Jigsaw Unintended Bias in Toxicity Classification

Sifat Ul Alam
sifatnabil@gmail.com

24th June, 2021

Abstract

Understanding human language is a difficult challenge for computers. In Machine Learning and Data Science, translating words to machine-understandable numbers and designing algorithms to interpret those has been a very intriguing and popular subject of research. In this report, we will be discussing such a topic where our goal is to understand toxic comments by analyzing a large corpus of texts and design a solution that can classify a toxic comment. Words can have varied meanings based on context, and commonly attacked identities (e.g. "gay") can easily be misinterpreted as toxic even when they aren't. The challenge was to reduce biases towards those identities while identifying a toxic comment. This study examines our perspectives on the challenge and the methods we used to tackle it.

1 Methodology

1.1 Dataset

We were provided with a large corpus of text. The Dataset was divided into train and test sets. The train set had 18,04,874 rows and 75 columns. The text of the individual comment was in `comment_text` column and the toxicity label was in the `target` column. The data also had several additional toxicity subtype attributes. The texts were filled with repetitive characters, special characters, and other unnecessary characters. So our first step was to clean the texts as much as possible.

1.2 Data Preprocessing

Before preprocessing, we counted the number of tokens for every item of the `comment_text` column and plotted a distribution. (see Figure 1). From the distribution we can observe that most of sentences have token lengths between 220 and some tokens have lengths nearly 500. As the text was not clean, we decided to clean the text using some methods that were known to us.

First, we converted each comment to lower case. Then we converted each contraction to its actual word form for better understanding. We then removed the punctuations from the texts and converted each word to its base form using stemming. As the dataset was large, the preprocessing took quite an amount of time. But, after processing the texts we were able to reduce sentence lengths and after counting number of tokens this time we got this (see Figure 2) distribution.

From the current distribution, we can observe that the token lengths have been reduced. After analyzing this distribution, we decided to the maximum length for all our texts to be 100.

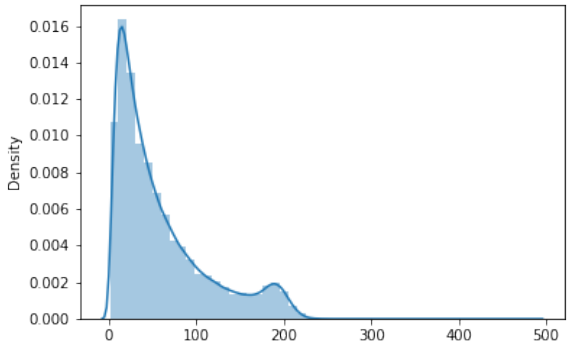

Figure 1: Distribution Before Processing

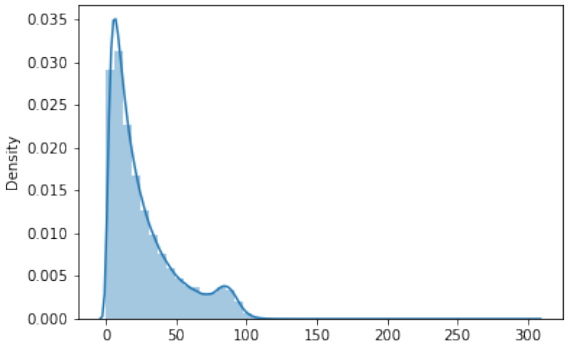

Figure 2: Distribution After Processing

1.3 Preparing the texts for Training

To train a model on our cleaned texts, we had to transform them into machine-readable numbers. As a result, we tokenized all sentences and provided padding to all texts with fewer than 100 words, and we also trimmed words in texts with more than 100 words.

For a more relevant depiction of our texts, we used word embeddings. We generated a word vector matrix using the `GloVe`[1] word embeddings of 300 dimensions. After that the dataset was split into 80% train and 20% validation set.

1.4 Designing and Training the model

Our model consists of an Embedding layer, Bidirectional LSTM layer, an average pooling layer, a dropout followed by a Dense layer. We choose Adam optimizer with a learning rate of 0.001. (Figure 3) is a summary of our model and parameters. The model was trained for 5 epochs on GPU for faster training.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 300)          177044400
_________________________________________________________________
bidirectional (Bidirectional (None, 100, 256)          439296
_________________________________________________________________
global_average_pooling1d (Gl (None, 256)               0
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 257
=================================================================
Total params: 177,483,953
Trainable params: 439,553
Non-trainable params: 177,044,400
```

Figure 3: Structure and Parameters of our model.

2 Result Analysis

After completing training we were able to achieve 0.9498 accuracy on the train set and 0.9444 accuracy on the validation. And upon submission, we achieved a 0.88474 private score on Kaggle[2]. We tweaked certain settings, such as the epoch count and model architecture, but the accuracy remained similar. Because words can have multiple meanings, we feel that how we processed the texts had a significant impact on the model's performance.

3 Conclusion

Understanding and identifying toxic texts can assist us in creating a more positive online community. Because texts can be tough to grasp, having clear and accurate wording is essential for providing any generic solution. The performance of our model was greatly influenced by the quality of our text processing. As a result, there is still potential for improvement and additional trials to be conducted. We used stemming to translate the words to their base form. Another method is to apply lemmatization, which takes longer than stemming but produces a more useful representation of words. A more in-depth examination of the texts would provide us with greater insight into how we could analyze the data in the future. Attention-based methods would also be useful in helping our model understand the context.

References

[1] GloVe. `https://nlp.stanford.edu/projects/glove/`, Jeffrey Pennington, Richard Socher, Christopher D. Manning

[2] Jigsaw Unintended Bias in Toxicity Classification. `https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification`. Jigsaw/Conversation AI.