Auditing Twitter Harassment Detection Models for Racial Bias

Judah Axelrod

May 2021

Content Warning: This notebook presents tweets and keywords which contain language that is sexually violent, vulgar, abusive, racist, and/or threatening. These terms are not always censored as they illustrate important features of the data.

Abstract

Using a large labeled corpus of tweets, I investigate the effectiveness of text classification models in identifying harassing language on Twitter. I discuss in detail the pre-processing steps necessary to build all considered models, the various metrics for assessing model performance, and the potential bias against certain groups (chiefly African American English speakers) that can arise without careful treatment of the data and the research question. I show that while more complicated classifiers involving word embeddings and neural network architectures are better able to capture semantic-syntactic relationships present in the tweets, building an effective model is far from straightforward when we include fairness considerations and constraints. I find that my best-performing model, a GRU with pre-trained GloVe embeddings, achieves a new state-of-the-art on this dataset based on its weighted F1 score. Finally, I use these trained models to make predictions on an unlabeled corpus of tweets that includes the dialect-inferred race of their authors. I show that every one of the classifiers is over twice as likely to predict African American English-aligned tweets as harassing than White-aligned tweets.

1. Introduction

Toxic, hateful, and harassing language on social media is not only objectionable in and of itself, but it can also lead to real racially and ethnically-motivated crime (Williams et al., 2020). With the advances in Natural Language Processing (NLP) techniques, identifying this type of language seems to be an uncontroversial and worthy pursuit. However, neural language models and classification techniques are rife with their own biases. Sun et al. (2019) highlight the gender bias present in NLP methods spanning machine translation to word embeddings to training language models. Gehman et al. (2020) demonstrate that neural language models even when trained on benign corpora can devolve into toxic language. And multiple studies have found that hate speech detection models produce predictions that disproportionately label content from African American English (AAE) speakers, who are overwhelmingly (though not universally) Black, as offensive or objectionable (Davidson et al., 2019; Sap et al., 2019), a result which I expect to replicate here.

I limit this analysis to the consideration of racial bias, and I aim to show that at every stage of the data pipeline, important assumptions and decisions must be considered that are necessary (yet generally insufficient) conditions to ensure that the resulting model does not violate fairness constraints. I intentionally choose a training corpus that has led to comparably poorer model results in the literature in order to both underscore the difficulty of this problem and demonstrate that complex, black box models are not a panacea for answering nuanced and intersectional social questions. In Section 2, I describe the data used in this analysis. In Section 3, I walk through the extensive cleaning and pre-processing steps needed before any modeling can take place. Section 4 deals with the modeling methodology and describes the features, classifiers, and neural network architectures considered. In Section 5, I cover model evaluation metrics and the training and validation process. In Section 6, I discuss the classifier results on the test data. I also apply a subset of models to an unlabeled corpus with inferred user race in to identify possible racial biases embedded in these models. Finally, Section 7 concludes with limitations and next steps.

2. Data

2.1 Corpus Labeled by Harassment

My chosen corpus for this analysis is *A Large Human-Labeled Corpus for Online Harassment Research* constructed by Golbeck et al. (2017). I obtained an anonymized version of the dataset, i.e. with no user information, from the authors to be used solely for non-commercial purposes. The authors hand-label 20,360 tweets as constituting either harassment or non-harassment according to a rigorous methodology outlined in the paper. Along with the actual text of the tweet, this label is the only additional data I have for the corpus. Like most similar datasets, the labels are highly unbalanced, with just over 25% of the tweets being harassing. The data is not a representative sample of all harassing tweets, as the authors specifically search for keywords and hashtags related to anti-black, anti-semitic, Islamophobic, homophobic and misogynistic content. Additionally, while the authors do assign several classes to the data (including "threats", "hateful", and "potentially offensive"), the data I was given access to allows me to formulate a binary classification problem: is a given tweet harassing or non-harassing? Finally, Chakrabarty et al. (2019) find that identical models performed far worse on this dataset when compared with other popular corpus choices (Waseem & Hovy, 2016; Davidson et al., 2017; Wulczyn et al., 2017).

2.2 Corpus Labeled by User Race

After training, validating, and testing models on the Golbeck dataset, I apply a few of the top performers to make predictions on the TwitterAAE corpus developed by Blodgett et al. (2016). TwitterAAE contains about 56 million tweets, which the authors labeled with a lexical detection model that leverages geographic and census data to estimate posterior probabilities of users being African American, Asian, Hispanic, and White English speakers. Following previous work, I use only tweets with predicted probabilities of African American or White over 80%, dropping the Asian and Hispanic categories which are deemed unreliable by the authors. I also take a random subset of just 100,000 tweets, with half predicted as AAE and half predicted as White, for computational reasons. In the end, I have data on the ID and text of the tweet, as well as the inferred racial dialect.

Throughout this analysis, I refer to these datasets as "Golbeck" and "Blodgett" respectively for simplicity.

3. Cleaning and Pre-Processing

I implement a number of pre-processing steps on the Golbeck corpus ("**Data Build.ipynb**"). I remove whitespace and certain punctuation like quotations, as well as assigning special characters to many Twitter-specific strings, such as "RT" (retweet), "@" or "via" (mention of another user), and URLs/hyperlinks added by users. I also do extensive hashtag cleaning because the authors specifically mention searching for harassing tweets using strings such as "#whitepower", "#fuckniggers", among others. By breaking many of these hashtags into multiple words, the word embeddings can hopefully extract additional semantic meaning from them. I should note that no other paper using this data took these extra cleaning steps, and I am hopeful they will add a lot of signal to the modeling process.

Because I will be utilizing word embeddings in the neural language models, I also implement the specific cleaning steps followed by Pennington et al. (2014), the creators of GloVe embeddings. They make available a Ruby cleaning script, and I use a modified version of Kaggle user amackrane's corresponding Python script ("**preprocessingTwitter.py**"). This script applies additional steps such as lowercasing all tokens, finding Twitter-specific regular expressions such as smiley faces, and assigning special characters to strings such as "#" (hashtag), numbers, repeating characters, strings in all caps, and more that would convey additional semantic-syntactic meaning to the GloVe embeddings.

I also found that the data consists of erroneous entries that have multiple tweets in one as well as near- or perfect-duplicate tweets that have contrasting labels. The latter is a serious data issue that would confuse even a perfect model, and so I apply a fuzzy matching methodology ("**Fuzzy Matching.ipynb**") that calculates string similarities in Python between all possible pairs of tweets and removes any IDs that have similarity ratios over 85% and differing labels. These limitations leave me with 19,664 tweets, and I should again note that no other papers I could find that use this data dealt with either of these issues, making me hopeful for improved performance.

Finally, now that I have cleaned strings of tweets to serve as input for the word embeddings, I create a second set of tweets for the Logistic Regression model that will not use embeddings. For this model, I tokenize each tweet, remove most punctuation and common English stopwords (such as "the" or "and"), and apply the Snowball Stemmer from the *nltk* package to reduce words to their base forms. The reason I only take these steps for this second set of tweets is that I don't want to eliminate important semantic and syntactic information from the word embeddings, whereas for Bag-of-Words and tf-idf approaches (more on these below), I only make use of word frequencies, and these approaches help to declutter the tweets and remove noise from the corpus. This step follows directly from Davidson et al. (2019) who use a Logistic Regression with Bag-of-Words as their main classifier.

4. Methodology and Model Construction

4.1 Feature Generation

I use two sets of features in my analysis: the first is for my baseline model which leverages only information about token frequencies, and the second is fed directly into a pre-trained word embedding matrix.

4.1.1 Bag-of-Words and tf-idf

The first method requires generating a frequency distribution of all tokens in the data. Below is a distribution plot for just the 50 most-common tokens:

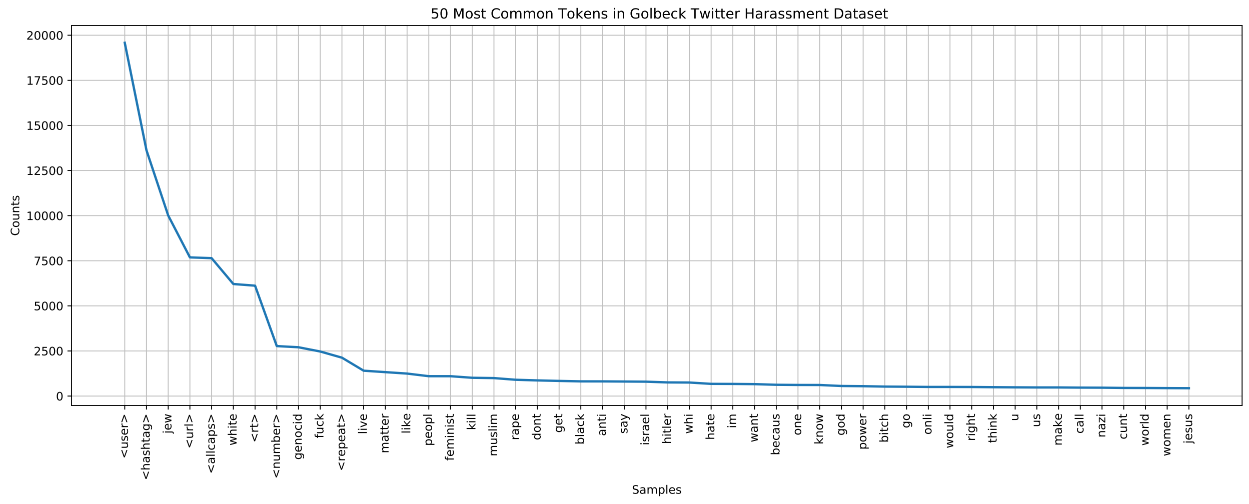

Figure 1: Frequency Distribution

Note that the special characters like retweet and hashtag appear often, as do tokens that are either explicitly abusive or can be used in a polarizing context, such as "jew", "bitch", "hitler", "rape", etc. There are two different ways that I generate my frequency-based features, both commonly used in the literature:

1. Bag-of-Words (BOW) approach: I take the n most common non-stopword tokens (for various values of n) appearing in the dataset and count the number of times each appears in a given tweet. The

simplifying assumption here is that the position of these tokens within the tweet does not matter. This may be reasonable for some tweets that contain such vulgar words that their position is irrelevant; for others, it may make less sense.

2. Tf-idf approach: For each tweet, tf-idf calculates term frequency and inverse document frequency weights according to the formula $w_{t,d} = tf_{t,d} * idf_t$, where $tf_{t,d}$ is the number of times token t appears in tweet d and $idf_t = log(\frac{1+N}{1+df_t}) + 1$ measures the ratio of the total number of tweets to the total number of tweets containing token t.

These features will only capture the presence and frequency of certain words in the tweet, independent of context and position. Intuitively, for the most vile and abusive words, their presence or absence should give the model a good indication as to whether a tweet is harassing; however, context and nuance matters, and this crude model is unlikely to capture everything. In fact, Sap et al. (2019) show that certain unigram tokens like "nigga" that have varying contexts depending on the identity of the speaker can have an outsized effect on models that use Bag-of-Words, so this can be a serious issue if not analyzed carefully. Additionally, I could ordinarily make use of other tweet metadata in the model, but as mentioned above, due to anonymity and ethical concerns, only the content of the tweet and the label are included in this dataset.

4.1.2 Word Embeddings

For the neural language model, I compare two different pre-trained word embeddings. Unlike the above methods which yield sparse vectors that can be as long as dimension $|V|$, the size of the vocabulary, word embeddings are short and dense, generally with d dimensions ranging from 50-1000. Each token has a d-dimensional representation.

1. GloVe, or Global Vectors, (Pennington et al., 2014) leverages global information about the corpus, starting with a matrix X whose ij^{th} entry is the number of times word i appears in the context of word j, so intuitively if $i = strawberry$, X_{ij} is greater for $j = jam$ than for $j = cyclone$. Using this, the authors use the following weighted least squares objective function:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w_j} + b_i + \tilde{b_j} - logX_{ij}),$$

   where w_i and $\tilde{w_j}$ are trainable d-dimensional word vectors with corresponding biases b_i and $\tilde{b_j}$. Additionally, f is a weighting function belonging to the class of functions parametrized as $f(x) = min(1, \frac{x}{x_{max}}^{\alpha})$. f is chosen such that both very rare and very common co-occurrences are not over-weighted. I experimented with both 100D and 200D versions of GloVe and found that model performance was not significantly different, so I use the smaller 100D embeddings here.

2. fastText (Bojanowski et al., 2017) represents each word as itself plus all possible sets of n-grams with added boundary symbols $<$ and $>$. For instance, for $n = 2$, the word *cat* would be represented as {<cat>, <c, ca, at, t>}. Then the authors train a skip-gram embedding on each n-gram, where the embedding for *cat* would be the sum of the learned embeddings for its n-grams: <c, ca, at, t>. At a high level, for each target word (e.g. "strawberry") in the vocabulary, the skip-gram model distinguishes between positive examples of context words appearing in a certain window of the target (e.g. "strawberry *jam*") and negative examples ("strawberry *cyclone*, *table*, *skunk*, etc.") and trains a logistic regression on these examples. For an exhaustive explanation of the skipgram model, see Mikolov et al., 2013 and Jurafsky & Martin, 2020, Chapter 6.

Below are plots of the 2-dimensional representations of the GloVe 100D and fastText 300D embeddings after PCA was applied. By zooming in on just the 50 most-common tokens, we can see that special characters have similar embedding vectors in this 2D space, as do commonly co-occurring tokens like "jews" and "genocide". These offer an indication of the improvements to a model's understanding of English language that embeddings can offer.

Word embeddings are able to capture a lot more information than just word frequencies, including important nuance such as context, syntactic structure, and sequence. However, there is an extensive literature on

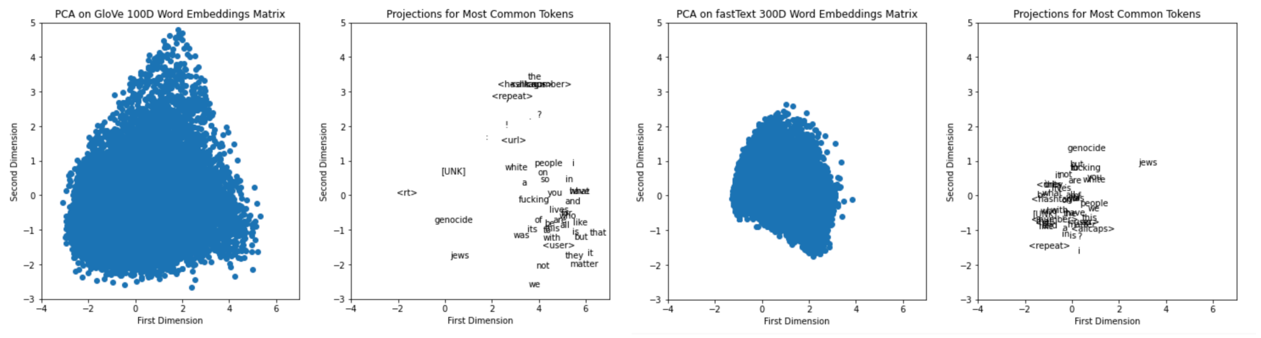

Figure 2: 2D-Representation of Word Embeddings

word embeddings encoding and replicating biases present in English language. Caliskan et al. (2017) derive Word Embedding Association Tests based on the well-known Harvard Implicit Association Tests in social psychology and find that popular pre-trained embeddings such as Word2Vec and GloVe fail these tests the same way that humans do, replicating racial and gender stereotypes. Zhou et al. (2019) find that this result is robust to Spanish (an explicitly gendered language) and Spanish-English bilingual embeddings. Bolukbasi et al. (2016) similarly find that these same out-of-the-box embeddings generate gender-stereotypical analogies such as "man:woman::computer programmer:homemaker". Consequently, I expect the neural language models to reflect not only the biases of the training data, but also the bias present in their embedding layer.

4.2 Classifiers

4.2.1 Regularized Logistic Regression with Bag-of-Words

My baseline model is a Logistic Regression with Bag-of-Words features. As Davidson et al. (2019) reason, it is likely that more sophisticated classifiers can improve predictive power, but we can more reasonably assume that any racial bias here is due to the training data rather than the classifier itself. As explained above, the neural language models utilize word embeddings for which this assumption likely does not hold. To limit overfitting to the training data, I include an L2 regularization term.

Naive Bayes is another commonly-used baseline model in NLP classification tasks. I omit Naive Bayes from this analysis for brevity, but a previous analysis I conducted found that Naive Bayes led to an accuracy on this dataset that was comparable to the Logistic Regression. Unlike Naive Bayes, a generative model which learns from the joint likelihood function of the data $P(X, Y)$, Logistic Regression is a discriminative model that uses the conditional likelihood $P(Y|X)$ to model the actual decision boundary between classes. L2-regularized logistic regression learns a set of weights θ associated with the input features X. The weights are computed through numerical optimization methods (such as stochastic gradient descent) that seek to optimize the equation:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum_{c=1}^{k} P(y^{(c)}|X^{(c)}) - \lambda \sum_{i=1}^{n} \theta_i^2,$$

where c is one of k possible classes and the second term penalizes model complexity according to the L2 norm $\|\theta\|_2 = \sum_{i=1}^{n} \theta_i^2$ (Jurafsky & Martin, 2020, Chapter 5).

4.2.2. Recurrent Neural Network (RNN)

Recurrent neural networks are a family of networks that contain a cycle within their set of connections. Unlike a feedforward neural network, the hidden layer at each time step h_t takes as input not just the weights from the data at that time step, x_t, but also the weights from the previous time step's hidden layer, h_{t-1}. The advantage of this approach is that RNNs can capture the sequential nature of language in a way that other models cannot, processing the tokens in the Golbeck corpus one-by-one in the same way that human beings would read a tweet. I utilize an RNN layer which has the following representation at time $t = 1, ..., T$:

$$
\begin{aligned}
e_t &= E^T x_t \\
h_t &= a(Wh_{t-1} + Ue_t + b) \\
y_T &= ReLU(Vh_T + c)
\end{aligned}
$$

for an input sequence x_t, d-dimensional embedding matrix E, weight matrices U, V, and W, bias vectors b and c, and hidden layer h (Jurafsky & Martin, 2020, Chapter 9; Goodfellow et al., 2016, Chapter 10). Finally, the output of this layer y_T is fed into a dense layer with sigmoid activation, which returns predicted binary class labels. Here is my own (slightly simplified) visualization of the network architecture for a sample tweet "#whitelivesmatter":

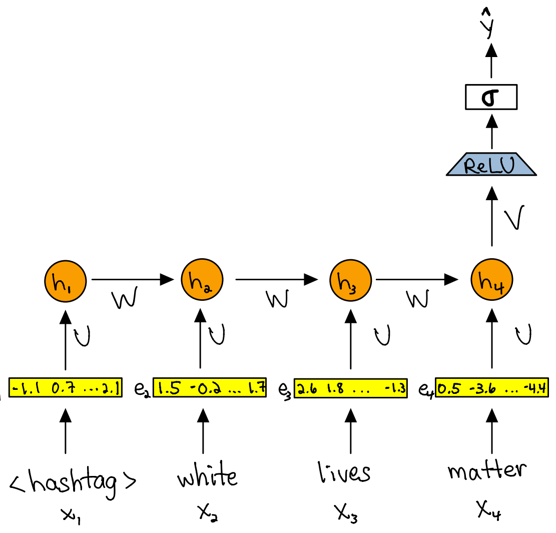

Figure 3: Recurrent Neural Network Model Architecture

I use 128 neurons in my RNN layer, which is the output dimension of y_T for each tweet. However, as we will see, this Recurrent Neural Network is unable to learn any classification of the training data beyond always choosing the majority label of "non-harassing". This result was robust to changing the type of word embedding used, changing the number of neurons, adding dropout, inserting additional Dense layers before the final layer, and even stacking multiple RNN layers. It simply seems that a different neural language model architecture will be necessary to glean more from this difficult corpus.

4.2.3 Gated Recurrent Unit (GRU)

Gated RNN models such as GRU and LSTM (Long short-term memory) address two key shortcomings of the Simple RNN used above:

1. While RNNs are more suitable for sequence modeling tasks such as this one, the contextual information about previous words in the sequence tends to be quite local; information contained in hidden layer h_1 may be mostly lost by the time that, say, layer h_{15} is reached. It is likely that in many tweets, meaningful information is contained in earlier tokens that are very distant from the final token, whose corresponding hidden layer is fed forward in the network.

2. As part of training, RNNs must backpropagate through time, which often leads to "exploding" or "vanishing" gradients. This could very well be why my RNN models are not able to learn the data whatsoever. Gated models allow for the network to forget obsolete information and remember distant information that is still needed (Jurafsky & Martin, 2020, Chapter 9; Goodfellow et al., 2016, Chapter 10).

The GRU model contains a reset gate r whose activity captures short-term dependencies and an update gate u whose activity captures long-term dependencies. Their respective sigmoid activation functions lead to binary outcomes that either block or let through information (thus the “gate” moniker). Ultimately, the gates are then combined as follows:

$$\begin{aligned} e_t &= E^T x_t \\ r_t &= \sigma(W_r h_{t-1} + U_r e_t + b_r) \\ u_t &= \sigma(W_u h_{t-1} + U_u e_t + b_u) \\ \tilde{h_t} &= a(U e_t + W(r_t \otimes h_{t-1}) + b) \\ h_t &= u_t \otimes h_{t-1} + (1 - u_t) \otimes \tilde{h_t} \end{aligned}$$

where we have separate weights W, U and biases b for each gate, and the remaining notation carries forward from the RNN model detailed above. Here is a visualization of my GRU model for the same sample tweet:

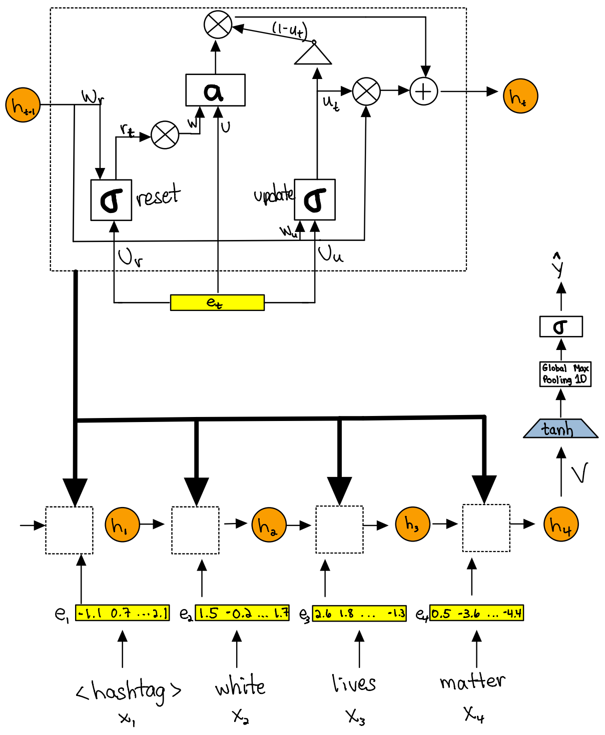

Figure 4: Gated Recurrent Unit Model Architecture

My preferred GRU model uses 256 neurons and includes a 1D Global Max Pooling layer right before the final dense layer. The pooling layer downsamples the input by taking the maximum across the time dimension. This has the effect of reducing complexity, making the representation of the tweet invariant to local translations, i.e. the exact placement of certain tokens in a local neighborhood becomes less important than their presence at all. While more typically associated with Convolutional Neural Networks, the pooling layer here consistently improved the model performance by providing regularization. As we will see, these models suffer from overfitting, and this was even worse when using other means of regularization such as dropout instead of/in conjunction with max pooling. In general, I experimented with other more complex GRU architectures, including stacking an RNN layer, increasing the number of units, etc., but the simpler model consistently achieved better performance, and global max pooling was a feature of all of the better-performing models.

4.2.4 Stacked Bi-Directional GRU with Self-Attention

My most complex model follows directly from the work of Chakrabarty et al. (2019), who stack bi-directional LSTM models with self- and contextual attention on this and other abusive language corpora. I employ a very similar architecture, replacing the LSTM layers with GRU layers that consistently achieved better performance on this data. There are two new features to introduce and motivate in this final neural language model:

1. Bi-directional GRUs represent a possible improvement upon the single GRU layer in that they can capture sequential information in both directions; separate models make a forward pass through the tweet and a backward pass through the tweet and concatenate their results. The authors cite Graves et al. (2013) as the precedent for stacking multiple such bi-directional layers, and they find improvements in doing so.

2. Self-Attention entails the comparison of a certain token x_t with other preceding tokens in the sequence. There are three roles that each have corresponding weight matrices in a self-attention module: the *query* is the current focus of attention in the sequence that is compared with previous tokens in the sequence; the *key* is the preceding token being compared to the query; and a *value* is used to compute the output for the query (Jurafsky & Martin, 2020, Chapter 9). We have:

$$\begin{aligned} e_t &= E^T x_t \\ q_t &= Q e_t \\ k_t &= K e_t \\ v_t &= V e_t \end{aligned}$$

for query q, key k, value v and corresponding weights Q, K, V. To make these comparisons, we use the scalar dot product scores of the query and each key: $score(e_t, e_i) = \frac{q_t \cdot k_i}{\sqrt{d_k}}$ for $i = 1, ..., t-1$, where the normalizing factor contains d_k, the dimension of the embedding. Intuitively, a higher score means two tokens are more similar. We then assign weights α_{it} by taking the softmax of each score and compute an output value $y_t = \sum_{i \leq t} \alpha_{it} v_i$. Like other layers, the output from a self-attention module can simply be passed along in a feedforward network. The diagram below is a stylized version of the stacked bi-directional GRU architecture.

I set the number of units in each GRU to half of the word embedding's dimension, such that each Bi-directional GRU is of dimension equal to the embedding layer. Like the authors, to deal with overfitting, I also include four 25% dropout layers in between the other layers as well as L2 regularization on the final dense layer; however, like the previous models, overfitting will remain an issue.

4.2.5 Other Models Considered

I experimented with some other common neural language models, including Convolutional Neural Networks, Bi-directional RNNs, and LSTM models. Surprisingly, like the simple RNN, the LSTM was unable to learn anything except the majority class. The other models performed worse than the GRU and Attention architectures, so I excluded them from final consideration.

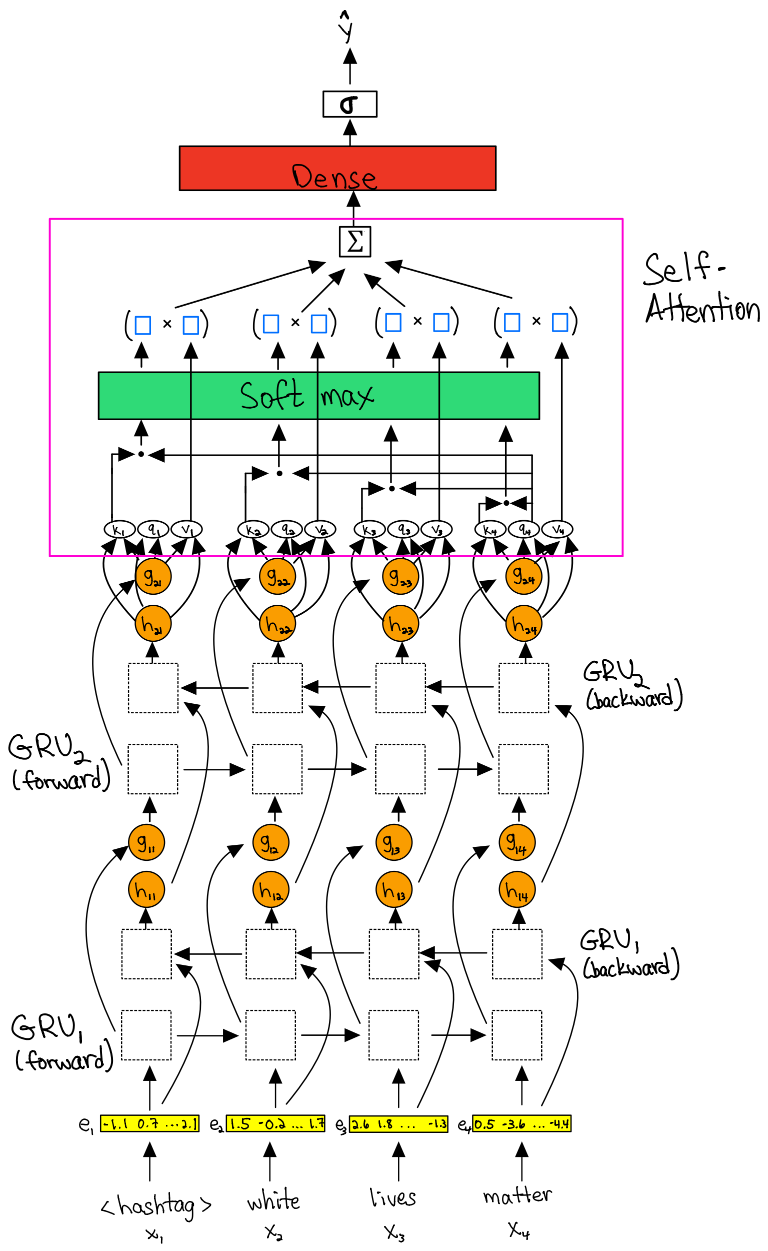

Figure 5: Stacked Bi-directional GRU with Self-Attention Model Architecture

5. Model Evaluation

5.1 Evaluation Metrics

I now walk through several possible evaluation metrics and explain my choice of weighted F1 as the preferred metric. To motivate this section, I introduce some familiar notation for completeness:

- Y, a binary variable corresponding to the true label of the tweet
- $\hat{Y}$, a binary variable corresponding to the classifier's predicted label for the tweet
- True Positives (TP): Tweets with $Y = 1$ and $\hat{Y} = 1$
- True Negatives (TN): Tweets with $Y = 0$ and $\hat{Y} = 0$

- False Positives (FP): Tweets with $Y = 0$ and $\hat{Y} = 1$
- False Negatives (FN): Tweets with $Y = 1$ and $\hat{Y} = 0$

5.1.1 Accuracy

There are many ways to evaluate the performance of a binary classifier, the simplest being its accuracy, the percentage of predictions it correctly makes, or $\frac{(TP+TN)}{Total}$. However, accuracy is a poor choice here for multiple reasons:

1. For this unbalanced dataset, a naive classifier that simply predicted every tweet as non-harassing would have an accuracy of nearly 75%, a deceptively high number that disguises how poor the classifier truly is.
2. Accuracy offers no granularity, so we have no idea if the classifier performed better or worse on harassing or non-harassing tweets, or for specific subgroups in the data.

5.1.2 Precision

Precision captures the proportion of all tweets *predicted* as harassing that are correctly classified, or $\frac{TP}{TP+FP}$. Unlike accuracy, for an unbalanced dataset like this one, precision allows us to see how good the model is at successfully "taking the risk" of predicting a tweet to be harassing (since there are so many fewer actual harassing tweets).

5.1.3 Recall

Recall captures the proportion of all tweets *actually labeled* as harassing that are correctly classified, also known as the True Positive Rate: $\frac{TP}{TP+FN}$. The distinction between precision and recall is that a model could have high precision on the tweets it *does* predict as harassing, but miss out on countless other tweets that it should have predicted as such. In the extreme, if the model predicts exactly one tweet as harassing and does so correctly, the precision would be 100%, but the recall would be virtually 0%. Conversely, if the model is aggressive in labeling most tweets as harassing, it is likely to capture most of the tweets with $Y = 1$ (high recall) at the expense of many false positive results (low precision).

5.1.4 F1 Score and Weighted-F1 Score

To manage this precision-recall tradeoff, the F1 score captures elements of both by taking the harmonic mean of precision and recall: $2 * \frac{precision*recall}{precision+recall}$, which also ranges between 0 and 1. There are multiple ways to assign weights in computing F1. One intuitive way is to compute the F1 score for each label (harassing and non-harassing) and take a simple average. However, my preferred metric, the weighted-F1, computes the F1 score for each class and then takes an average that is weighted by the proportions of each label, the goal being to reduce the impact of highly-imbalanced classes on the evaluation metric. This follows directly from Chakrabarty et al. (2019) who use weighted-F1 as their evaluation metric of choice.

5.1.5 Fairness Metrics

While weighted-F1 is a marked improvement over accuracy as an evaluation metric for this specific problem, having data on a protected class (in this analysis, race) would allow for a more careful and thorough model evaluation. Unfortunately, while I have a corpus with harassment labels and another corpus with inferred race labels, I do not have access to a sufficient sample size with a combination of the two. Hardt et al. (2016) and Barocas et al. (2019, Chapter 2) introduce three additional metrics that incorporate a sensitive attribute or protected class A (such as race, gender, etc.) and a predicted score R, the continuous version of $\hat{Y}$:

1. Independence: The score R is independent of the attribute A, or $P(R|A = a) = P(R|A = b)$. Independence in this context would mean that the predicted probability of a harassing tweet is the same for White English and AAE speakers.

2. Separation (or Equalized Odds): Given the true value Y, the score R is independent of the attribute A, or $P(R|Y, A = a) = P(R|Y, A = b)$. Separation means that the True Positive Rate and False Positive Rate are equal for all groups, or that all groups occupy the same point along the ROC curve.
3. Sufficiency: The true label Y is independent of the attribute A, given the score R, or $P(Y|R, A = a) = P(Y|R, A = b)$. Sufficiency translates to two tweets having the same true probability of being labeled as harassing, *independent of racial dialect*, given that they were both predicted to be harassing.

Like precision and recall, there are trade-offs present among these three metrics that are beyond the scope of this analysis, but having data that would allow me to consider these metrics would open the door to an entire array of bias auditing and mitigation techniques in the fairness literature. For now, I must make do with weighted F1.

5.2 Model Training and Validation

First, I designate 15% of the corpus as the test set. The remaining data is used for training and validation as follows:

5.2.1 Logistic Regression

To determine which word vectorizer (BOW or tf-idf) to use and how many features to consider, I use 5-fold cross validation to perform a grid search, ultimately choosing the model with the highest average weighted-F1 across the five folds. The chosen baseline model - with a weighted F1 cross validation score of 72.02% - is a the Logistic Regression with 2,000 BOW features, meaning that the frequencies of the 2,000 most common tokens are considered as regressors.

5.2.2 Neural Language Models

First, due to label imbalance in the corpus, I assign class weights so that the model will give higher importance to every harassing tweet in the training set when calculating loss. This is a standard step taken for modeling on data with imbalanced classes, and it helps mitigate against the non-harassing tweets coming to dominate the model's learning process.

Second, in order to select the preferred model weights for predicting on the 15% test set, I store validation accuracy at the end of each training epoch and reload the weights that yielded the highest validation accuracy at the end of the training process. Due to computational limitations, I don't use cross validation like I did for the Logistic Regression model. While using validation accuracy follows from Sap et al. (2019), this represents a potentially significant limitation in my methodology, as I do not consider accuracy to be a reliable measure of model performance in this setting. The risk is that if accuracy and weighted F1 diverge from each other, the wrong weights may be chosen and the test results could suffer from high variance. Rerunning the models multiple times, there did not seem to be enormous differences in weighted F1 on the test set, and fortunately the same models seem to achieve the best performance, so I am cautiously optimistic that this represents more of a minor issue than a major one.

Finally, I train in Keras with TensorFlow backend, using the Adam optimizer with an initial learning rate of 0.001, a batch size of 64, and 25 epochs (cutting short the training for the models that only learn the majority class labels). These decisions were made by both consulting the previous literature and experimenting with the response to small changes in the hyperparameters. All of the models run in a reasonable amount of time using a Google Colab GPU. As is standard for a binary classification problem, I optimize over the binary cross-entropy loss function: $L(y, \hat{y}) = -ylog(\hat{y}) - (1 - y)log(1 - \hat{y})$.

Below I plot training and validation losses for the neural language models that learned more than the majority labels, i.e. the non-RNN models. Even with regularization included, the models all suffer from extreme overfitting, as the training losses converge to zero. Additionally, after a few epochs of decrease and fluctuation, the validation loss trends upwards thereafter. The results suggest that an early stopping criterion would have made sense here, but because I track the best validation loss after each epoch, my approach is roughly equivalent (just with longer runtime), and this way I ensure that I don't inadvertently stop before finding my “best epoch” weights. Ultimately, these loss plots look less than ideal, but I found that any further regularization led to poorer weighted F1 scores or even model degeneracy to just learning the majority labels. These diagnostic plots demonstrate well the difficulty of the task and this dataset in particular.

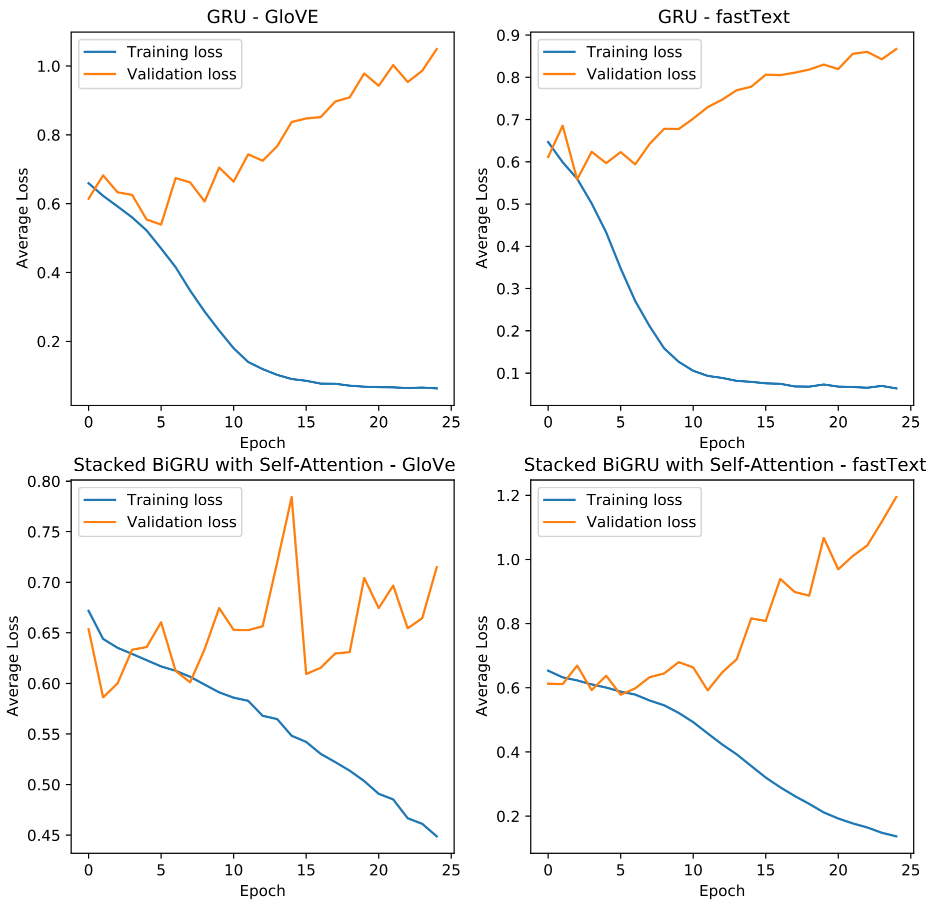

Figure 6: Neural Language Model Training and Validation Losses on Golbeck Corpus

6. Results and Discussion

6.1 Performance on the Golbeck Corpus

Looking at the table of results below, the best test set results in terms of weighted F1 belong to the GRU using GloVe embeddings with a weighted F1 of 73.83%, narrowly edging out the Stacked Bidirectional GRU with Attention and fastText embeddings. To put this number in perspective, the simple RNN models that only predict the majority class have a weighted F1 of 63.72%, so this should be considered the baseline. Both of these models strongly outperform the baseline and also outperform the stacked Bi-LSTM model with contextual attention used by Chakrabarty et al. (2019) on this dataset, which had a weighted F1 score of 72.75%, as well as many other previously state-of-the-art models the authors implemented in their analysis. Considering that the GRU architecture is fairly simple, I suspect that this great result is driven by the extensive preprocessing steps highlighted above that other researchers did not report implementing on this data.

The table also demonstrates the trade-offs each model makes. The Logistic Regression model has the highest test accuracy and precision, but this is mostly driven by conservative predicting behavior, as its low recall suggests that most harassing tweets are being missed. Meanwhile, the other Stacked Bi-GRU with Attention that uses GloVe embeddings is far and away the best at capturing harassing tweets with a recall over 42%, but this comes at the expense of much lower precision in its predictions.

Table 1: Test Set Classification Results on Golbeck Corpus

Classifier	Accuracy	Precision	Recall	F1	weighted F1
Logistic Regression (BOW)	**0.7620**	**0.5706**	0.2587	0.3560	0.7274
Simple RNN (GloVe)	0.7458	0.0000	0.0000	0.0000	0.6372
Simple RNN (fastText)	0.7458	0.0000	0.0000	0.0000	0.6372
GRU (GloVe)	0.7603	0.5475	0.3307	0.4123	**0.7383**
GRU (fastText)	0.7400	0.4823	0.3080	0.3759	0.7189
Stacked BiGRU with Attention (GloVe)	0.7037	0.4180	**0.4213**	0.4197	0.7041
Stacked BiGRU with Attention (fastText)	0.7553	0.5277	0.3560	**0.4252**	0.7379

6.2 Assessing Racial Bias on the Blodgett Corpus

Finally, I apply all of the non-RNN models to the Blodgett dataset to gauge whether or not my hypothesis is confirmed that the models will predict AAE-labeled tweets as harassing at a much higher rate. The first chart below plots the breakdown of predictions by dialect and label. We can see that the overwhelming majority of both AAE and White tweets are predicted to be non-harassing, but it is clear across all five models that there are many more AAE tweets predicted as harassing than there are White-labeled tweets.

The second plot shows the proportion of tweets predicted as harassing conditional on a user's dialect, and the results are even more damning. The Logistic Regression is over 3 times as likely to predict an AAE-aligned tweet as harassing than a White-aligned tweet; the factor is in the 2.2 to 2.5 range for the GRU models and a whopping 3.8 to 5.1 for the Self-Attention models. As Sap et al. (2019) note, verifying the *a priori* hypothesis that AAE and White English speakers are equally likely to use toxic language on Twitter requires a deeper dive into the sociological and linguistic research, but since AAE speakers are overwhelmingly Black, a toxic or hateful speech detector that yields such disparate results is both counterproductive and outright discriminatory. Testing the sensitivity of specific tokens is beyond the scope here, but previous work has found that words such as "n*gger" or “bitch” can be highly predictive of hate speech in these models, even though they have vastly different meanings and contexts depending on the racial identity and dialect of the speaker.

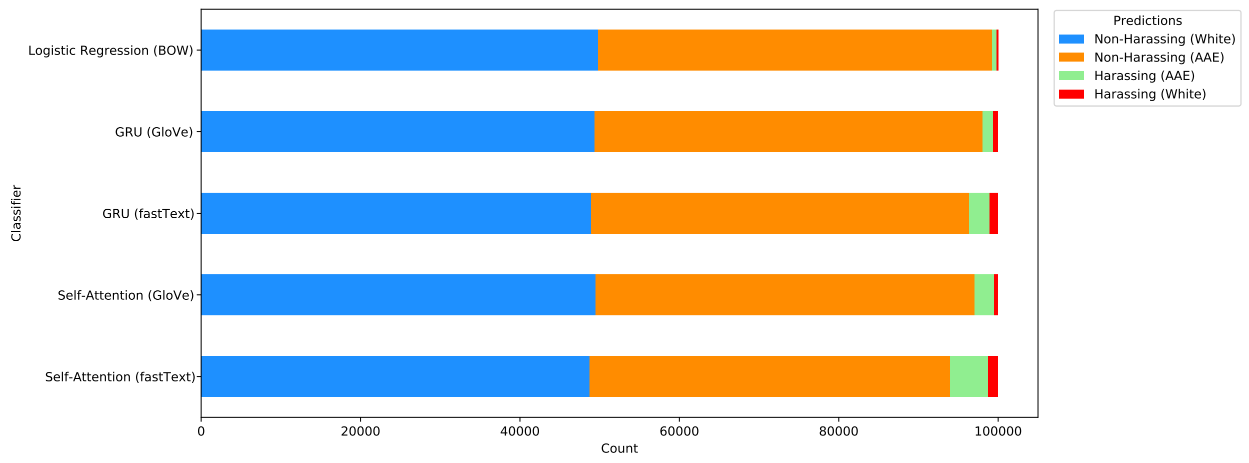

Figure 7: Model Prediction Frequencies on Blodgett Dataset

The results of this analysis should demonstrate the difficulties and trade-offs inherent in designing a toxic language detection system, as well as the multiple possible objectives. If the goal is to capture as many

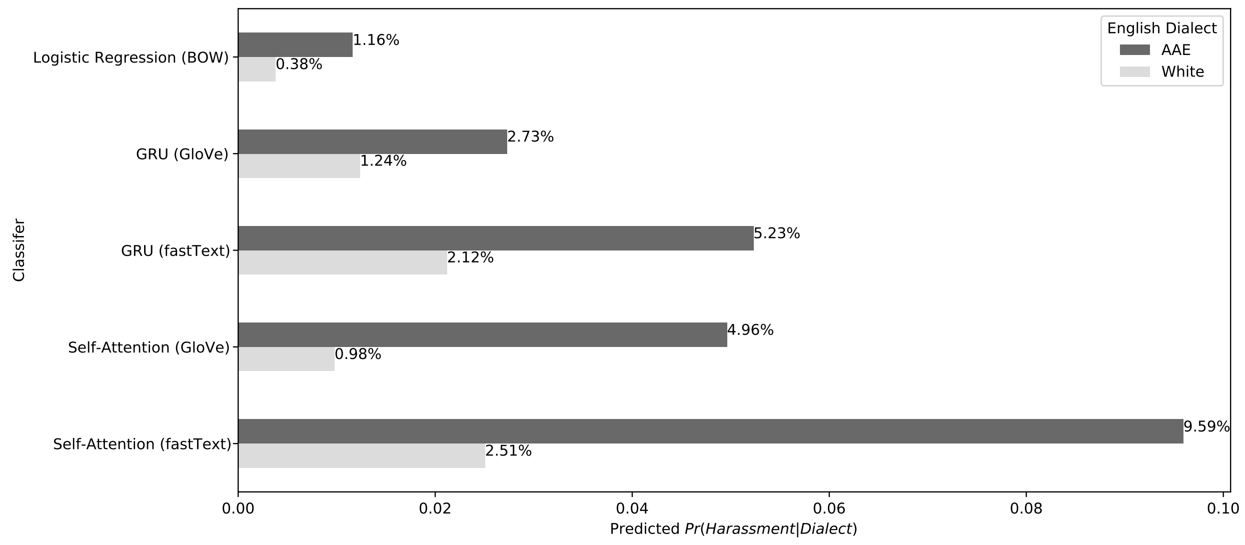

Figure 8: Conditional Probability of Tweet Predicted as Harassing Given English Dialect

harassing tweets as possible, even at the risk of wrongly censoring non-harassing tweets, then the high-recall Stacked Bi-GRU with Attention is a logical best choice. However, if these false positives are disproportionally AAE-aligned tweets, then the model is discriminating against a minority class. On the other hand, the Logistic Regression model is more precise and conservative in its predictions, but then the risk is that truly harassing language fails to be flagged by the model. Only with careful bias auditing, ideally with the ability to evaluate the fairness metrics defined above, can we move towards a classifier that is both effective and non-discriminatory. Clearly, despite the good performance of my best models in terms of weighted F1 score, that constraint is far from satisfied here.

7. Conclusion

In this analysis, I found that Twitter harassment detection is a difficult binary classification task, even while achieving a higher weighted F1 score using a GRU with GloVe embeddings than any of the past literature on the Golbeck corpus. More importantly, I found that even with careful consideration at each stage of the modeling process, my classifiers were far likelier to to predict an AAE-aligned tweet as harassing than a White-aligned tweet. It is possible that if this model were operationalized, it would end up doing more harm than good to the very group it seeks to protect from hate speech and toxic language. Further interdisciplinary research is required to ensure that these classifiers accomplish their objective without the unintended side effects demonstrated here.

7.1 Limitations

The clearest limitation of this analysis is the lack of a corpus with labels for both harassment and racial dialect of the user. The evidence presented in Section 6.2 is certainly indicative of the inherent racial biases present in all of the language models; however, a more complete auditing would require actual labels, an effort requiring considerable time, care and training.

As I discussed above, the use of validation accuracy in the neural language models rather than validation weighted F1 also represents a methodological issue. One possible workaround is to manually calculate the desired metrics using the weights at the end of every epoch. This would require saving all of the weights from dozens of epochs, which would likely incur high storage costs, but it is something worth looking into as an extension to this analysis.

Rather than apply my models to multiple corpora as much of the past literature does, I decided to spend extensive time understanding and processing this one dataset, so that I would be better positioned to discuss potential bias at *every* stage of the pipeline. This seems to have been what led to such impressive results on the Golbeck corpus relative to the literature, but it is also fair to question the external validity of models trained on a very specific, non-randomized corpus.

7.2 Next Steps

There are several next steps I would like to take that could both better highlight bias of neural language models and potentially improve their performance.

First, Bolukbasi et al. (2016), Zhou et al. (2019), and others have proposed methods of debiasing word embeddings that include the collapsing of embedding vectors to the axis within a certain “gender” or “race” subspace. These methods could be applied to both GloVe and fastText embeddings or even embeddings that I train myself. Another such method for training embeddings worth exploring is BERT (Bidirectional Encoder Representations from Transformers), in which tokens are associated with different embedding vectors for different contexts. This seems especially applicable to abusive language detection on Twitter, where words like "n*gga" can have different connotations depending on the identity of the user. To align with these contextual embeddings, adapting my Self-Attention models to use Contextual Attention is also worth doing.

Second, in order to address the limitation highlighted throughout this analysis of not having a corpus with both labels and racial identities, I would like to apply the lexical detection model made public by Blodgett et al. (2016) to a fully labeled corpus or even a combination of corpora. The issue in my analysis was that not enough tweets in the test set had inferred racial dialects with posterior probabilities of AAE or White that were at least 80% to constitute a decent sample, but with an entire large corpus that was not already used for training, this issue could be mitigated. This would allow for a full audit of model bias along racial lines following the metrics outlined above.

Third, once we can audit for bias, many pre-, in- and post-processing methods exist in the fairness literature for mitigating bias. I already applied a class-weighting scheme due to label imbalance, and the same can be done to ensure that a protected group, i.e. AAE-speaking Twitter users, is adequately represented by the model if they are in the minority. In terms of in-processing adjustments, Agarwal et al. (2018) propose adding a fairness constraint to a binary classifier’s objective function to create a cost-sensitive classification problem that can be solved using the exponentiated gradient algorithm. And finally, Hardt et al. (2016) walk through adjustments to classification thresholds that can be made after prediction using ROC curve analysis in order to satisfy fairness constraints such as independence, separation, and sufficiency that were described in Section 5.1.5. This analysis aimed for depth over breadth, and these are just some of the logical next steps to build upon what I have presented here.

References

Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J., & Wallach, H. (2018, July). A reductions approach to fair classification. In International Conference on Machine Learning (pp. 60-69). PMLR.

Barocas, S., Hardt, M., & Narayanan, A. (2019). Fairness and machine learning. fairmlbook.org. http://www.fairmlbook.org.

Blodgett, S. L., Green, L., & O’Connor, B. (2016). Demographic dialectal variation in social media: A case study of African-American English. arXiv preprint arXiv:1608.08868.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135-146.

Bolukbasi, T., Chang, K. W., Zou, J., Saligrama, V., & Kalai, A. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. arXiv preprint arXiv:1607.06520.

Caliskan, A., Bryson, J. J., & Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. Science, 356(6334), 183-186.

Chakrabarty, T., Gupta, K., & Muresan, S. (2019, August). Pay “attention” to your context when classifying abusive language. In Proceedings of the Third Workshop on Abusive Language Online (pp. 70-79).

Davidson, T., Bhattacharya, D., & Weber, I. (2019). Racial bias in hate speech and abusive language detection datasets. arXiv preprint arXiv:1905.12516.

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In Proceedings of the International AAAI Conference on Web and Social Media (Vol. 11, No. 1).

Gehman, S., Gururangan, S., Sap, M., Choi, Y., & Smith, N. A. (2020). Realtoxicityprompts: Evaluating neural toxic degeneration in language models. arXiv preprint arXiv:2009.11462.

Golbeck, J., Ashktorab, Z., Banjo, R. O., Berlinger, A., Bhagwan, S., Buntain, C., … & Wu, D. M. (2017, June). A large labeled corpus for online harassment research. In Proceedings of the 2017 ACM on web science conference (pp. 229-233).

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.

Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). Ieee.

Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. arXiv preprint arXiv:1610.02413.

Jurafsky, D., & Martin, J. H. (2020). Speech and Language Processing (draft).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Sap, M., Card, D., Gabriel, S., Choi, Y., & Smith, N. A. (2019, July). The risk of racial bias in hate speech detection. In Proceedings of the 57th annual meeting of the association for computational linguistics (pp. 1668-1678).

Sun, T., Gaut, A., Tang, S., Huang, Y., ElSherief, M., Zhao, J., … & Wang, W. Y. (2019). Mitigating gender bias in natural language processing: Literature review. arXiv preprint arXiv:1906.08976.

Waseem, Z., & Hovy, D. (2016, June). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In Proceedings of the NAACL student research workshop (pp. 88-93).

Williams, M. L., Burnap, P., Javed, A., Liu, H., & Ozalp, S. (2020). Hate in the machine: Anti-Black and anti-Muslim social media posts as predictors of offline racially and religiously aggravated crime. The British Journal of Criminology, 60(1), 93-117.

Wulczyn, E., Thain, N., & Dixon, L. (2017, April). Ex machina: Personal attacks seen at scale. In Proceedings of the 26th international conference on world wide web (pp. 1391-1399).

Zhou, P., Shi, W., Zhao, J., Huang, K. H., Chen, M., & Chang, K. W. (2019). Analyzing and Mitigating Gender Bias in Languages with Grammatical Gender and Bilingual Word Embeddings. ACL: Montréal, QC, Canada.