. Its challenges in oversensitivity to specific keywords, and difficulty in processing subtleties and nuanced expressions lead to both false positives and negatives. The model's reliance on word frequencies and inability to understand the broader context and tone of a statement significantly impact its accuracy.

6.2.3. Recommendations for Improvement

To enhance the Naive Bayes model's performance in hate speech detection, we recommend:

1. **Incorporation of Contextual Features:** Introducing features that capture the broader context of a statement could help mitigate the model's limitations in understanding complex language structures.

2. **Refinement of Feature Selection:** Adjusting the model to reduce its sensitivity to specific keywords and to better interpret the overall sentiment and tone of a statement.

3. **Hybrid Modeling Approaches:** Combining Naive Bayes with other models that excel in processing contextual and nuanced language could improve overall detection accuracy.

In conclusion, while the Naive Bayes model is valuable for its efficiency and simplicity, its limitations in accurately detecting hate speech underscore the need for further development and integration with more context-aware approaches in sensitive applications like online content moderation.

6.3. BERT

6.3.1. Error Analysis

BERT performs much better than the Naive Bayes model, and correctly labeled most of the samples which Naive Bayes model misclassified. However, it still has nonnegligible error.

Our in-depth manual error analysis of the BERT model, focusing on a selection of misclassified samples, provides valuable insights into the model's limitations and challenges in accurately detecting hate speech.

1. **Ambiguity and Subtlety in Language:** The model sometimes misclassified statements with nuanced or ambiguous expressions of hate speech.

   Example: *"It's not as if they were planning to mix us out of existence by using mass immigration of young men."*
   Actual Label: *not hate*, Predicted Label: *hate*.
   This sentence, though containing potentially charged language, does not directly express hate speech. Its classification as 'hate' reveals BERT's difficulty in navigating complex sentences where intent is not explicit.

2. **Challenges with Context-Dependent Interpretation:** BERT struggled with sentences requiring an understanding of broader social contexts or personal experiences.

   Example: *"I am invited to the wedding of gay people I don't know what to do."*
   Actual Label: *not hate*, Predicted Label: *hate*.
   This statement reflects personal uncertainty rather than hate speech. The model's misclassification suggests a limitation in differentiating between hate speech and expressions of personal confusion or dilemma.

3. **Misinterpretation of Neutral Statements with Charged Keywords:** The model sometimes misclassified neutral statements due to the presence of words typically associated with hate speech.

   Example: *"black widows are disgusting, they kill their victims horribly."*
   Actual Label: *not hate*, Predicted Label: *hate*.
   Here, the reference to 'black widows' (spiders) is factual, yet the model misinterprets it as hate speech, possibly due to the negative descriptors. This indicates a challenge in distinguishing between literal descriptions and metaphorical or derogatory usage of terms.

4. **Occational Misclassification of Explicit Hate Speech** BERT occasionally failed to identify clear instances of hate speech.

   Example: *"the jews should be destroyed wtf."*
   Actual Label: *hate*, Predicted Label: *not hate*.
   This blatant expression of hate speech being classified as 'not hate' highlights a significant gap in the model's training or a potential anomaly in how it processes explicit hate speech.

6.3.2. Discussion on Performance

Despite BERT's advanced capabilities in contextual understanding, our analysis reveals its limitations in handling subtleties, ambiguities, and contextually complex statements in hate speech detection. The model's occasional failure to recognize explicit hate speech and its misinterpretation of neutral statements underscore the need for further refinement in model training and contextual awareness.

6.3.3. Recommendations for Improvement

Based on our findings, we recommend the following enhancements to improve BERT's performance:

1. **Contextual Data Enrichment:** Incorporating more diverse and contextually rich examples in the training set could help the model better understand the complexities and nuances of language used in hate speech.

2. **Fine-Tuning for Subtlety and Ambiguity:** Adjusting the model to better capture and interpret subtle and ambiguous expressions of hate speech, including indirect and sarcastic remarks.

3. **Hybrid Approaches:** Employing hybrid models that combine BERT's strengths with other models adept at processing sarcasm, irony, or metaphor could enhance overall detection accuracy.

In conclusion, while BERT exhibits strong potential in hate speech detection, its current limitations in interpreting complex language nuances highlight the necessity for ongoing model development and human oversight in sensitive applications like content moderation.

6.4. Overall Performance and Expectations

6.4.1. Model Expectations vs. Reality

The performance of the models in our experiments presented both expected and surprising elements. The Majority Classifier, as anticipated, served merely as a rudimentary baseline, lacking any real analytical capability. The Naive Bayes model, while expected to perform moderately well, demonstrated a significant improvement over the baseline, particularly in its ability to capture keyword-based patterns. However, its limitations in contextual understanding were more pronounced than initially hypothesized, especially in cases involving subtlety and nuance.

The most notable revelation came from the BERT model. Its ability to understand context and nuance in text was remarkable as anticipated, although it did encounter challenges with ambiguities and complex sentence structures.

6.4.2. Surprising Model Outcomes

Certain outcomes from the model evaluations were particularly surprising:

- **Naive Bayes**: The extent to which this model was influenced by specific keywords, often leading to misclassifications even in seemingly neutral contexts, was more pronounced than expected.
- **BERT**: While its overall high performance was anticipated, BERT's occasional failures to recognize clear instances of hate speech or its misclassification of neutral statements with charged keywords highlighted the complexity of hate speech detection. Such instances underscored the model's need for further refinement to better grasp the subtleties and complexities of language.

7. Conclusion

In this study on hate speech detection, we evaluated various models, including Naive Bayes and the more advanced BERT. Our findings highlight the distinct capabilities and limitations of each model in addressing the complexities of automated hate speech detection.

Naive Bayes demonstrated moderate effectiveness, particularly in identifying explicit forms of hate speech. However, it struggled with contextually nuanced language, often misclassifying sentences based on specific keywords.

The BERT model, with its advanced architecture, showed superior performance in understanding the intricacies of language in hate speech. Despite its strengths, BERT also faced challenges in interpreting ambiguous or context-dependent statements.

This study underscores the importance of advanced models like BERT for tasks requiring deep contextual understanding. However, the limitations observed in both models suggest a need for continuous improvement and human oversight.

We propose future work to focus on enhancing data enrichment, fine-tuning model sensitivity to subtleties, and exploring hybrid modeling approaches. These efforts aim to further refine hate speech detection systems, contributing to more effective and nuanced online content moderation.

8. Code & Data Submission

The code and dataset for this project are at https://github.com/Clara-z/CSCI467-final-project.

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dixon, L., Li, J., Sorensen, J., Thain, N., and Vasserman, L. Measuring and mitigating unintended bias in text classification. In *AAAI/ACM Conference on AI, Ethics, and Society*, pp. 67–73, 2018.

Kennedy, B., Jin, X., Davani, A. M., Dehghani, M., and Ren, X. Contextualizing hate speech classifiers with post-hoc explanation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5435–5442, 2020.

Sharengaraju, U. Dynamically generated hate speech dataset. Kaggle, 2021. URL https://www.kaggle.com/datasets/usharengaraju/dynamically-generated-hate-speech-dataset/data.