CSE556: Natural Language Processing
Hate Speech Detection

Aryan Dhull 2021520 | Deepanshu 2021524 | Pranav Aggarwal 2021551 | Prerak Gupta 2021552

Abstract

The rise of social media platforms has revolutionized communication and information dissemination. However, this progress has been accompanied by a concerning surge in hate speech content. This language, characterized by its offensive, hurtful and derogatory nature, targets specific people or groups, fostering a climate of hostility and division. This project aims to address this critical issue by developing a system for automated hate speech detection within social media.

Natural Language Processing (NLP) offers a powerful tool to tackle this challenge. We propose an NLP-based system to identify and flag hateful content before it reaches a wider audience. We aim to utilize the social bias frames dataset for this task. The project aims to create safer social media spaces free from hate speech, while still upholding the principles of free expression. By achieving this, we can foster more civil and productive online discourse for everyone.

1. Introduction

Scrolling through the internet, we often catch comments like "This is because of [some derogatory term for a group or individual]". This kind of language is a prime example of hate speech. Hate speech is any form of communication that attacks a person or group on the basis of their race, religion, ethnicity, gender or disability. It's more than just rude or offensive comments - it's language that spreads negativity and promotes discrimination. It can take many forms, from slurs and insults to jokes or stereotypes that demean entire groups of people. For Instance, recent hate related attacks contributers were found to have a long history of hate comments,tweets across various social media websites suggesting the detection of hate speech could substantially reduce the spread of hateful content.

Problem Statement: The explosion of social media content has made it difficult to monitor and address hate speech, which targets individuals or groups based on race, religion, etc. Traditional methods rely on manual review by humans, a slow and resource-intensive process. Natural Language Processing (NLP) offers a powerful tool to automate hate speech detection. Developing an NLP-based system with high accuracy is crucial to effectively flag hateful content before it spreads, while still allowing for legitimate online discussions.

Motivation: The rise of hate speech online can be discouraging, but the fight against it is gaining momentum with Natural Language Processing (NLP). Existing models on platforms like Twitter and Facebook proactively remove harmful content, even when it's cleverly disguised. YouTube's system combines multiple NLP techniques to filter hate speech in comments. The advancements in NLP motivate further development in hate speech detection. By automating detection and analyzing complex language, NLP can significantly improve the online experience for everyone, fostering a more positive and inclusive digital space.

2. Related Works

Several researchers have explored Natural Language Processing (NLP) techniques to address the growing issue of hate speech detection outlined above. Following are some of them.

2.1. A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media

Mozafari et al in *A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media*[2] introduced a novel transfer learning approach using the pre-trained language model BERT for hate speech detection on social media. The paper proposed new fine-tuning method to adapt for this task, leveraging its ability to capture contextual information within social media content. The model started with the pre-trained BERT base model, the BERT outputs were passed through

nonlinear layers, BiLSTM layer and finally through CNN layer to capture the local patterns in the data. They used two publicly available datasets (Waseem and Hovy Dataset, Davidson et al. Dataset) annotated with racism, sexism, hate or offensive content on Twitter. The results has improved precison and recall compared to other existing models. The paper also highlighted the model's potential to detect biases in data annotation and collection, which could lead to more accuracte hate speech detection models in the future.

2.2. Using Convolutional Neural Networks to Classify Hate-Speech

Sikdar et al in *Using Convolutional Neural Networks to Classify Hate-Speech*[4] presented a system that uses Convolutional Neural Networks (CNNs) to classify tweets into categories like racism, sexism, both, or non-hate speech. The system utilized character 4-gram and word vectors created using word2vec vectors, which were then downsized by max-pooling. The model based on word2vec embeddings outperformed others with a 78.3% F1 score, indicating higher precision than recall in classifying tweets. The model was trained on the English Twitter hate speech dataset created by Waseem(2016). The paper also suggested exploring alternative word embeddings and different network architectures, such as BiLSTM, for improving hate-speech detection.

2.3. A Unified Deep Learning Architecture for Abuse Detection

Founta et al in *A Unified Deep Learning Architecture for Abuse Detection*[1] presented a deep learning architecture for detecting various forms of abusive behaviour online, focusing on Twitter data. They took a holistic approach, considering user and textual properties to detect multiple inter-related abusive behaviours without model tuning for each task. The proposed method significantly outperformed state-of-art methods, showing improvements in AUC between 21 and 45% depending on the dataset. The study highlighted the importance of combining metadata with text for detecting abuse and introduced a training technique focusing on each input separately. The paper addressed the challenge of detecting hate speech, sexism, racism, bullying, sarcasm, and other types of abuse on social media platforms.

2.4. Modified TF-IDF with Machine Learning Classifier for Hate Speech Detection on Twitter

Nandini et al in *Modified TF-IDF with Machine Learning Classifier for Hate Speech Detection on Twitter* [3] proposed a methodology that combines modified TF-IDF (a statistical method for analyzing word importance) with machine learning classifiers. First, data preprocessing was performed, essentially cleaning and formatting raw tweet data for machine learning models. This involved handling missing information, converting categorical data into usable formats, and ensuring all features were on a similar scale. Next, the study employed a modified version of TF-IDF, a statistical technique for analyzing word importance, to extract features from the tweets. This modification aimed to capture the unique characteristics of hate speech content compared to regular text. These extracted features were then fed into machine learning algorithms, such as Decision Trees or Naïve Bayes, to learn patterns and identify hateful content within the tweets.

Finally, the paper evaluated the system's performance using metrics like accuracy to assess its effectiveness in detecting hate speech. This methodology aimed to create a powerful tool for monitoring and mitigating hate speech on Twitter, fostering a safer online environment for users. The study also acknowledged the evolving nature of hate speech, highlighting the need for adaptable techniques in the fight against it.

3. Dataset

For our experiments, we used the social bias frames dataset available on the hugging face. The social bias frames dataset is a valuable collection of posts gathered from various social media platforms, including Twitter, Reddit, and Stromfront. It encompasses a diverse range of online discourse, providing a robust foundation for training and evaluating our hate speech detection model. The dataset contains 147,139 entries, each characterized by 19 features.

3.1. Preprocessing:

For our project's specific needs, we focused on extracting two key features:

- **"post"**: This feature holds the actual text content of the social media post.
- **"offensiveYN"**: This feature provides the corresponding label for the post, indicating its offensive nature. The label can take on three values:
  - **0**: Not offensive
  - **0.5**: Maybe offensive
  - **1**: Offensive

Addressing Ambiguity: The presence of the "maybe offensive" (0.5) label presented a challenge. To ensure a more definitive classification for our model, we opted to convert

all instances labeled ”0.5” to ”1” (offensive). This decision aimed to prioritize the identification of potentially offensive content during training.

Handling Multiple Same Entries: The social bias frames dataset boasts a valuable aspect - annotation by a diverse group of individuals. These annotators varied in age (26-46), gender (55% women, 42% men, 1% non-binary), ethnicity (82% White, 4% Asian, 4% Hispanic, 4% Black), and political preference. This diversity enriches the dataset by incorporating a range of perspectives. However, it also presented a challenge – multiple labels for each post due to the various annotators. To address this challenge, we employed a voting mechanism. For each post, we assigned the label that received the most votes from the annotators. This approach streamlined the data by consolidating the multiple labels into a single, representative label for each post.

3.2. Final Dataset:

Through the preprocessing steps outlined above, the initial dataset of 147,139 entries was reduced to a more manageable size of 44,756 unique entries. Special characters and emojis present in the entries were removed or replaced for our task. This final dataset was further divided for training, validation, and testing purposes. The training set comprised 35,424 entries, while both the validation and testing sets contained 4,666 entries each. This division ensures a robust training process for our model while reserving data for validation and evaluation of its performance.

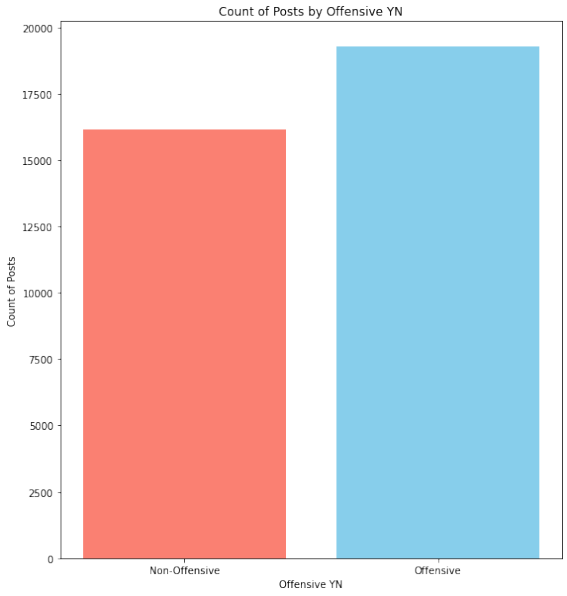

Figure 1: Number of posts of each label in training

4. Methodologies

To achieve our goal of classifying social media posts as offensive or not, we explored various machine learning (ML), deep learning (DL), and natural language processing (NLP) techniques. This included classic ML classifiers like Random Forest, ADABoost, and SVM, alongside deep learning approaches like Multi-layer Perceptron (MLP). We also investigated specialized NLP techniques such as BERT Encoder, BiLSTM, and CNN. Each model was trained on a dedicated set of 35,424 entries from the preprocessed data. To ensure generalizability and avoid overfitting, we employed a separate validation set of 4,666 entries to evaluate each model's performance. Finally, the best performing model was further assessed on a final testing set of 4,666 entries. This multi-pronged approach allowed us to identify the most effective technique for classifying hate speech within social media posts.

4.1. TF-IDF Vectorizer

The text data from the Train, Val, Test datasets were vectorized using the Term Frequency-inverse Document Frequency (TF-IDF) method. The vectorizer was limited to a maximum of 1500 features, which are selected based on their significance in the dataset. TF-IDF stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus.

- First, we used **Random Forest Classifier** to handle the high-dimensional feature spaces and reduce overfitting instead of Decision Tree Classifier.
- Second, we used **ADABoost (Adaptive Boosting)**, for its ability to combine multiple weak classifiers to form a strong classifier.
- Third, we used **Support Vector Classifier**, for its effectiveness in handling high-dimensional feature spaces and its flexibility in definied non-linear decision boundaries through the use of different kernel functions.

Classification reports were generated for each of the 3 datasets (Train, Val and Test) to help assess the model on unseen data and prevent overfitting. Metrics such as F1 Score, Precision and Recall have been used to gauge the model's performance in predicting each label.

4.2. BERT based Fine-tuning

The text data from the 3 datasets were tokenized using the BERT (Bidirection Encoder Representations from

Transformers) Tokenizer which employs WordPiece tokenization to break down words into subword units. The dataset was constrained to have a maximum length of 50 tokens (the average length was found to be about 30). For sequences shorter than 50 tokens, padding was applied to ensure uniform input size. The pretrained BERT model 'bert-base-uncased' was utilized to generate contextualize word embeddings for the tokenized sequences. This models has been trained on a large-scale corpora to capture rich semantic information from the text.

- **Model 1: BERT Classifier:**

  Following the BERT Embedding layers, three fully connected layers were added to serve as a classifier for the classification task. This involved updating the model's parameters through backpropagation using a suitable optimizer (here, AdamW optimizer) and an appropriate loss function (here, cross-entropy loss).

- **Model 2: Multi-layered CNN:**

  In this model, the embeddings after each hidden layers has been extracted, resulting in a series of embeddings capturing the contextual information at different levels of abstraction. The model then incorporates convolutional layers which learns the spatial features obtained after each hidden layer. ReLU activation function is applied after each convolution operation to introduce non-linearity, followed by max pooling to downsample the feature maps and retain the most salient information.

  Further, after flattening the pooled feature maps from each convolutional layer, the model includes fully connected layers to perform classification. Dropout layers have been employed to mitiage overfitting, and a linear layer with softmax activation is employed to generate logits for binary classification tasks. AdamW optimizer and cross-entropy loss have been used.

- **Model 3: Stacked BiLSTM with Convolutional Layers:**

  In this model as well, the embedding after each hidden layer has been extracted. In this case, before applying the convolutional, activation and pooling layers, we have employed a bidirectional LSTM layer. This helps capturing sequential dependencies in the input embeddings. The BiLSTM layer processes the token embeddings in both forward and backward directions, allowing the model to capture information from both past and future tokens. The model architecture is shown in figure number 2 on page number 6

Finally, for each of the model F1 scores and Accuracy has been calculated to assess its performance.

5. Results and Analysis

5.1. TF-IDF Vectorizer:

The various classifers, we used gave nearly same results with the accuracy around 60% and F1 score around 0.4. The specific results for each classifier are as follows:

CLASSIFIER	ACCURACY	F1-SCORE
Random Forest	0.62	0.40
ADA Boost	0.60	0.48
SVC	0.61	0.43

5.2. BERT based Fine-Tuning:

Model 3 (Stacked BiLSTM with Convolutional Layers) gave best results with F1 score of 0.72 and accuracy of 73%. Model 1 and Model 2 gave nearly same results. Specific results are as follows:

MODEL	ACCURACY	F1-SCORE
Model 1	0.69	0.67
Model 2	0.66	0.66
Model 3	**0.73**	**0.72**

6. Observations

- Among the TF-IDF based models, including Random Forest, ADABoost, and SVC, demonstrate moderate performance in terms of accuracy and F1 score, with ADAboost achieving the highest F1 score.
- The BERT-based models consistently outperform the TF-IDF models across both accuracy and F1 score metrics, highlighting the benefits of using contextualized word embeddings for text classification tasks.
- Among the BERT based models, the BiLSTM CNN architecture achieves the highest accuracy and F1 score, indicating its effectiveness in capturing sequential dependencies in the data as well as the ability of CNNs to capture local patterns.
- Further, the usage of embeddings after each hidden layer helps leverage the importance of semantic information extracted at each layer and hierarchial learning

Overall, the BiLSTM CNN models utilizing BERT embeddings emerges as the best-performing model, showcasing the importance of incorporating deep learning architectures and contextualized embeddings for text classification tasks.

7. Conclusion and Future Work

- In this project, we have explored the effectiveness of different models for hate classification tasks using TF-IDF and BERT embeddings.

- The results emphasize the importance of contextualized embeddings. The inclusion of BiLSTM showcases the benefits of combining deep learning architectures in NLP tasks.

Future work in the direction of this study that can be explored are:

- **Model Optimization** : Further optimization of hyperparameters could potentially improve the performance of the models using techniques such as Grid Search or Bayesian optimization.
- **Transfer Learning** : Transfer Learning approaches can be explored, such as fine-tuning a model trained for sentiment classifaction on hate speech specific data.
- **Domain Adaptation** : Adversarial training or multitask learning can help the model to perform well even on data from different domains or sources.

References

[1] Founta et al. *A Unified Deep Learning Architecture for Abuse Detection* [1].

[2] Mozafari et al. *A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media* [2].

[3] Nandini et al. *Modified TF-IDF with Machine Learning Classifier for Hate Speech Detection on Twitter* [3].

[4] Sikdar et al. *Using Convolutional Neural Networks to Classify Hate-Speech* [4].

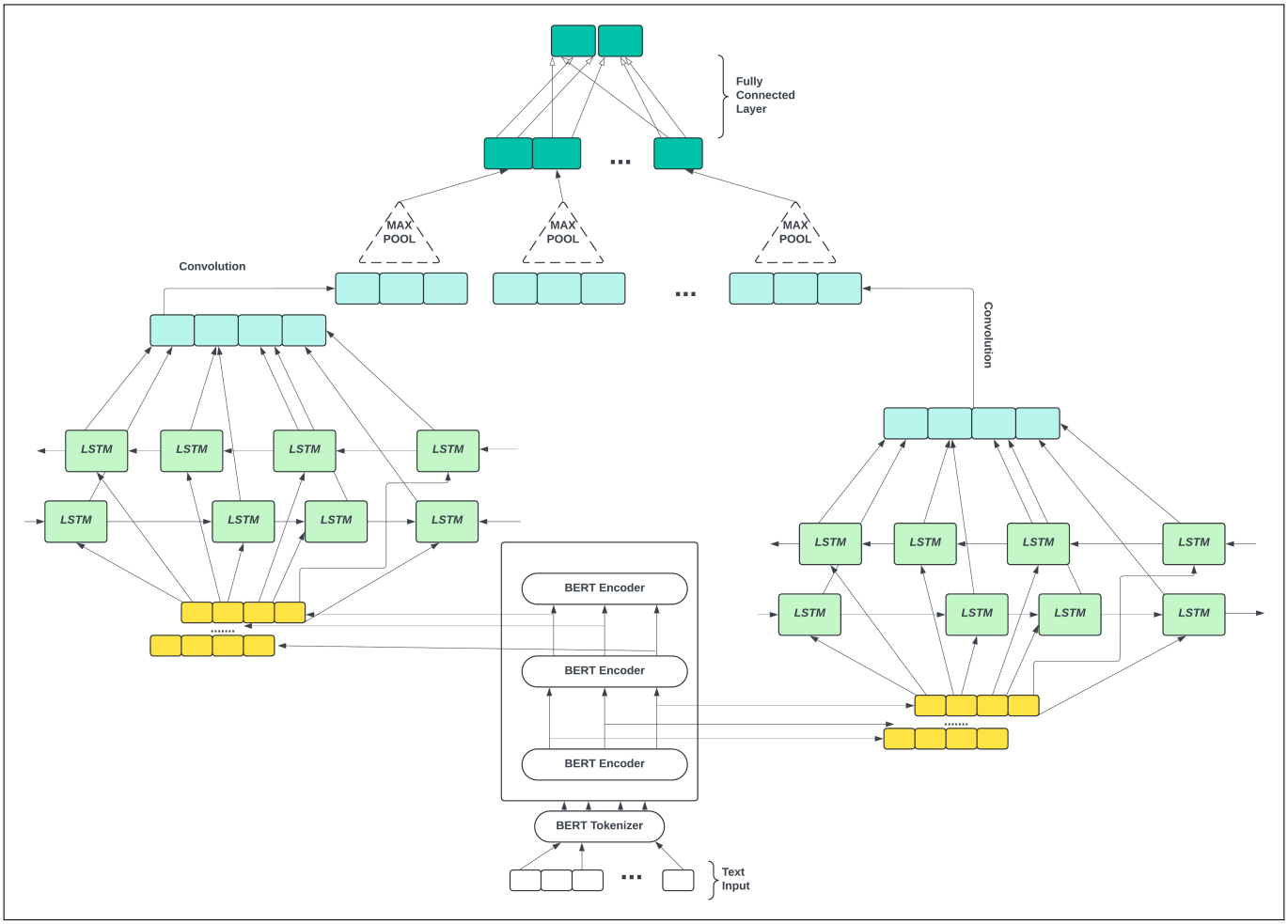

Figure 2: Stacked BiLSTM with Convolutional Layers