# The Application of Sentiment Models to Hate Speech Detection

**Marina Sanchez**
New College of Florida
marina.sanchezmill23@ncf.edu

**Mileva Van Tuyl**
New College of Florida
mileva.vantuyl23@ncf.edu

## Abstract

In this paper, we consider three sentiment models (Naive Bayes, LSTM, and VADER) and study how the models perform on a hate speech detection task in order to determine if sentiment models are effective at detecting hate speech and whether hate speech has predominantly negative sentiment. To do so, we use the Sentiment140 dataset to train and evaluate our sentiment models. We then apply the sentiment models to the Hate Speech and Offensive Language dataset. The performance of the sentiment models on the hate speech dataset do not provide conclusive results as to whether sentiment models are an effective technique for hate speech detection. That said, the results do indicate that hate speech tends to be associated with negative sentiment. The code for this paper can be found in the following GitHub repository: https://github.com/milevavantuyl/twitter_sentiment_analysis.

## 1 Introduction

The detection of hate speech on social media platforms is a critical task as the uncontrolled propagation of hate speech on these platforms has the potential to hurt individuals, groups, and our society as a whole. However, its detection is complex for several reasons. First, automated identification is challenging because social media posts frequently use poorly written text, hashtags, and emoticons. In addition, the lack of agreement on what constitutes hate speech provides additional challenges [1].

Motivated by the observations that "hate speech and sentiment analysis are closely related" and that "usually negative sentiment pertains to a hate speech message," we propose using sentiment analysis techniques to address the challenge of hate speech detection [2].

Sentiment analysis techniques, used to detect the polarity of a text, are prevalent throughout the field of natural language processing. Additionally, they are becoming increasingly popular within companies as a method to understand their users better with the purpose of providing better services and improving their business. While some new sentiment models are being developed to detect categories such as joy and love, our work will focus on the most common sentiment models that classify texts into one of three categories: positive, negative, or neutral.

## 2 Related Work

There is a large body of research that focuses on the development of machine learning models to address the hate speech detection problem. Some of them make use of lexical characteristics like dictionaries [3] and bag-of-words [4], though they were shown to be incapable of comprehending the context. N-gram-based techniques were also applied, and they produce results that are noticeably superior [5].

Additionally, given the assumption that hate speech and sentiment analysis are intimately connected, several systems use hybrid approaches that incorporate sentiment analysis techniques to address the hate speech detection

task. A multistep technique is used in [6], where the first step consists of a classifier designed to detect negative sentiment and the second step consists of a classifier that looks for evidence of hate speech. Similarly, Liu and Forss create a hybrid model that combines sentiment analysis techniques and topic modeling to identify hate speech [3].

Similar to the research described above, we explore machine learning approaches to address the task of hate speech detection. However, our work focuses on understanding the benefit that sentiment analysis techniques alone can bring to solving the hate speech detection problem.

## 3 Methodology

### 3. 1 Datasets

We have used two different datasets to perform the main tasks of this project, both obtained from Kaggle [7]. For training and evaluating the performance of the three sentiment classification models we used the Sentiment140 dataset [8]. This dataset from Stanford University contains 1.6 million tweets about products and brands written by users. The tweets are labeled according to the polarity of the text (0-negative, 4-positive).

For assessing the performance of the sentiment models on a hate speech detection task, we use the Hate Speech and Offensive Language dataset [9]. This dataset obtained from Kaggle consists of 24,783 tweets written by users. Each tweet is labeled according to the intention of the message (0-hate speech, 1-offensive language, 2-neutral).

We considered additional hate speech datasets, including [10], which contains texts rated according to toxic behavior. We decided to discard this dataset due to time limitations. However, including this data in future work could allow us to gain more insight about the application of sentiment classification models to hate speech detection.

### 3. 2 Data Cleaning

The initial data cleaning process is common for both datasets and consists of the following steps: 1) case folding, 2) removal of leading and trailing white spaces, 3) removal of non-ASCII characters, 4) removal of numbers, 5) tokenization, which is the process of splitting text into smaller units (words), and 6) lemmatization, which converts all words to their associated lemma form. Additional preprocessing steps specific to the Naive Bayes and LSTM classifiers will be discussed in their respective sections [12].

We also recoded the target labels in each of the datasets. In the sentiment dataset, we coded the negative sentiment observations to ones, and the positive sentiment observations to zeros. In the hate speech dataset, we combined the hate speech and offensive language categories into a single category and coded it as one. The neutral category was recoded as zero.

The sentiment dataset was used to train the Naive Bayes and LSTM classifiers. Thus, it was partitioned into a training set, for training the models; a validation set, for hyperparameter tuning; and a test set, to evaluate the classifiers' performance on unseen observations. We applied a 75-15-10 split for the train, validation, and test sets respectively using the train_test_split function from the scikit-learn package.

### 3. 3 Sentiment Models

#### *3.3.1 Naive Bayes Model*

The first sentiment model we implemented and trained on the Sentiment140 dataset is a Naive Bayes classifier.

The Naive Bayes classifier is effectively utilized in several Natural Language Processing applications, including spam filtering, text classification, recommendation systems and sentiment analysis [11], and. This classifier is a classification algorithm based on Bayes' Theorem. The theorem finds the probability of an event A occurring given that another event B has already occurred. Bayes' theorem can be described mathematically as shown in Figure 1.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

**Figure 1: Mathematical representation of Bayes' theorem**

Naive Bayes assumes independence among the features. This implies that the contribution of a particular feature in determining the probability of a class for a given observation is unrelated to the rest of the predictors. Let's say we have the features pet, whiskers, small, and claws, and the observation is classified as a cat. All of those features will contribute independently to the probability of the observation belonging to the class cat, regardless of the potential relationship that may exist between those features describing the animal.

While the independence assumption may sometimes be unrealistic, the Naive Bayes classifier is a simple, fast algorithm that performs very well on large datasets.

For this project, we implemented a Multinomial Naive Bayes (MNB) classifier. In MNB, a document $D$ is seen as a vocabulary-sized feature vector $v$, with each element representing the number of terms in the document $D$. This vector $v$ then follows a multinomial distribution.

In our approach we remove the stop words and apply a TF-IDF (Term Frequency-Inverse Document Frequency) transformation to the data before passing it to the MNB model. With this transformation we convert the collection of raw documents to a matrix of TF-IDF features. TF-IDF is a measure used to quantify the importance of a word in a document among a collection of documents. Thus, after applying this transformation to our data we no longer consider raw word counts.

The utilized Multinomial Naive Bayes classifier is available in the scikit-learn package. The model includes a smoothing parameter, alpha, whose optimal value has been selected via a random search and 5-fold cross validation as provided by the RandomizedSearchCV function in scikit-learn. This smoothing parameter helps address the issue of zero probability in Naive Bayes algorithm. The zero frequency problem occurs when the algorithm cannot find an instance of a class label associated with a certain attribute value.

### *3.3.2 Long Short-Term Memory Model*

In addition to the Naive Bayes model, we implemented and trained a Long Short-Term Memory (LSTM) Recurrent Neural Network on the Sentiment140 dataset. Unlike the Naive Bayes model, which uses a bag-of-words representation, the LSTM model treats tweets as a sequence of words. This approach, thus, allows the model to consider word order and word context when performing the sentiment classification task.

The Sentiment140 dataset had to be cleaned and preprocessed in advance of training the LSTM sentiment classification model. While the cleaning process was the same as that used for Naive Bayes, the preprocessing for LSTM required us to represent the text of each tweet as a sequence. Unlike Naive Bayes, we did not remove stop words or apply the TF-IDF transformation.

Figure 2 depicts the architecture of the first LSTM model we built, which consisted of three layers: an embedding, bidirectional LSTM, and dense layer. In this iteration, we allowed the model to learn fifty dimensional word embeddings during the training process. This model architecture, however, was susceptible to overfitting.

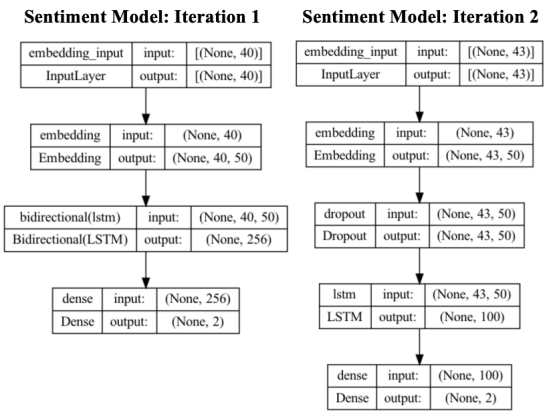


Figure 2: Architectures for the LSTM Sentiment Model

To address the issue of overfitting observed in our first model, we proposed a second architecture that is depicted in Figure 2 and Figure 3. In this model, we added a dropout layer and also converted the bidirectional LSTM layer to a unidirectional LSTM layer. Additionally, we did not allow the model to

train the 25 million parameters associated with the embedding layer. Instead, we used the pre-trained Twitter word vectors from GLoVE that are based on 2 billion tweets and 27 billion tokens [13]. This second architecture addressed the issues of overfitting observed in the first model.

For both architectures, we trained the models on the training set and evaluated them on the validation set. Additionally, we used an adam optimizer, categorical cross entropy loss, and accuracy as the primary metric. The models were built using the Keras API and were each trained for 5 epochs.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 43, 50)            25050900

dropout (Dropout)            (None, 43, 50)            0

lstm (LSTM)                  (None, 100)               60400

dense (Dense)                (None, 2)                 202

=================================================================
Total params: 25,111,502
Trainable params: 60,602
Non-trainable params: 25,050,900
```

**Figure 3: Summary of Final LSTM Sentiment Model**

### *3.3.3 VADER*

VADER, Valence Aware Dictionary for Sentiment Reasoning, is the final sentiment model we considered. We chose this model as it was developed with the goal of sentiment analysis and social media texts in mind.

Unlike the Naive Bayes and LSTM models, VADER is a rule-based model that requires no training data. Instead, it relies on both a sentiment lexicon, a list of words with their associated sentiment polarity, and a set of rules devised based on the grammar and syntax of the texts [14]. Some of the rules consider punctuation, words and adverbs that indicate the intensity of the sentiment, and the use of words like "but" that signal a transition from one sentiment to another within the text.

Unlike the process for the Naive Bayes and LSTM models, VADER did not need to be trained on the Sentiment140 dataset prior to prediction. Thus, the only steps we took involved cleaning the data as described in Section 3.2. The VADER model and Sentiment140 dataset were then prepared for the evaluation phase.

## 4 Results and Analysis

The results and analysis of our models occur in two stages: 1) evaluate the performance of the sentiment models on the Sentiment140 dataset, and 2) obtain the performance of the sentiment models on the Hate Speech and Offensive Language dataset. In both cases, we use the F1-score, accuracy, and precision as our metrics.

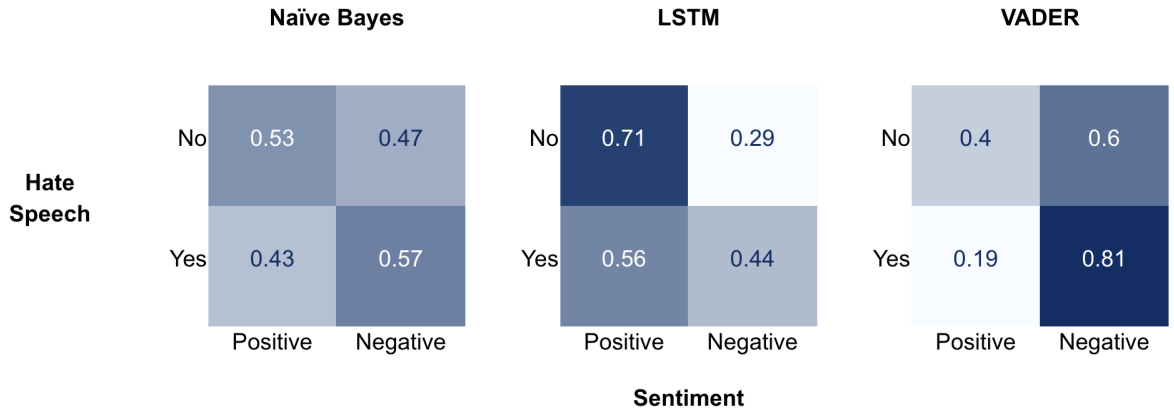


**Figure 4: Results of each sentiment model (Naive Bayes, LSTM, VADER) on the hate speech detection task.**

### 4. 1 Sentiment Analysis

We first want to determine how well the three sentiment models (Naive Bayes, LSTM, and VADER) perform when classifying the sentiment of unseen tweets. To do this, we evaluate our sentiment models on the withheld Sentiment140 test set.

The results are shown in Table 1, which indicates that the LSTM model (F1: 0.789, Accuracy: 0.788) performs best, though it is more computationally intensive to train than the Naive Bayes and VADER models. Then, VADER performs the next best on the sentiment detection task and Naive Bayes performs the worst with results only slightly better than a random guess.

| | Naive Bayes | LSTM | VADER |
|---|---|---|---|
| F1 | 0.581 | **0.789** | 0.661 |
| Accuracy | 0.578 | **0.788** | 0.653 |
| Recall | 0.586 | **0.792** | 0.679 |

**Table 1: Performance on the Sentiment140 Dataset**

### 4. 2 Hate Speech Detection

After evaluating the performance of the sentiment models trained and evaluated on the Sentiment140 dataset, we set out to answer two questions: 1) are sentiment analysis techniques an effective approach for identifying hate speech, and 2) is there a relationship between hate speech and negative sentiment. To answer these questions, we applied the sentiment models described in Section 3.3 to the Hate Speech and Offensive Language dataset.

#### *4.2.1 Effectiveness of Sentiment Models for Hate Speech Detection*

To address the first question of whether sentiment analysis techniques can be an effective approach for identifying hate speech, we prioritized the F1-score which is the harmonic mean of precision and recall and performs better than accuracy in the case when we have imbalanced classes.

The confusion matrices in Figure 4 illustrate the performance of each of the models on the Hate Speech and Offensive Language dataset. We have normalized across each row in order to address the imbalance in classes when displaying the results.

We observe mixed results. The Naive Bayes model slightly favors the true positives (hate speech and negative sentiment) and true negatives (not hate speech and positive sentiment) over the false positives (not hate speech and negative sentiment) and false negatives (hate speech and positive sentiment) as we would expect. The LSTM performs well on observations labeled as neither hate speech nor offensive language, while VADER performs well on the observations flagged as hate speech or offensive.

The tabular representation of these results shown in Table 2 also indicates that it is inconclusive as to whether sentiment classification models are an effective approach for detecting hate speech. This is because the F1-scores for both Naive Bayes and LSTM are low, 68.7% and 58.3% respectively. The VADER sentiment model is the only model that performs above 80% (83.8%) on the hate speech detection task.

| | Naive Bayes | LSTM | VADER |
|---|---|---|---|
| F1 | 0.687 | 0.583 | **0.838** |
| Accuracy | 0.565 | 0.482 | **0.740** |
| Recall | 0.572 | 0.435 | **0.810** |

**Table 2: Performance on the Hate Speech and Offensive Language Dataset**

#### *4.2.2 Hate Speech and Negative Sentiment*

To assess if there is a relationship between hate speech and negative polarity, we focus on the recall metric. Recall highlights what percentage of hate speech tweets are thought to have negative sentiment.

Table 2 shows that recall is over 50% except for the LSTM model. Specifically, we observe a recall of 57.2% for Naive Bayes and 81.0% for VADER. Thus, our results indicate that hate

speech leans in the direction of negative sentiment.

## 5 Results and Analysis

In this project, we studied three sentiment models (Naive Bayes, LSTM, and VADER) and evaluated their performances on both a sentiment analysis task using the Sentiment140 dataset and a hate speech detection task using the Hate Speech and Offensive Language dataset. For the sentiment analysis task, the LSTM model performed best with VADER and Naive Bayes performing second and third best respectively. As one future step, we propose further optimizing the Naive Bayes model and exploring additional architectures for the LSTM model in order to improve the performance of these models on the sentiment data.

The performance of these sentiment models when applied to the task of hate speech detection, however, varied. While VADER performed well, both LSTM and Naive Bayes performed poorly. The strong performance of VADER on the hate speech detection task could be attributed to the fact that it is a rule-based model and was not explicitly trained on a sentiment dataset like the Naive Bayes and LSTM models. Additionally, VADER is known to generalize well to different contexts, and by extension may also generalize better than the other models to different tasks [14].

Ultimately, the results we obtained are inconclusive as to whether sentiment analysis techniques are effective for hate speech detection.

In order to draw a more definitive conclusion, we propose evaluating the performance of other sentiment models, including TextBlob, on the hate speech detection task. Additionally, it would be valuable to gain further insight into the features and rules VADER employs during the classification process that enable the VADER model to perform well on both the sentiment classification and hate speech detection tasks.

## 6 References


[1] Sanchez, Marina and Mileva Van Tuyl. “Natural Language Processing: Research Ideas.” 2022.

[2] Schmidt, Anna, and Michael Wiegand. "A survey on hate speech detection using natural language processing." Proceedings of the fifth international workshop on natural language processing for social media. 2017.

[3]Liu, Shuhua, and Thomas Forss. "New classification models for detecting Hate and Violence web content." 2015 7th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3K). Vol. 1. IEEE, 2015.

[4] P. Burnap and M. Williams, “Us and them: identifying cyber hate on
Twitter across multiple protected characteristics”

[5] C. Nobata, J. Tetreault, “Abusive Language Detection in Online User Content”

[6] Dinakar, Karthik and Birago Jones, Common sense reasoning for detection, prevention and mitigation of cyberbullying, https://web.media.mit.edu/~kdinakar/a18-dinakar.pdf

[7] https://www.kaggle.com/

[8]https://www.kaggle.com/datasets/kazanova/sentiment140

[9]https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset

[10]https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data

[11] https://web.stanford.edu/~jurafsky/slp3/4.pdf

[12] Lee, K. C. (202). Sentiment analysis-comparing 3 common approaches: Naive bayes, LSTM, and vader. Medium. Retrieved December 12, 2022, https://towardsdatascience.com/sentiment-analysis-comparing-3-common-approaches-naive-bayes-lstm-and-vader-ab561f834f89.

[13] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

[14] Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014