Defining Hate: Using Sentimental Analysis via BERT Determine Hate Speech

David Danialy, Aidan Jones, Lawrence Nguyen, Uyen Nguyen, Christian Oh

Abstract— The ability to quickly send messages with a degree of anonymity on the Internet led to a rise of problematic sentiments with minimal accountability. Platforms such as Twitter and Facebook enable methods of reporting and mitigating these sentiments from spreading, but these methods have limitations. With Social Media increasing in users over time, the solution to not only help detect problematic sentiments in the present but also learn about new forms of sentiments in the future involves Machine Learning. Using our model, we can differentiate between purposely problematic statements with malicious intent and undertone from statements that may be written as satire or sarcasm. Using a dataset of 25,000 tweets, our model learns to define tweets as hate speech, offensive language, or neither. We implemented a natural language processing model called the Bidirectional Encoder Representations from Transformers (BERT) to be trained and differentiate the data. Based on our results, we can implement BERT with high accuracy and tweets are correctly categorized via the model.

Index Terms—Machine Learning, Natural Language Processing, Data Mining, Sentimental Analysis, Bidirectional Encoder Representations from Transformers, Detecting Hate Speech

Table Of Contents

1. Introduction

The rise of social media allows information to be spread across the Internet in real-time. From insightful articles to simplified messages like Tweets, text can be sent, read, and shared to multiple people with minimal effort from both sender and receiver. Information and its content, however, varies greatly and includes very problematic statements that can often go unnoticed or without proper accountability established. One solution to ensure negative content such as hate speech, offensive language, and similar topics are noticed and dealt with. This approach involves using techniques in Machine Learning to improve our ability to find inappropriate statements over time.

1.1. The Problem

The growing complexity of the Internet and the freedoms it allows has resulted in an increased tendency for its users to post toxic and hateful content under anonymous names. Some domains are okay with this, but many are trying to combat it through various means. The most common of such means is to implement a "reporting" system, where other users can manually flag posts, either for the system to delete automatically or for a moderator to delete personally. In either case, there are significant flaws in the method of reporting. Mass reporting on non-rule-breaking content subverts the purpose of reporting content that is intended to be reported. Scalability is a major factor in social media, as websites and applications must be able to consider its growing number of users, making manpower to moderate networks very impossible.

1.2. The Problem Significance

The problem of hate speech proliferation is important in most online communities, or anywhere that people can express their opinions. Under the circumstances of the domain being owned by a private company, usually the ability to post toxic or hateful content is restricted. Thus, to enforce this restriction, the creation of a scalable (possibly to an enormous level) solution is necessary.

1.3. The Solution

With the growing amount of online media, automatic classification is the only way to organize the sheer amount of content. The complex nature of language guides us to machine learning. Satire, sarcasm, or other outliers could lead to conventional programming algorithms yielding poor results in accuracy. In addition, automatic processing is the only way to go through the amount of data the internet offers. There have been many different machine learning algorithms developed for natural language processing. For this project, BERT will be used to identify hate speech, offensive language, or neither in a data set of classified tweets.

2. Data

Our dataset is a compilation of nearly 25,000 unique tweets that have been used to research hate speech detection. The dataset contains tweets that belong to one of three different labels: hate speech, offensive language, or neither. Because of the content of our data set, our team is aware that the data use rhetoric that can be defined as racist, misogynistic, a combination of both, or offensive for different reasons. The team was advised to approach the data with minimal bias in our model.

This dataset also contains the number of users who classified each tweet into one of the three categories, and the total number of people who

classified the tweet. The dataset that we are using can be found here.

Before creating any machine learning models, we performed some general analyses of the dataset. For example, it may be useful to understand how many tweets fall into each category of offensive language, hate speech, or neither.

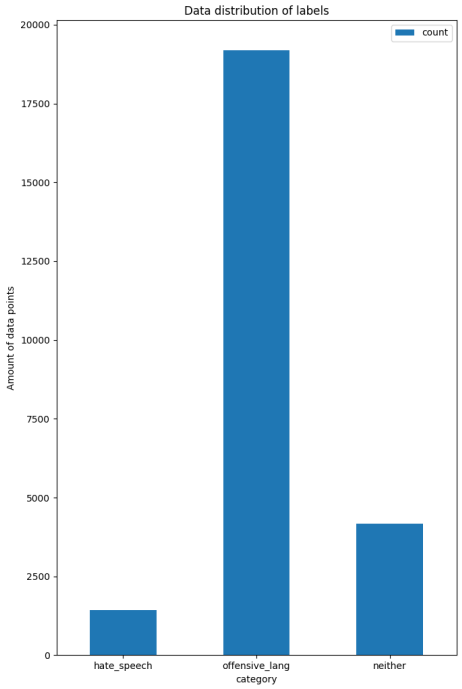

Figure #1 Category distribution of the dataset

As is shown in the figure, the dataset is fairly unevenly distributed. There are close to 19,000 observations of offensive language; however only 1,800 and 4,000 observations of hate speech or neither, respectively.

Another way to gain insight into the dataset was to try to understand what causes something to be classified as hate speech versus offensive language versus neither. The number of curse words per tweet seemed like a good indicator of the class the tweet belongs to. To analyze this, we plotted the number of curse words per class which can be seen in the following plot.

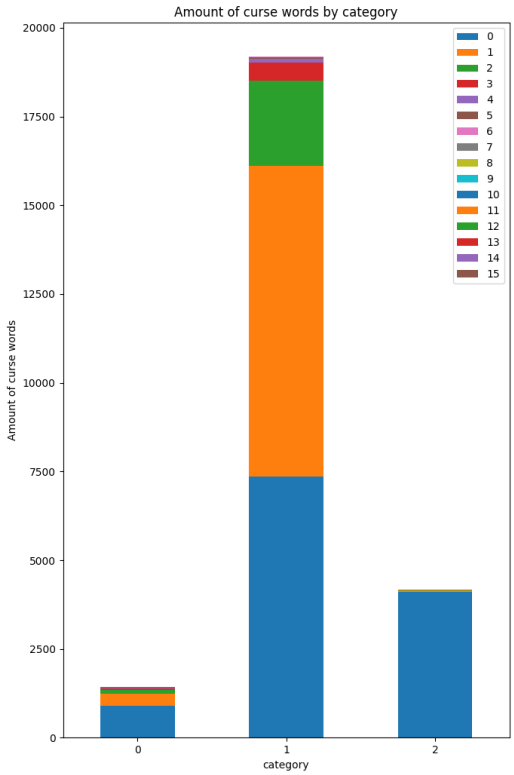

Figure #2 Curse words by tweet class

Unsurprisingly, neither hate speech nor offensive language class is dominated by tweets with 0 curse words. However, what is surprising is that both the offensive language class and the hate speech class have large quantities of tweets with no curse words. Even more surprising was that more than 50% of tweets classified as hate speech have no curse words. Thus it can be concluded that the amount of curse words in a tweet is not a good indicator of whether that tweet is offensive or hate speech.

2.1. Data Sourcing

This dataset was sourced from Kaggle, a Google subsidiary that serves as an online hub for machine learning and data science enthusiasts. On Kaggle, people may publicly post and

download datasets, join competitions and work with others to solve different problems. The dataset used in this project was posted by Andriy Samoshyn in mid-June of last year.

2.2. Data Preprocessing

The raw data contains columns that are unnecessary for the model we are trying to apply. BERT only really needs two columns to function: the text and the label. Therefore, the count columns could be dropped. We added one additional column that functions as a word label for users to understand which number corresponds to which label. After reducing the number of columns, we cleaned the tweets. This means removing unwanted characters such as those generated by Twitter, special characters, and mentions. The figures below display the difference between the data's text before and after processing.

	tweet
0	!!! RT @mayasolovely: As a woman you shouldn't...
1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...

Figure #3 Tweets before processing

```
0    As a woman you shouldn't complain about clean.
1    boy dats cold tyga dwn bad for cuffin dat hoe.
2    Dawg You ever fuck a bitch and she start to c.
3                            she look like a trani
4    The shit you hear about me might be true or i.
```

Figure#4 Tweets after processing

3. BERT

BERT is a natural language processing model, with the significant difference between it and others being that it can process text bi-directionally using a technique called Masked LM (MLM), whereas other models process text from left to right, right to left, or a combination of both. The usage of MLM is through replacing 15% of the words in a body of text with masked tokens, which the model will try to predict the original values of from the context of the other words in the sentence. The paper detailing its architecture displays results significantly outperforming the previous state-of-the-art.

3.1. The Initial Model

Our initial model is a pre-trained BERT model, provided by Hugging Face's transformers package, that is fine-tuned to meet the specifications of our project. During the fine-tuning, we used training hyperparameters as per with recommendations made in this paper published on BERT. to fine-tune the model, we followed the steps provided by Chris McCormick and Nick Ryan. After the fine-tuning of the initial model, we achieved an accuracy of 0.9 and an average training loss of 0.1.

```
  Average training loss: 0.10
  Training epcoh took: 0:04:54

Running Validation...
  Accuracy: 0.90
  Validation took: 0:00:12

Training complete!
```

Figure #5 Final result of initial training

3.2. Change Hyperparameters

The only things that we will change about the implementation of our model will be the hyperparameters (e.g. batch size, learning rate, and the number of epochs) we used. Batch size controls the number of iterations before the model's parameters get updated. Learning rate controls the magnitude of the weight adjustment. The number of epochs is the number of times the model will change weights.

We are at the maximum recommended number of epochs, so we will try to reduce the learning rate from 3e-5 to 2e-5 and keep everything else the same.

```
  Average training loss: 0.13
  Training epcoh took: 0:05:09

Running Validation...
  Accuracy: 0.91
  Validation took: 0:00:13

Training complete!
```

Figure #6 Final result of the first round of changes

With these changes, we can see that there is a higher accuracy but at the cost of higher average training loss.

3.3. Third Iteration

For the final iteration, we will change the learning rate once again, but this time we will increase it from the initial value to 5e-5.

```
  Average training loss: 0.09
  Training epcoh took: 0:05:08

Running Validation...
  Accuracy: 0.91
  Validation took: 0:00:13

Training complete!
```

Figure #7 Final result of the second round of changes

The higher learning rate results in an overarching improvement from the initial model, which better numbers in both training loss and accuracy.

4. The Results

The third iteration of our BERT model, with the learning rate of 5e-5, turned out to be both the most accurate and have the least average loss. Shown is a table comparing the accuracy and loss of all three iterations.

Learning Rate	Accuracy	Loss
V1: 3e-5	0.9	0.1
V2: 2e-5	0.91	0.13
V3: 5e-5	0.91	0.09

Figure #8 Comparing BERT iterations

5. Conclusion

The trained model has decent accuracy at 91% in detecting hate speech, offensive language, and neither. BERT, or NLP machine learning algorithms, could be used to help classify offensive language and hate speech for people or companies.

5.1. Lessons Learned

We learned a lot of valuable lessons from this project. First of all, we all learned about and got hands-on experience creating BERT models for NLP problems. For most of the members of the group, this was the first time being exposed to any NLP problems. This project was also a good opportunity to improve our soft skills, such as teamwork, communication, and time management. We did an excellent job at pacing ourselves throughout the semester so that we were not rushing at the end of the semester to finish all of the work. And most importantly, we learned that Machine Learning is a versatile tool that can be used to make the world a better place.

Appendix

Data collection
We found our data by browsing Kaggle and choosing an appropriate dataset for hate speech detection.

Variety
No other group in the class is doing hate speech detection NLP as far as we are aware.

Visualization
Charts were added to the report to visualize the distribution of data labels in the dataset.

Presentation skills
Practice runs were done to determine which subjects should be discussed, manage the presentation time, and overall help us remember the key points needed to be addressed in the presentation.

Significance to the real world
As shown in section 1, hate speech detection is very relevant in the current state of the internet.

Data reduction for a quick demo
The model artifact is exported.

Veracity
As stated in the earlier lectures of the class, social media involves a lot of veracity regarding the data it produces. The project handles veracity by using Twitter's data via Kaggle to train the data.

Code walkthrough
The walkthrough will be presented as a part of the presentation.

Report presentation (completeness, format, plagiarism, etc.)
The report is summarized with as little prose as possible. Grammarly is used to find and correct grammatical errors in the report and presentation. Turnitin will verify that the report is original.

Tools usage
Tools used are mentioned in the report and code. Our Git Repository is listed here:
https://github.com/uyen-carolyn/CMPE188-Project

Volume
As stated in section 2, our dataset has nearly 25,000 observations.

Version control
Github's repository has version control.

Discussion and Q&A
There is a Q&A section in the slides in case questions are not asked during the presentation. We will answer any questions that the professor or any other students may have during and after.

Lessons learned
A lessons learned section is included as part of the report's conclusion.

Prospects of winning a competition or having the report published
We sent our results to our interviewee, Thad Hughes, for his comments and critiques on our results.

Velocity
The .py file includes a display capability to process data and view its speed as it is inputted via the Python console.

Innovation
BERT is cutting edge NLP, this model can be deployed to be used for flagging hate speech on a production website.

Evaluation of performance
The results section of the report details how well the model performed.

Teamwork
Work is delegated to optimize efficiency and completion of tasks. Communication is done via Discord and updates are frequent to keep everyone in the loop.

Technical difficulty
The project requires an understanding of BERT, which is a deep learning algorithm. This adds a learning curve that allows us to be challenged and overcome the difficulty involved in learning and implementing the model.

Paired programming practice
Paired programming is done virtually via Discord. The Git repository allows us to upload and update code as progress is made.

Agile Practice
The artifact detailing our schedule is included in our submission.

Grammatical correction and revision
Grammarly is used to correct and revise both the report, significant paper, and presentation.

Demo and elevator pitch
Demo and elevator pitch is uploaded YouTube: https://youtu.be/7cBnNd69w_A