Hate Speech Detection

Vasile Godja

June 2024

1 Introduction

In today's digital era, social media platforms influence public discourse. While they enable free expression, they also serve as breeding grounds for hate speech, which can lead to severe societal repercussions, including psychological harm, violence, and social discord.

This raises the following question:

1. **How effectively can various machine learning and deep learning models classify hate speech posts?**

Our main goal is to find valuable insights and key metrics that can be used to train various models, which can be further incorporated into tools to prevent the spread of hate speech.

Data

In order to answer these questions, we rely on the **hate_speech18** dataset from Hugging Face's datasets library. This dataset contains text extracted from Stormfront, a white supremacist forum. A random set of forum posts has been sampled from several subforums and split into sentences. Those sentences have been manually labelled as containing hate speech or not, according to certain annotation guidelines [1]. An initial look at the dataset can be seen in Figure 1, which contains the columns text, user_id, subforum_id, num_contexts, and label.

First few rows of the dataset:

	text	user_id	subforum_id	num_contexts	label
0	As of March 13th , 2014 , the booklet had been...	572066	1346	0	0
1	In order to help increase the booklets downloa...	572066	1346	0	0
2	(Simply copy and paste the following text int...	572066	1346	0	0
3	Click below for a FREE download of a colorfull...	572066	1346	0	1
4	Click on the `` DOWNLOAD (7.42 MB) " green ...	572066	1346	0	0

Figure 1: First few rows of the dataset

The dataset has over 10,000 labeled social media posts. The distribution of labels is shown in Figure 2, indicating that the majority of posts are classified as 0, while a smaller number are identified as 1, and even fewer as 2 and 3. The four labels present in the dataset are associated as follows:

0 Posts without hate speech or offensive content.

1 Posts explicitly promoting hate against individuals or groups.

2 Posts with unclear intent or insufficient context.

3 Posts that, when viewed in relation to others, may be considered hate speech.

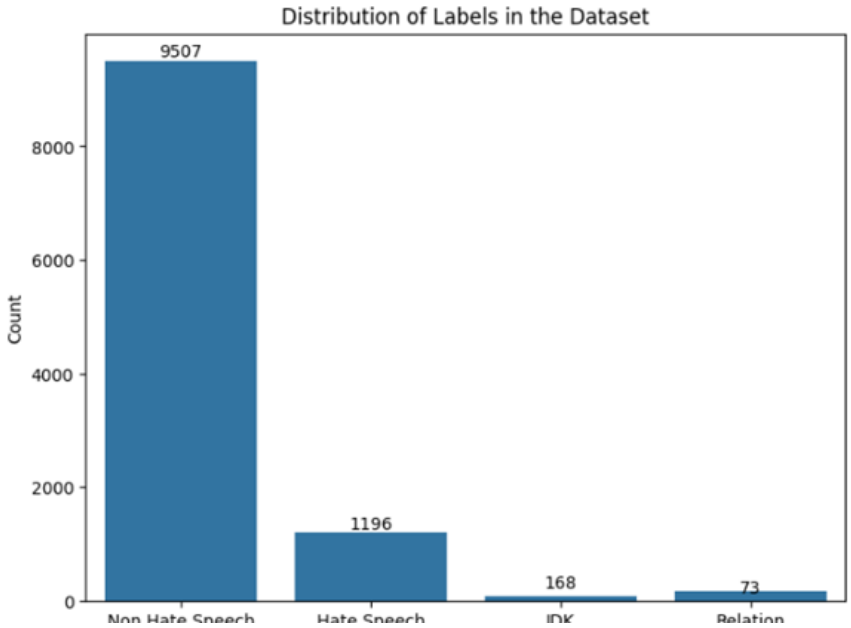

Figure 2: Distribution of Labels in the Dataset

Data Processing

Prior to analysis and model training, we performed several preprocessing steps:

1. Dropped unnecessary columns (user_id, subforum_id, num_contexts).
2. Filtered out labels 2 and 3 due to their representation of ambiguous content, which diverges from our main focus.
3. Special Character Removal: Regular expressions were used to remove characters outside the alphanumeric range (a-z, A-Z); whitespace (\s) and newline characters (\n) were replaced with spaces.
4. All text was converted to lowercase to ensure case-insensitive processing, avoiding treating "Good" and "good" as different words, simplifying the analysis.
5. Stopword Removal: In order to reduce noise and emphasize content-carrying words, stopwords were removed using the Natural Language Toolkit (NLTK) library.
6. Lemmatization: To capture the core meaning of words and improve model generalizability, we used lemmatization, which reduces words to their base form [3].

Our preprocessing pipeline resulted in a dataset of 10,944 samples with an imbalanced class distribution, containing 1,196 samples labeled as hate speech and 9,507 labeled as non-hate speech (Figure 3). This imbalance can hinder model training performance but can be addressed using oversampling or undersampling techniques.

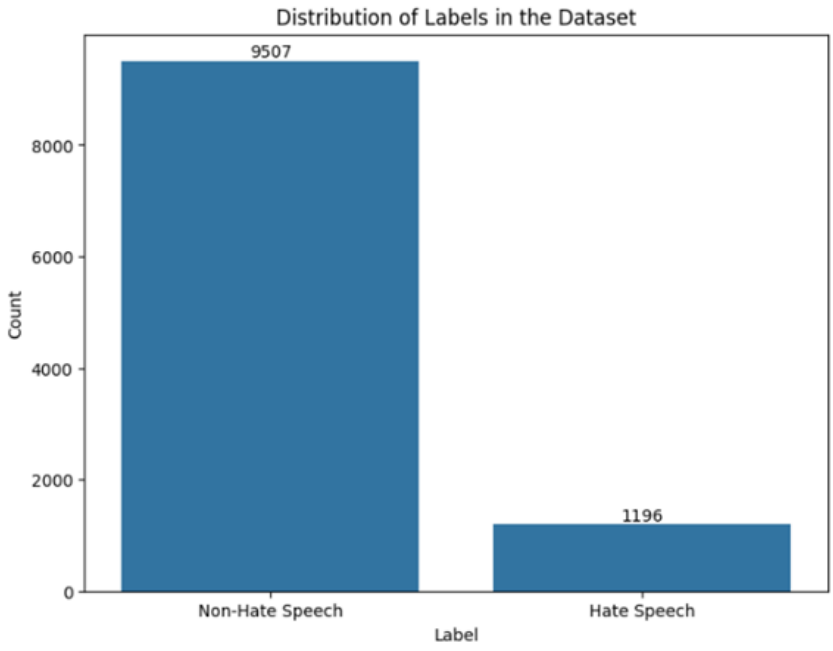

Figure 3: Final distribution of labels in the dataset

Results and Discussions

LinearSVC

Linear Support Vector Classifier is a powerful machine learning algorithm specifically designed for text classification tasks. It belongs to the family of support vector machines (SVMs), which are known for their ability to find optimal decision boundaries that separate data points into distinct classes [4].

Our experiment employs pipelines, which are a convenient way to chain together data processing and model training steps. Each pipeline consists of three components:

1. CountVectorizer: Converts textual data into a numerical representation by counting the occurrences of each word in the documents.

2. TfidfTransformer: Considers both word frequency (TF) and inverse document frequency (IDF) to assign weights to words. Words that appear frequently across all documents receive lower weight, while those specific to hate speech documents gain higher weight, focusing on words that distinguish hate speech.

3. Machine Learning Model: We evaluate three models:

- LinearSVC: A powerful algorithm for text classification that finds an optimal hyperplane to separate the data points (hate speech and non-hate speech) in the feature space created by TF-IDF.
- Random Forest: An ensemble method that combines multiple decision trees, improving model generalizability and reducing overfitting.
- Naive Bayes: A probabilistic classifier that assumes independence between features and calculates the probability of a document belonging to the hate speech class based on individual word probabilities.

1.0.1 LinearSVC Performance

Based on the results shown in Figure 4:

- All models achieve relatively high accuracy scores (above 89%), indicating their ability to classify hate speech and non-hate speech effectively.
- LinearSVC exhibits the best overall performance across most metrics. It achieves a balance between precision (correctly classifying hate speech) and recall (not missing true hate speech).
- Random Forest closely follows LinearSVC in accuracy but shows lower recall, suggesting it might miss some hate speech instances.
- Naive Bayes has the lowest performance, particularly in precision. While it seems to have a high recall, a significant portion might be false positives (incorrectly classifying non-hate speech as hate speech). This points towards potential overfitting and the need for further hyperparameter tuning for Naive Bayes.

	Model	Accuracy	Precision	Recall	F1-Score
0	LinearSVC	0.9019	0.7594	0.6259	0.6615
1	Random Forest	0.8963	0.7793	0.5392	0.5466
2	Naive Bayes	0.8919	0.4460	0.5000	0.4714

Figure 4: Model Performance Comparison

SimpleRNN, LSTM, and Bidirectional SimpleRNN

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data [2]. In our case, we use some variants of RNNs:

1. SimpleRNN: A basic RNN architecture that processes sequences one element (word) at a time. Each cell in a SimpleRNN takes the current input and the previous cell's output as inputs, allowing it to capture some short-term context within the sequence.

2. LSTM (Long Short-Term Memory): A more sophisticated RNN architecture specifically designed to address the limitations of SimpleRNNs. LSTMs introduce memory cells with gates (input, output, forget) that control the flow of information:

    - Input gate: Regulates the flow of new information into the cell.
    - Output gate: Determines what information is passed on to the next cell.
    - Forget gate: Controls what information from the previous cell is retained.

Neural Network Performance

The performance evaluation of three neural network models—SimpleRNN, LSTM, and Bidirectional SimpleRNN—highlights the nuanced differences in handling sequential data tasks (Figure 5). The LSTM model outperformed the others, achieving the highest accuracy (87.29%), precision (65.58%), recall (62.99%), and F1-score (64.10%). These metrics suggest its superior capability in managing long-term dependencies and minimizing both false positives and false negatives. The ROC-AUC score (62.99%) further corroborates its proficiency in class discrimination. In contrast, the SimpleRNN model, with an accuracy of 86.55% and lower precision and recall, indicates limitations in capturing complex patterns, likely due to the vanishing gradient problem. The Bidirectional SimpleRNN, while incorporating future context, showed a modest improvement over SimpleRNN but did not surpass the LSTM in overall performance. This comparison underscores the efficacy of LSTM architectures in applications requiring robust handling of sequential dependencies.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
SimpleRNN	0.8655	0.5914	0.5535	0.5633	0.5535
LSTM	0.8729	0.6558	0.6299	0.6410	0.6299
Bidirectional SimpleRNN	0.8708	0.6360	0.5983	0.6121	0.5983

Figure 5: Neural Network Model Performance

DistilBERT

DistilBERT is a pre-trained transformer model developed by Google AI [5]. It's a smaller and faster version of the original BERT model, making it ideal for tasks where computational resources are limited. Despite its reduced size, DistilBERT

retains a significant amount of the language understanding capabilities of its larger counterpart.

Our experiment utilizes the transformers library to integrate DistilBERT into our hate speech detection pipeline. Here's a breakdown of the key steps:

1. **Tokenization**: DistilBERT operates on sequences of tokens, which are smaller units of text (words or sub-words). We employ the DistilBertTokenizer class to convert our preprocessed text data into numerical token sequences.

2. **Padding and Truncation**: Since DistilBERT requires sequences of a fixed length, we pad shorter sequences with special tokens and truncate longer sequences to ensure all input data conforms to the model's requirements.

3. **Model Training**: The pre-trained DistilBERT model is fine-tuned on our labeled hate speech dataset. During fine-tuning, the model parameters are adjusted to specialize in classifying hate speech content.

4. **Evaluation**: Once trained, the DistilBERT model is evaluated on a separate test dataset to assess its performance in generalizing to unseen hate speech examples. Metrics like accuracy, precision, recall, and F1-score are used to gauge the model's effectiveness.

The DistilBERT model achieved an accuracy of 0.8828 on the training data during the final epoch. This indicates the model learned to classify hate speech content from the training examples. Furthermore, the evaluation on a separate test set yielded an accuracy of 0.8919 but we need to explore additional metrics beyond accuracy.

Conclusion

Overall, the results demonstrate the potential of both traditional and deep learning models for hate speech detection. However, careful consideration of class imbalance and selection of appropriate metrics are necessary for robust evaluation. Our research partially answers the initial question by highlighting the effectiveness of various models. Further investigation into hyperparameter tuning and addressing class imbalance could lead to even better performance.

References

[1] Ona de Gibert, Naiara Perez, Aitor García-Pablos, and Montse Cuadros. Hate Speech Dataset from a White Supremacy Forum. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 11–20, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[2] IBM. Recurrent neural networks, 2023.

[3] IBM. Stemming and lemmatization, 2023.

[4] Scikit learn Developers. sklearn.svm.linearsvc, 2023.

[5] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.