Hate Speech Detection Model

Raghav Upadhyay
raghavupadhyay@arizona.edu
Student ID: 23936430

GitHub Repository

Abstract

With the rise of social media, hate speech and abusive language have become serious problems. This project focuses on building a model to identify and classify hate text online. I used a dataset of 56,745 text samples, which I cleaned and prepared for analysis. I then trained a neural network model, achieving an accuracy of 93.51% when tested. My results show that machine learning can help tackle online hate speech, making social media platforms safer for everyone.

1 Introduction

The internet has transformed the way we communicate, allowing people to connect globally. However, this has also led to the spread of hate speech and abusive language, which can have harmful effects. Hate speech includes any form of communication that promotes violence or discrimination against a particular group based on attributes like race, religion, gender, or sexual orientation.

As social media platforms grow, managing hate speech becomes challenging. Manual monitoring is insufficient due to the large amount of content generated daily. Therefore, there is a need for automated systems that can identify hateful text.

This project aims to develop a machine-learning model to detect hate speech in text. Using a dataset of 56,745 text samples, I cleaned and preprocessed the data to prepare it for analysis. The goal is to create a model that helps identify hate speech, making online platforms safer and more respectful for everyone.

2 Engines

The process of building the hate speech detection model involved several steps: data collection, preprocessing, model development, and evaluation.

2.1 Data Collection

The dataset for this project was obtained from Kaggle and contains about 56,745 text samples labeled as either hate speech or non-hate speech. It covers a range of topics and types of hate speech, providing a diverse set of examples for training.

The data is split into two parts: an imbalanced dataset with 31,962 samples focused on hate speech and a raw dataset with 24,783 samples that includes hate speech, abusive language, and non-hate speech.

To prepare the raw dataset, I removed unnecessary columns and unified the labeling system for consistency. After adjusting the labels, the two datasets were combined to create a complete dataset for training the hate speech detection model.

2.2 Data Preprocessing

Proper preprocessing of the dataset was crucial for model performance. The following steps were used:

- **Data Cleaning:** The text was converted to lowercase to maintain consistency. URLs, HTML tags, punctuation, and numbers were removed. Stopwords—common words like "and," "the," and "is" were also removed. Finally, the remaining words were checked to reduce them to their base forms, helping the model focus on the core meanings of words.

- **Tokenization:** After cleaning the text, it was split into individual words or tokens. This process allows the model to process the text, which is essential for understanding the context and meaning of the words.

- **Padding:** Since the input text varied in length, padding was applied to ensure that all input rows had the same number of tokens. This step is necessary because the model requires

uniform input sizes for effective training and evaluation.

2.3 Model Architecture

The architecture used for this project is based on a Recurrent Neural Network (RNN), a type of model that is particularly effective for sequential data like text. The architecture includes:

- **Embedding Layer:** This layer converts words into dense vector representations, allowing the model to capture the semantic meaning of words within a context. The embeddings are fine-tuned during training to capture nuances in hate speech.
- **RNN Layer:** The core of the model processes sequences of words and captures patterns in the text. The RNN architecture, particularly Long Short-Term Memory (LSTM) units, helps capture both short-term and long-term dependencies between words, which is essential for detecting hate speech spread across a sentence.
- **Dense Layer:** A fully connected layer processes the output of the RNN and makes predictions. A soft-max activation function is applied to output the probability distribution over the classes (hate speech or non-hate speech).

2.4 Training

The model was trained using cross-entropy loss, which is suitable for classification tasks. The Adam optimizer was employed for faster convergence.

- **Batch Size:** Training was done in mini-batches with a batch size of 32 to balance memory efficiency and model convergence.
- **Dropout:** A dropout layer was included to prevent overfitting, ensuring the model generalizes well to unseen data.
- **Learning Rate:** The learning rate was fine-tuned during training, starting with an initial rate of 0.001 and adjusted as training progressed.

2.5 Evaluation

After training, the model was evaluated using a hold-out test set to measure its performance. Several metrics were used for evaluation:

- **Accuracy:** The model achieved an accuracy of approximately 93.51%, meaning it correctly classified 93.51% of the test samples.
- **Precision:** The precision for hate speech detection was around 0.88, indicating that 88% of the posts classified as hate speech were correctly identified.
- **Recall:** The recall was 0.85, meaning the model correctly identified 85% of the actual hate speech posts in the dataset.
- **F1-Score:** The F1-score, a balance between precision and recall, was approximately 0.87, suggesting that the model was well-balanced in its classification abilities.

2.6 Results

The model successfully identified hate speech with high accuracy, making it an effective tool for automating content moderation on social media platforms. The RNN-based approach proved efficient in capturing contextual dependencies between words, which is crucial for detecting nuanced hate speech. Future improvements could involve experimenting with more advanced architectures, such as transformers, to enhance performance further.

3 Discussion

The main goal of this project was to create a model that can accurately find and classify hate speech in social media posts. The results show that the model achieved about 93.51% accuracy, highlighting how effective machine learning can be in addressing the problem of online hate speech.

One of the model's strengths was the data preparation process. Cleaning the text by removing common words (stopwords) and checking the remaining words helped the model focus on the important parts of the text, leading to better results.

Although the model performed well, there is still room for improvement. Expanding the dataset or exploring newer models, such as transformers, could enhance the model further.

It is also important to consider the moral side of using such models. Automated moderation can lead to false positives or unnecessary censorship, so human oversight is necessary to ensure fairness.

4 Limitations

The following are the limitations:

- **Dataset Diversity:** The dataset may not represent all types of hate speech, leading to misclassification.
- **Context Sensitivity:** The model might struggle with understanding nuances like sarcasm or cultural references.
- **Language Variations:** The model was trained on English text, limiting its performance in other languages.
- **Evolving Language:** The model may become outdated if not updated regularly with new data.
- **Ethical Concerns:** Bias in predictions could disproportionately affect certain groups.

5 Conclusion

In this project, I developed a machine learning model to detect and classify hate speech in social media posts. I started with a large dataset of 56,745 samples and went through a careful process of data cleaning and preparation. By using a Recurrent Neural Network (RNN) with LSTM units, my model achieved an accuracy of 93.51

The results demonstrate the potential of machine learning in addressing the growing issue of hate speech online. My approach highlights the importance of thorough data preprocessing and the effectiveness of RNNs in understanding the context of language. However, I recognize that there is still room for improvement, especially in terms of expanding the dataset to include a wider variety of hate speech types and exploring newer model architectures.

Ultimately, this project contributes to ongoing efforts to create safer online environments. As social media continues to grow, tools like this can play a vital role in identifying and mitigating hate speech, ensuring that online spaces remain respectful and inclusive for all users. In future work, I plan to further refine the model and carefully consider the ethical implications of automated content moderation.

References

[1] TensorFlow. (n.d.). *TensorFlow Documentation*. Retrieved from https://www.tensorflow.org/

[2] Bird, S., Klein, E., & Loper, E. (n.d.). *Natural Language Processing with Python*. Retrieved from https://www.nltk.org/book/

[3] Kaggle. (n.d.). *Datasets*. Retrieved from https://www.kaggle.com/datasets

[4] Scikit-learn. (n.d.). *Scikit-learn: Machine Learning in Python*. Retrieved from https://scikit-learn.org/stable/

[5] Chollet, F. (2015). *Keras*. Retrieved from https://keras.io/

[6] Youtube Tutorial(s) for the ideas and troubleshooting errors. Retrieved from https://www.youtube.com/live/9D2n-WpO_88?si=Lpv8Lz_fTW8JGif2

Find the Code

The code for this project is available on GitHub: **https://github.com/raghav-upadhyay2002/Text-Classification**